

Un Petit Guide d'utilisation du logiciel MATLAB

Par

Abdellatif EL GHAZI

Email : elghazi@msn.com

et

Saïd EL HAJJI

Email : elhajji@fsr.ac.ma

Université Mohammed V – Agdal

Faculté des Sciences

Département Mathématiques et Informatique

BP 1014, Rabat, Maroc

Page web : <http://www.fsr.ac.ma/ANO/elhajji>

Plan

1. Introduction
2. Les vecteurs
3. Les matrices
4. Les boucles
5. Le graphisme
6. Les procédures
7. Les fonctions
8. Les matrices creuses

Introduction :

Ce Guide d'utilisation du logiciel MATLAB est essentiellement une adaptation d'un guide Matlab rédigé par S. El Hajji pour les étudiants en Informatique (I3) promotions 2000-2001, 2001-2002 et 2002-2003 de la Faculté des Sciences de Rabat, des formations Matlab faites par A. Eberhard (Université de Grenoble) en 1999 et par A. Nachaoui (Université de Nantes) en 2001, des pages de TP écrites par Kelly Black de l'université du New Hampshire <http://www.math.unh.edu/~mathadm/tutorial/software/Matlab/> et ...

Puisque de nombreuses procédures existent déjà dans Matlab, on va utiliser Matlab comme logiciel de programmation. Le langage de base est proche des notations utilisées en algèbre linéaire et analyse numérique, mais il faut connaître quelques extensions de syntaxe que nous allons introduire dans ce Guide d'utilisation du logiciel MATLAB. On commencera par les notions les plus simples sur les vecteurs et les matrices pour arriver jusqu'à la programmation.

Les vecteurs

●Pour créer un vecteur (0, 2, 4, 6, 8) on tape:

➤ **[0,2,4,6,8]**

ans =

0 2 4 6 8

●Ou bien (on va de 0 à 8 avec un pas égal à 2)

➤ **0:2:8**

ans =

0 2 4 6 8

Les vecteurs

●Pour obtenir la transposée du précédent vecteur :

➤ **ans'**

ans =

0

2

4

6

8

●Pour créer le vecteur ligne v :

➤ **v = [0:2:8]**

v =

0 2 4 6 8

➤ **v'**

ans =

0

2

4

6

8

●Pour afficher les 3 premières composantes de v :

➤ **v(1:3)**

ans =

0 2 4

●Pour afficher les composantes 1 et 3 de v :

➤ **v(1:2:3)**

ans =

0 4

● Vous pouvez aussi utiliser un incrément négatif pour définir un vecteur en partant de la dernière composante.

```
>v=[0,-2,-4,-6,-8]
```

```
v =
```

```
0 -2 -4 -6 -8
```

```
> v=[0:-2:-8]
```

```
v =
```

```
0 -2 -4 -6 -8
```

```
>v(1:3)-v(2:4)
```

```
ans =
```

```
2 2 2
```

La commande *clear* permet d'effacer toutes les variables

```
>clear
```

```
>v
```

```
??? Undefined function or variable 'v'
```

Les matrices

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

```
A =
```

```
1 2 3
3 4 5
6 7 8
```

```
>> B = [ [1 2 3]' [2 4 7]' [3 5 8]']
```

```
B =
```

```
1 2 3
2 4 5
3 7 8
```

● La commande *whos* : Pour avoir la taille et la liste des variables utilisés

```
>> whos
```

Name	Size	Bytes	Class
A	3x3	72	double array
B	3x3	72	double array

Grand total is 18 elements using 144 bytes

Les matrices

```
>> A(1:2,3:4)
```

```
??? Index exceeds matrix dimensions.
```

```
>> A(1:2,2:3)
```

```
ans =
```

```

      2  3
      4  5
>> A(1:2,2:3)'
ans =
      2  4
      3  5

```

```

>> v = [0:2:8]
v =
      0      2      4      6      8
>> A*v(1:3)
??? Error using ==> *
Inner matrix dimensions must agree.
>> A*v(1:3)'
ans =
      16
      28
      46

```

Les matrices (Inverse)

```

>> inv(B)
ans =
    -3.0000    5.0000   -2.0000
    -1.0000   -1.0000    1.0000
     2.0000   -1.0000     0

```

```

>> C = [1 1 1; 2 1 2; 0 0 0]
C =
      1      1      1
      2      1      2
      0      0      0

```

```

>> inv(C)
Warning: Matrix is singular to working precision.
ans =
      Inf      Inf      Inf
      Inf      Inf      Inf
      Inf      Inf      Inf

```

```

>> inv(A)
Warning: Matrix is close to singular or badly scaled.

```

Results may be inaccurate. RCOND = 4.565062e-18

```

ans =
      1.0e+15 *
    -2.7022    4.5036   -1.8014
     5.4043   -9.0072    3.6029

```

-2.7022 4.5036 -1.8014

Or si on calcule le déterminant de A :

```
>> det(A)
```

ans =

0

Résolution de systèmes linéaires

On cherche à calculer la solution du système $Bx = v$ avec

```
>> v = [1 3 5]'
```

v =

1

3

5

```
>> x=B\v
```

x =

2

1

-1

```
>> y = A\v
```

Warning: Matrix is close to singular or badly scaled.

Results may be inaccurate. RCOND = 4.565062e-18

y =

1.0e+15 *

1.8014

-3.6029

1.8014

```
>> x1 = v'/B
```

x1 =

4.0000 -3.0000 1.0000

```
>> x1*B
```

ans =

1.0000 3.0000 5.0000

Attention : la commande A/B effectue l'opération $A \cdot \text{inv}(B)$ alors que la commande $A \backslash B$ effectue la commande $\text{inv}(A) \cdot B$.

Les valeurs propres et Les vecteurs propres

```
>> eig(A)
```

ans =

```

    14.0664
    -1.0664
    0.0000
>> [v,e] = eig(A)
v =
    -0.2656    0.7444   -0.4082
   -0.4912    0.1907    0.8165
   -0.8295   -0.6399   -0.4082

e =
    14.0664     0     0
     0     -1.0664     0
     0     0     0.0000

```

Les opérations sur les matrices et les vecteurs

```

>> v = [1 2 3]';size(v)
      ans =
           3     1

```

```

>> v = [1 2 3];size(v)
      ans =
           1     3

```

```

>> length(v)
      ans =
           3

```

```

>> a=1; size(a)
      ans =
           1     1

```

ADDITION, SOUSTRACTION

```

>> v = [1 2 3]'

v =

     1
     2

```

3

ou bien

```
>> v = [1; 2; 3]
```

```
>> b = [2 4 6]'
```

```
b =
```

2

4

6

```
>> v+b
```

```
ans =
```

3

6

9

MULTIPLICATION

```
>> v*b
```

```
Error using ==> *
```

```
Inner matrix dimensions must agree.
```

```
>> v*b'
```

```
ans =
```

2 4 6

4 8 12

6 12 18

```
>> v'*b
```

```
ans =
```

28

Multiplier chaque élément du vecteur v par l'élément correspondant du vecteur b

```
>> v.*b
```

```
ans =
```

```
2
8
18
```

Diviser chaque élément du vecteur v par l'élément correspondant du vecteur b

```
>> v./b
```

```
ans =
```

```
0.5000
0.5000
0.5000
```

Pour des matrices:

```
>> A=[1 2; 3 4], B=[1 -1; 2 2]
```

```
A =
```

```
1 2
3 4
```

```
B =
```

```
1 -1
2 2
```

```
>> A*B, A.*B
```

```
ans =
```

```
5 3
11 5
```

```
ans =
```

```
1 -2
6 8
```

```
>> A/B, A./B
ans =
    -0.5000    0.7500
    -0.5000    1.7500

ans =
    1.0000   -2.0000
    1.5000    2.0000
```

CONCATENATION

```
>> [A,B]
ans =
     1     2     1    -1
     3     4     2     2
>> [A B]
ans =
     1     2     1    -1
     3     4     2     2
```

```
>> [A ;B]
ans =
     1     2
     3     4
     1    -1
     2     2
```

FONCTIONS DES MATRICES

```
>> A=[0.5 0.2; -0.1 0.3]
A =
    0.5000    0.2000
   -0.1000    0.3000
>> EA=exp(A), EMA=expm(A)
EA =
```

```
1.6487 1.2214
0.9048 1.3499
```

EMA =

```
1.6333 0.2979
-0.1489 1.3354
```

```
>> log(EA), logm(EMA)
```

ans =

```
0.5000 0.2000
-0.1000 0.3000
```

ans =

```
0.5000 0.2000
-0.1000 0.3000
```

Programmer sous MATLAB

- Scripts et fonctions.
- Opérateurs de comparaison .
- Opérateurs logiques .
- Instructions de contrôle .

Scripts et fonctions :

Un **script** est un ensemble d'instruction MATLAB qui joue le rôle de programme principal. Si le *script* est écrit dans le fichier de nom *nom.m* on l'exécute dans la fenêtre MATLAB en tapant après « >> » nom .

Function[vars1 ,...,varsm]=fonc(vare1,...varen)

Séquence d'instructions

Où : *vars1 ,...,varsm* sont les variables de sortie de la fonction

vare1,...varen sont les variables d'entrée de la fonction

Séquence d'instructions est le corps de la fonction.

Il est impératif que la fonction ayant pour nom *fonc* soit enregistrée dans un fichier de nom *fonc.m* sans quoi cette fonction ne sera pas « visible » par MATLAB.

Opérateurs de comparaison

- Les opérateurs de comparaison sont :
- `=` : égal à (`x=y`)
- `>` : strictement plus grand que (`x>y`)

- $<$: strictement plus petit que ($x < y$)
- $>=$: plus grand ou égal à ($x >= y$)
- $<=$: plus petit ou égal à ($x <= y$)
- $\sim$: différent de ($x \sim y$)

Les opérateurs logiques sont :

- $\&$: et ($x \& y$)
- $|$: ou ($x | y$)
- $\sim$: non ($\sim x$)

Instructions de contrôle

- Boucle **for** (parcours d'un intervalle)
- Boucle **while** (tant que...faire)
- L'instruction conditionnée **if**
- Choix ventilé , l'instruction **switch**

Boucle for

●Syntaxe :

for indice = borne_inf : borne_sup
Séquence d'instructions
end

●Où *indice* est une variable appelée l'indice de la boucle

●*Borne_inf* et *borne_sup* sont deux constantes

●On peut utiliser un incrément (*pas*) autre que 1. La syntaxe est alors *Borne_inf* : *pas* : *borne_sup*.

Boucle while

Syntaxe :

while expression logique
Séquence d'instructions
end

expression logique est une expression dont le résultat peut être vrai ou faux

séquence d'instructions est le traitement à effectuer tant que *expression logique* est vraie.

L'instruction conditionnée IF

Syntaxe :

if expression logique
séquence d'instructions
end

expression logique est une expression dont le résultat peut être vrai ou faux

Il n'y a pas de mot clé « then »

L'instruction conditionnée IF

Syntaxe :

if expression logique
séquence d'instructions 1
else

séquence d'instructions 2
end

L'instruction conditionnée IF

Il est possible d'effectuer un choix en cascade :

Syntaxe :

if *expression logique 1*

séquence d'instructions 1

elseif *expression logique 2*

séquence d'instructions 2

...

elseif *expression logique N*

séquence d'instructions N

else *séquence d'instructions par défaut*

end

L'instruction switch

● **Syntaxe :**

switch *var*

case *cst1,*

séquence d'instructions 1

case *cst2,*

séquence *d'instructions 2*

...

case *cstN,*

séquence d'instructions N

otherwise *séquence d'instructions par défaut*

end

L'instruction switch

var est une variable numérique ou une variable chaîne de caractères

cst1,...,cstN, sont des constantes numérique ou des constantes chaîne de caractères

séquence d'instructions i est une séquence d'instructions à exécuter si le contenu de la variable *var* est égal à la constante *csti* (*var* = *csti*).

L'instruction switch

Il est possible de regrouper plusieurs « cas » si la séquence d'instructions à exécuter est la même pour ces différents cas. La syntaxe est alors :

Case{ *cst1,...,cstN*}

Séquence d'instructions commune

Graphisme

Gestion des fenêtres graphiques

Graphisme 2D

Améliorer la lisibilité d'une figure

Les entrées – sorties

Les formats d'affichage des réels

Lecture

Gestion des fenêtres graphiques

Une instruction graphique ouvre une fenêtre dans laquelle est affiché le résultat de cette commande.

Par défaut, une nouvelle instruction graphique sera affichée dans la même fenêtre et écrasera la figure précédente.

On peut ouvrir une nouvelle fenêtre graphique par la commande *figure*.

Chaque fenêtre se voit affecter un numéro *n*.

Ce numéro est visible dans le bandeau de la fenêtre sous forme d'un titre.

Le résultat d'une instruction graphique est par défaut affiché dans la dernière fenêtre graphique ouverte

On rend active une fenêtre graphique précédemment ouverte en exécutant la commande *figure(n)*

La commande *close* permet de fermer la fenêtre graphique active.

On ferme une fenêtre graphique précédemment ouverte en exécutant la commande *close(n)*

Il est également possible de fermer toutes les fenêtres graphiques en tapant *close all*.

Graphisme 2D

la commande *fplot*

```
>>fplot('nomf', [xmin , xmax])
```

où : *nomf* est le nom d'une fonction MATLAB incorporée, soit une expression définissant une fonction de la variable *x*, soit le nom d'une fonction utilisateur.

[*xmin* , *xmax*] est l'intervalle pour lequel est tracé le graphe de la fonction.

```
>>fplot('sin',[-2*pi 2*pi])
```

Pour tracer le graphe de la fonction $h(x) = x \sin(x)$ on peut définir la fonction utilisateur *h* dans le fichier *h.m* de la manière suivante :

```
function y=h(x)
```

```
y=x.*sin(x);
```

On obtient alors le graphe de la fonction *h* par l'instruction :

```
>>fplot('h',[-2*pi 2*pi])
```

L'autre façon de procéder est d'exécuter l'instruction :

```
>>fplot(' x* sin(x) ',[-2*pi 2*pi])
```

Il est possible de tracer plusieurs fonctions sur la même figure. Il faut pour cela utiliser la commande *fplot* de la manière suivante:

```
fplot('[nomf_1 , nomf_2 , nomf_3]', [xmin,xmax])
```

où *nomf_1* , *nomf_2* , *nomf_3* est le nom d'une fonction MATLAB incorporée, soit une expression définissant une fonction de la variable *x*, soit le nom d'une fonction utilisateur.

Pour limiter le graphe aux ordonnées comprises entre les valeurs *ymin* et *ymax* on passera comme second argument de la commande *fplot* le tableau [*xmin,xmax,ymin,ymax*].

Une autre possibilité pour gérer les bornes des valeurs en ordonnées est d'utiliser la commande *axis* après utilisation de la commande *fplot*.

La syntaxe est *axis([xmin, xmax, ymin, ymax])*.

```
>> fplot('sin(x)/x , cos(x)/x', [-5, 5, -1, 1])
```

La commande *plot* permet de tracer un ensemble de points de coordonnées (x_i, y_i) , $i=1, \dots, N$. La syntaxe est *plot(x,y)* où *x* est le vecteur contenant les valeurs x_i en abscisse et *y* est le vecteur contenant les valeurs y_i en ordonnée.

Les vecteurs *x* et *y* doivent être de même dimension mais il peut s'agir de vecteurs lignes ou colonnes.

Par défaut, les points (x_i, y_i) sont reliés entre eux par des segments de droites.

Pour tracer le graphe de la fonction

$h(x) = x \sin(x)$

```
>> x = [-2*pi:0.01:2*pi]; y = x.*sin(x); >> plot(x,y)
```

```
>> x = [-2*pi:1:2*pi]; y = x.*sin(x); >> plot(x,y)
```

On peut spécifier la couleur d'une courbe, le style de trait et/ou le symbole à chaque point (x_i, y_i) .

On donne un troisième paramètre d'entrée à la commande *plot* qui est une chaîne de 3 caractères de la forme '*cst*' :

c désignant la *couleur* du trait

s le *symbole* du point

t le *type* de trait.

y -jaune	. point	- trait plein
m magenta	o cercle	: pointillé court
c cyan	x marque	x - pointillé long
r rouge	+ plus	-. pointillé mixte
g vert	* étoile	< triangle (gauche)
b bleu	s carré	> triangle (droit)
w blanc	d losange	p pentagone
k Noir	v triangle (bas)	^ triangle (haut)

Les valeurs par défaut sont $c = b$, $s = .$ et $t = -$ ce qui correspond à:

Un trait plein

Bleu

Il n'est pas obligatoire de spécifier chacun des trois caractères

La commande *grid* permet d'obtenir un quadrillage de la figure

Il est possible de tracer plusieurs courbes sur la même figure en spécifiant plusieurs tableaux $x_1, y_1, x_2, y_2, \dots$, comme paramètres de la commande *plot*.

Si l'on souhaite que les courbes aient une apparence différente, on utilisera des options de couleurs et/ou de styles de traits distincts après chaque couple de vecteurs x, y .

On trace sur l'intervalle $[-5, 5]$ la fonction $x^2 \cos(x)$ en trait plein bleu et la fonction $x \cos(x)$ en trait pointillé rouge.

```
>> x = [-5:0.01:5]; >> y = x.^2.*cos(x); z = x.*cos(x);
```

```
>> plot(x,y,'b-',x,z,'r:');
```

la commande *loglog(x,y)* permet d'afficher le vecteur *log(x)* contre le vecteur *log(y)*.

La commande *loglog* s'utilise de la même manière que la commande *plot*.

```
>> x = [1:10:1000]; y = x.^3;
```

```
>> loglog(x,y)
```

Semologx = graphisme avec échelle log sur l'axe des x seul

Semology = graphisme avec échelle log sur l'axe des y seul

Améliorer la lisibilité d'une figure

Maquillage (habillage, légendes) d'une figure :

La commande *xlabel* permet de mettre un texte en légende sous l'axe des abscisses.

```
>> xlabel(' légende ')
```

La commande *ylabel* fait de même pour l'axe des ordonnées. La commande *title* permet de donner un titre à la figure.

```
>> title('le titre').
```

On peut écrire un texte donné à une position précise sur la figure grâce à la commande *text*.

```
text(posx , posy, ' un texte ')
```

posx et *posy* sont les coordonnées du point.

La commande *gtext* permet quant à elle de placer le texte à une position choisie sur la figure à l'aide de la souris.

```
gtext('un texte ').
```

```
>> P = 5; >> t = [0:.01:2];
```

```
>> c = 12*exp(-2*t) - 8*exp(-6*t);
```

```
>> plot(t,c); grid
```

```
>> xlabel('temps en minutes')
```

```
>> ylabel('concentration en gramme par litre')
```

```
>> title(['evolution de la concentration du produit ', num2str(P), ... ' au cours du temps '])
```

```
>> gtext('concentration maximale')
```

Afficher plusieurs courbes dans une même fenêtre

la commande *hold on* permet d'afficher plusieurs courbes dans une même fenêtre

Pour rétablir la situation antérieure (le résultat d'une nouvelle instruction graphique remplace dans la fenêtre graphique le dessin précédent) on tapera *hold off*.

```
>> e = exp(1);
```

```
>> figure
```

```
>> hold on
```

```
>> fplot('exp',[-1 1])
```

```
>> fplot('log',[1/e e])
```

```
>> plot([-1:0.01:e],[-1:0.01:e])
```

```
>> grid
```

```
>> hold off
```

la commande *subplot*. décompose une fenêtre en sous-fenêtres et d'afficher une figure différente sur chacune de ces sous-fenêtres

subplot(m , n , i)

m est le nombre de sous-fenêtres verticalement

n est le nombre de sous-fenêtres horizontalement;

i sert à spécifier dans quelle sous-fenêtre doit s'effectuer l'affichage.

>> *figure*

>> *subplot(2,3,1), fplot('cos',[0 4*pi]), title('cosinus'), grid*

>> *subplot(2,3,2), fplot('sin',[0 4*pi]), title('sinus'), grid*

>> *subplot(2,3,3), fplot('tan',[-pi/3 pi/3]), title('tangente'), grid*

>> *subplot(2,3,4), fplot('acos',[-1 1]), title('arc-cosinus'), grid*

>> *subplot(2,3,5), fplot('asin',[-1 1]), title('arc-sinus'), grid*

>> *subplot(2,3,6), fplot('atan',[-sqrt(3) sqrt(3)]), title('arc-tangente'), grid*

Sauvegarder une figure

La commande *print* permet de sauvegarder la figure d'une fenêtre graphique dans un fichier sous divers formats d'images.

>> *print -f<num> -d<format> <nomfic>*

<num> désigne le numéro de la fenêtre graphique.

<nomfic> est le nom du fichier dans lequel est sauvegardée la figure.

<format> est le format de sauvegarde de la figure.

Ces formats sont nombreux. On pourra obtenir la liste complète en tapant *help plot*.

ps : PostScript noir et blanc

psc: PostScript couleur

eps: PostScript Encapsulé noir et blanc

epsc : PostScript Encapsulé couleur

jpeg : Format d'image JPEG

tiff : Format d'image TIFF

Les entrées – sorties

Les formats d'affichage des réels

format long :format long à 15 chiffres.

format short e:format court à 5 chiffres avec notation en virgule flottante.

format long e:format long à 15 chiffres avec notation en virgule flottante.

>> *p1ans = 3.1416*

>> *format long>> pi*

>> *format short e*

>> *pi^3*

>> *format short g*

>> *pi^3*

>> format short

Lecture

La commande input permet de demander à l'utilisateur d'un programme de fournir des données.

```
var = input(' une phrase ').
```

Une phrase est affichée et MATLAB attend que l'utilisateur saisisse une donnée au clavier.

Cette donnée peut être une valeur numérique ou une instruction MATLAB.

Un retour chariot provoque la fin de la saisie.

Une valeur numérique est directement affectée à la variable var

Une instruction MATLAB est évaluée et le résultat est affecté à la variable var.

Il est possible de provoquer des sauts de ligne pour aérer le présentation en utilisant le symbole \n

```
var = input("\n une phrase : \n '").
```

Pensez à mettre un point virgule (;) à la fin de l'instruction si vous ne souhaitez pas voir s'afficher var = .

Pour saisir une réponse de type chaîne de caractères

```
var = input(' une phrase ','s').
```

Signalons qu'un retour chariot (sans autre chose) initialise la variable var au tableau vide

```
rep = input(' Affichage du resultat ? o/n [o] ','s');  
if isempty(rep), rep = 'o'; end  
if rep == 'o' | rep == 'y'  
disp(['Le resultat vaut ', num2str(res)])  
end
```

Références :

1. Eberhard A.(LMC, Grenoble France) : Formation MATLAB. Ecole sur Algèbre Linéaire et Application. FSR et INPT, Rabat, Septembre 1999.
2. El Hajji S. : Guide MATLAB , FSR, 2003.
(<http://www.fsr.ac.ma/ANO/elhajji>)
3. Lapreste J.T. : Introduction à MATLAB ; Ellipses.
4. Kelly Black, Université du New Hampshire
(<http://www.math.unh.edu/~mathadm/tutorial/software/Matlab/>).
5. MATLAB Reference Guide. MathWorks
6. MATLAB User's guide. MathWorks.
7. Mokhtari et all : Apprendre et Maîtriser MATLAB. Springer Verlag
8. Nachaoui A. (Université de Nantes, France) : Formation MATLAB. Kénitra, 2001.