Introduction à MATLAB

Nicolas KIELBASIEWICZ*

30 juillet 2007

MATLAB est un logiciel payant développé par MathWorks sous Windows, Linux et Mac, et dédié au calcul numérique, ainsi qu'au problèmes de commande. Pour l'acheter et obtenir de plus amples informations, consulter :

http ://www.mathworks.fr

MATLAB dispose d'un langage de programmation basé essentiellement sur le calcul matriciel, avec des fonctionnalités mathématiques et graphiques étendues.

L'objectif de ce document n'est pas de donner une liste exhaustive des diverses fonctionnalités de MATLAB , mais plutôt de donner un aperçu des commandes de base et des fonctions les plus couramment utilisées.

Table des matières

*Unité de Mathématiques Appliquées, École Nationale Supérieure de Techniques Avancées

1 Premiers contact avec MATLAB

1.1 La Fenêtre MATLAB

Pour démarrer, il suffit de cliquer sur l'icône de MATLAB ou de taper **matlab &** dans une fenêtre de terminal. Le programme se lance et une fenêtre graphique s'ouvre.

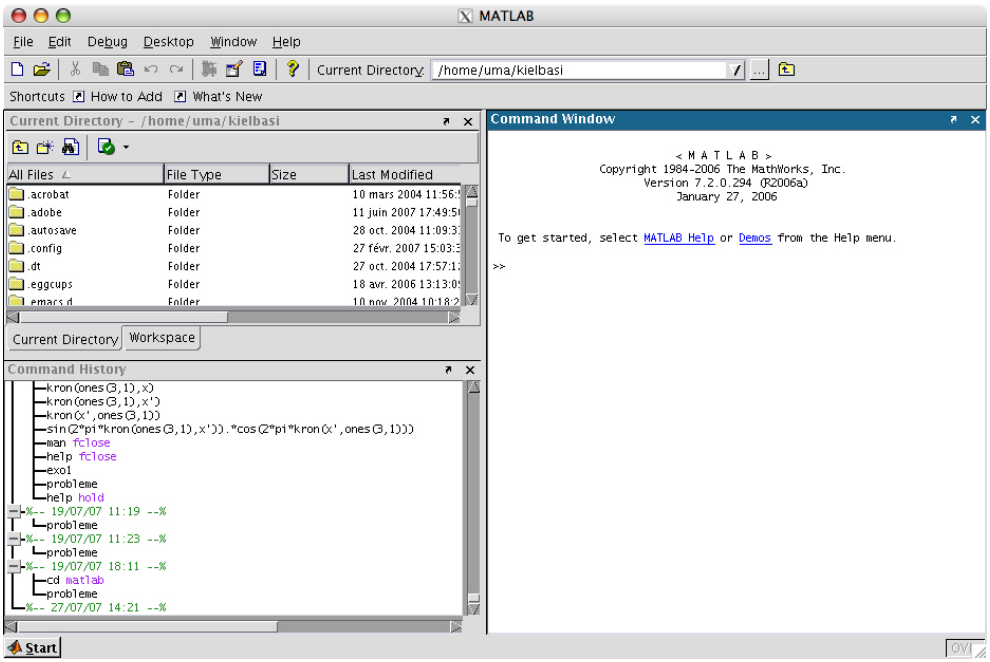

FIG. 1 – La fenêtre MATLAB

Cette fenêtre assez simple d'utilisation comporte un menu très standard. En dessous du menu, on distingue trois fenêtres distinctes. En haut à gauche, il s'agit de l'explorateur de fichiers, placé au niveau du répertoire actuel. Juste en dessous, on a l'historique des commandes exécutées. La partie droite est l'invite de commandes MATLAB dans laquelle on va taper les commandes et lancer les programmes.

1.2 Les fichiers MATLAB

Les fichiers MATLAB sont des fichiers texte dont l'extension est .m . Il en existe néanmoins deux types.

1.2.1 Les librairies

Un fichier de librairie est dédié à la définition d'une fonction. Pour des raisons pratiques, la fonction et le fichier ont le même nom. On ne peut définir qu'une seule fonction par fichier, et nous verrons plus loin comment.

Pour charger une librairie qui n'est pas dans le répertoire courant mais dans un autre chemin nom_chemin, on va utiliser la commande **addpath('nom_chemin')** pour ajouter le chemin dans la liste des répertoires que MATLAB va consulter à l'exécution d'une commande.

1.2.2 Les scripts

Ils sont exécutés par MATLAB en tapant simplement leur nom et contrairement aux fonctions, ils ne prennent pas d'arguments en entrée.

Ces fichiers contiennent une suite d'instructions. Ces instructions pourraient tout aussi bien être exécutés directement dans l'invite de commande MATLAB . On peut donc y effectuer des opérations d'entrées/sorties, des calculs, exécuter des commandes et des fonctions, ...

1.3 L'aide

Hormis le menu **help**, on peut utiliser la commande **help nom_fct** pour accéder à l'aide sur la comande voulue.

2 Les types de données

2.1 Les constantes spéciales

Voici la liste des commandes prédéfinies dans MATLAB :

NaN	Not-A-Number
Inf	l'infini
ans	la dernière réponse en date
eps	le zéro machine
i ou j	le nombre imaginaire
pi	le nombre π

Pour i et j, il est fortement déconseillé de les utiliser (en général, on les utilise comme indices de boucles usuels), même si on ne manipule pas des complexes.

2.2 Vecteurs et matrices

Pour définir un vecteur dont les valeurs sont uniformément réparties, il existe deux méthodes :
- **u=deb :pas :fin ;**
- **u=linspace(deb,fin,nbval) ;**

Pour définir un vecteur ou une matrice élément par élément, il faut procéder comme suit :
- Les crochets [] servent à encadrer la matrice ;
- un espace ou une virgule , sépare deux éléments d'une même ligne ;
- un point-virgule ; sépare les lignes entre elles.

Exemples :
- **[2 3 -5] ;** donne

```
ans =
     2  3  -5
```

- **[3 ;1] ;** donne

```
ans =
     3
     1
```

Les opérations matricielles sont définies dans le tableau suivant :

symbole	définition
[]	définition matricielle et concaténation
;	séparateur de colonne
()	extraction/insertion d'un élément
'	transposition
+	addition
-	soustraction
*	produit matriciel
\	division à gauche
/	division à droite
^	puissance
.*	produit élément par élément
.\	division à gauche élément par élément
./	division à droite élément par élément
.^	puissance élément par élément

3 Fonctions ou macros

Comme il a été dit précédemment, il est préférable de définir des fonctions dans des fichiers de librairies plutôt que dans les scripts eux-mêmes, et on ne peut définir qu'une seule fonction par fichier. Voici la syntaxe pour définir une fonction :

function [y1, y2, . . . , yn]=nom_fct(x1, x2, . . . , xp)
. . .
. . .
. . .

4 Boucles et instructions de contrôle

4.1 Les boucles

Il en existe de deux types en MATLAB : les boucles for et les boucles while.
Les boucles for s'écrivent de la façon suivante :

for i=deb :pas :fin
. . .
. . .
end

Quand le pas (qui est un entier relatif) n'est pas précisé, il est fixé à la valeur 1 par défaut.
Il existe une autre manière d'écrire une boucle for dont l'indice de boucle parcourt les éléments d'un vecteur v :

for i=v
. . .
. . .
end

On peut aller encore plus loin en affectant un vecteur colonne à notre indice de boucle. On utilise alors une matrice A, dont chaque colonne sera la valeur de l'indice de boucle, suivant la syntaxe suivante :

for i=A
...
...
end

Les boucles while s'écrivent de la façon suivante :

while *expression*
...
...
end

4.2 Les instructions de contrôle

Pour effectuer un test, on peut utiliser la combinaison classique if-then-else :

if *condition*
...
elseif
...
else
...
end

On peut également avoir à utiliser switch-case dans le cas où le test comprend au moins trois possibilités :

switch *var*
case *val1*
...
case *val2*
...
otherwise
...
end

4.3 Les opérateurs logiques

Voici la liste des opérateurs logiques servant entre autres à écrire les conditions :

==	égal à
∼=	différent de
<	inférieur à
<=	inférieur ou égal à
>	supérieur à
>=	supérieur ou égal à
&	et
|	ou
∼	non

5 Entrées/Sorties

5.1 Ouvrir un fichier

La commande la plus courante est :

fid=fopen('fichier.xxx',mode) ;

La variable *mode* peut être "r" (lecture) ou "w" (écriture).

5.2 Lire dans un fichier

La commande la plus courante est :

x=fscanf(fid,format,size) ;

La variable *format* est une chaine de caractères constituée de caractères du type %s (chaine de caractères sans espace), %d (entier), %f (flottant), ...

La variable *size* définit le nombre d'éléments à récupérer. Si $size = [m, n]$, alors on récupère une matrice de taille n x m. Si m est inf, alors on récupère tout jusqu'à la fin du fichier.

5.3 Ecrire dans un fichier

La commande la plus courante est :

x=fprintf(fid,format,a1, ..., an) ;

5.4 Fermer un fichier

fclose(fid) ; fclose('all') ; % pour fermer tous les fichiers actuellement ouverts.

6 Les graphiques

6.1 Tracés en 2D

6.1.1 plot : la commande de base

La commande **plot** est la fonction d'affichage graphique la plus élémentaire. Elle a subi des modifications substantielles depuis MATLAB 4.0 pour ressembler à son équivalent dans MATLAB. Elle permet maintenant la même gestion des couleurs et des styles de lignes.

valeur	correspondance	valeur	correspondance
-	ligne continue	r	rouge
–	ligne discontinue	g	vert
:	ligne pointillée	c	cyan
+	signes plus	m	magenta
o	cercles	y	jaune
*	astérisques	k	noir
.	points	w	blanc
x	croix		
s	carrés		
d	diamant		
p	pentagramme		
^	triangles vers le haut		
v	triangles vers le bas		
<	triangles vers la gauche		
>	triangles vers la droite		

6.1.2 Commandes plus spécifiques

- La commande **fplot** permet de tracer une courbe définie par une fonction. Ses arguments sont les mêmes que **plot** à ceci près que la matrice des ordonnées est remplacée par le nom de la fonction.
- D'autres fonctions : **bar**, **polar**, **errorbar**, **stream2** (pour les écoulements 2D), **quiver** (pour les champs de vitesses 2D), **contour** ...

6.2 Tracés en 3D

6.2.1 La commande surf

Elle sert à tracer des surfaces.

1. **surf(X, Y, Z, C) ;**
   - X, Y, Z et C sont 4 matrices de même taille (m,n). Si la matrice des couleurs C n'est pas précisé, alors $C = Z$.
   - La surface tracée est définie par l'ensemble des points de coordonnées X_{ij}, Y_{ij} et Z_{ij} et la couleur associée est C_{ij}.
2. **surf(x, y, Z, C) ;**
   - x et y sont des vecteurs de tailles nx et ny. Z et C sont des matrice de taille (ny,nx).
   - La surface tracée est définie par l'ensemble des points de coordonnées X_i, Y_j et Z_{ji}

Dans les deux cas, on peut paramétrer l'affichage avec les commandes **view**, **shading** et **axis**.

6.2.2 Autres commandes

- La commande **plot3** permet d'afficher des courbes paramétrées. Contrairement à **surf**, son troisième argument z n'est pas une matrice, mais un vecteur de la même dimension que x et y. On peut également l'utiliser pour afficher des nuages de points.
- La commande **pcolor** permet de tracer une surface vue de dessus. Elle a pour arguments X, Y et C. En choisissant $C = Z$, on obtient alors le même résultat qu'avec la commande **surf** avec une vue de dessus. Néanmoins, il est préférable dans ce contexte d'utiliser **pcolor**.
- D'autres fonctions : **quiver3** et **coneplot** (pour les champ de vitesses), **stream3** (pour les écoulements), **contour3** ...