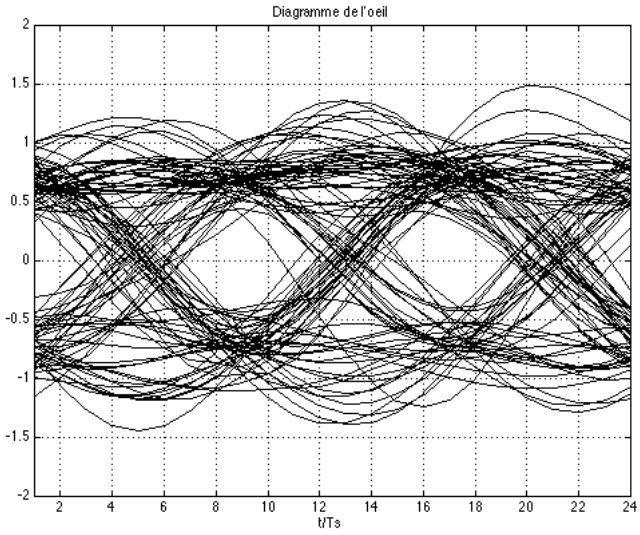
Petit guide d'introduction à MATLAB

K. Amis et C. Laot



Département Signal et Communications Version 2 le 06/09/04

1. Introduction

Ce document a pour but de donner les informations de base pour utiliser le logiciel Matlab. Matlab est un logiciel de programmation numérique. Il permet une visualisation efficace des variables utilisées soit directement dans l'espace de travail (workspace) soit par l'intermédiaire de graphiques et de figures. Son principal intérêt réside dans une prise en main rapide et un traitement matriciel à l'aide de nombreuses fonctions prédéfinies.

2. Démarrage

- Sous UNIX et Linux
Placez-vous dans le répertoire dans lequel vous souhaitez exécuter vos programmes. Puis lancez Matlab par la commande "matlab" dans le même xterm. En cas de problème, vérifier les versions de Matlab disponibles par la commande SETUP, puis changer de version de Matlab par un SETUP.

- Sous PC Windows et Mac
Cliquez sur l'icône Matlab et c'est parti !!

3. Principe de Matlab

Une fois démarré Matlab ouvre une fenêtre de commande. Dans son principe, Matlab permet de travailler soit à partir de la fenêtre de commande (une instruction à la fois), soit à partir d'un fichier source avec l'extension**.m** (M-files : plusieurs instructions).

Une instruction peut être directement tapée dans la fenêtre de commande. Chaque instruction de la fenêtre de commande est précédée du sigle **>>** (**prompt**). Une simple validation (**Return**) permet d'exécuter la ligne d'instruction.

Généralement dans un programme, on utilise un fichier source (M-file) qui contient une suite d'instructions. Chaque ligne comprend une instruction. Pour exécuter le programme contenu dans un fichier "source.m", il suffit de taper le nom du fichier dans la fenêtre de commande suivi de **Return**. Sous Windows, il est possible d'exécuter le programme à partir d'un menu déroulant en choisissant l'option "save and execute".

4. Informations générales

L'instruction **help** permet d'obtenir une description de l'ensemble des fonctions disponibles. **help** suivi du nom d'une fonction permet d'obtenir la description de la fonction. Attention, dans l'aide en ligne, le nom de la fonction est en majuscules, mais son utilisation requiert des minuscules.

Si vous souhaitez rechercher une fonction dont vous pensez connaître le nom, utilisez :
lookfor *"nom approximatif de la fonction"*.

Le site Web de Matlab peut s'avérer également très utile:
http://www.mathworks.com

À tout moment en tapant l'instruction **whos** il est possible de visualiser l'ensemble des variables de l'espace de travail. Pour afficher le contenu d'une variable, il suffit de taper le nom de la variable dans la fenêtre de commande suivi de **Return**. L'instruction **clear** permet d'effacer l'ensemble des variables de l'espace de travail ou une variable spécifiée.

Les instructions **load** et **save** permettent de charger ou de sauvegarder l'ensemble des variables ou quelques variables de l'espace de travail dans des fichiers **.mat.**

Les commentaires sont précédés du sigle **%**. Pour continuer une instruction sur plus d'une ligne utiliser trois points : **...**. Le sigle point virgule **;** en fin d'instruction évite l'affichage dans la fenêtre de commande du résultat de l'instruction.

5. Déclaration d'un vecteur ou d'une matrice

5.1 Vecteur de taille connue *n*

Vecteur ligne

x=[1 2 3 4 5 6];

x=zeros(1,n); % Déclaration de x et initialisation à 0_{1xn}

x=ones(1,n); % Déclaration de x et initialisation à 1_{1xn}

Vecteur colonne

x=[1;2;3;4;5;6];

x=zeros(n,1); % Déclaration de x et initialisation à 0_{nx1}

x=ones(n,1); % Déclaration de x et initialisation à 1_{nx1}

5.2 Vecteur de taille inconnue

x=[ ]; % Initialisation de x comme vecteur vide

Au cours d'un processus récursif, on peut insérer de nouvelles valeurs dans le vecteur x. Par exemple, on vient insérer le vecteur ligne y dans le vecteur x

x=[x y]; % Insertion de y à la fin de x

x=[y x]; % Insertion de y au début de x

x=[x(1,1:k-1) y x(k-1:end)]; % Insertion de y à partir de la k-ième composante de x

5.3. Déclaration d'une matrice

Si on connait la taille de la matrice et/ou ses éléments, on peut initialiser la matrice par les instructions suivantes :

X=[1 2 3; 4 5 6]; % X=$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

X=zeros(n,m) % Matrice nulle de taille $n \times m$

X=ones(n,m) % Matrice dont tous les éléments sont égaux à 1

X=eye(n) % Matrice identité de taille $n \times n$

La méthode décrite pour un vecteur de taille inconnue reste valable pour les matrices à condition de prendre quelques précautions sur les tailles des matrices insérées.

5.4. Exemples de déclarations de vecteurs et de matrices

A partir de la fenêtre de commande taper, l'instruction :

A=[1 2 3 ; 4 5 6 ; 7 8 9] **%** *Création d'une matrice avec affichage du résultat*

Après **Return**, Matlab réserve la mémoire nécessaire pour la matrice A et affiche sa valeur :

A=

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**

Pour éviter l'affichage de cette matrice dans la fenêtre de commande, il suffit d'ajouter en fin d'instruction un **point virgule** :

A=[1 2 3 ; 4 5 6 ; 7 8 9]; **%** *Création d'une matrice sans affichage du résultat*

On peut très facilement isoler des éléments de la matrice **A(ligne, colonne)**

Affectation à la variable a1 de l'élément de A situé ligne 3 colonne 2

a1=A(3, 2)

a1 =

8

Affectation à la variable a2 de la deuxième colonne de A

a2=A(:, 2)

a2 =

2
5
8

Affectation à la variable a3 de la première ligne de A

a3=A(1, :)

a3 =

1 2 3

Affectation à la variable a4 d'une matrice 2 lignes et 3 colonnes extraite de A

a4=A(1:2, :)

a4 =

1 2 3
4 5 6

Transforme la matrice a4 en un vecteur colonne a5 (lecture par colonne)

a5=a4(:)

a5 =

1
4
2
5
3
6

Transforme le vecteur a5 en la matrice a4 précédente

a4=reshape(a5,2,3);

Concaténation des vecteurs et des matrices

B=[A a2]

B=

1	**2**	**3**	**2**
4	**5**	**6**	**5**
7	**8**	**9**	**8**

X=0:0.1:1; *création d'une suite numérique allant de 0 à 1 avec un pas de 0.1*
X=0:1; *création d'une suite numérique allant de 0 à 1 avec un pas de 1*
A=zeros(3,3); *crée une matrice nulle*
A=ones(3,3); *crée une matrice dont tous les éléments sont à 1*
A=eye(3); *crée une matrice identité*

Il n'est pas obligatoire de définir une matrice avant de lui affecter une valeur, elle est créée automatiquement lors de l'affectation. Cependant si la taille de la matrice doit évoluer dans un programme, il est plus intéressant en termes de rapidité d'exécution de définir la taille de la matrice au préalable. En effet si la matrice n'est pas prédéfinie, Matlab doit à chaque nouveau changement de taille réallouer de la mémoire et l'on ralentit l'exécution du programme.

6. Opérations sur les vecteurs et les matrices

6.1 Opérations élémentaires sur les matrices

Dans le cas des opérations sur les matrices ou vecteurs, les dimensions doivent être appropriées et respecter l'algèbre matricielle classique.

Les opérations de base travaillent aussi bien sur les matrices que sur les scalaires.
+ - * / ^ (addition, soustraction, multiplication, division, puissance)

Il est tout à fait possible d'effectuer les opérations de multiplication, de division et de puissance, éléments par éléments entre deux matrices. L'opérateur est alors précédé d'un point **(.).** Les matrices ou vecteurs doivent avoir les mêmes dimensions**.**

.* ./ .^ (multiplication, division, puissance)

Les opérateurs d'égalité et d'inégalité sont respectivement == et ~=

Soit a_{pq} (resp b_{pq}) l'élement de la matrice A (resp. B) à l'intersection de la ligne p et de la colonne q. Les opérations élémentaires sur ces matrices sont données ci-dessous.

C = A * B % $c_{qp} = \sum_k a_{qk}.b_{kp}$
C = A. * B % $c_{qp} = a_{qp}.b_{qp}$
C = A. / B % $c_{qp} = a_{qp}/b_{qp}$

Matlab gère les nombres complexes. Il suffit pour cela d'écrire :
X=A+i*A *matrice composée d'éléments complexes* (**i^2=-1***)*

Remarque : On peut remplacer i par j. Il convient d'éviter d'utiliser i et j comme indices car ce sont des lettres réservées.

6.2 Quelques fonctions élémentaires sur les matrices :

real(X);	*partie réelle des éléments de X*
imag(X);	*partie imaginaire des éléments de X*
size(A)	*taille d'une matrice*
A.'	*transposée d'une matrice*
A'	*transposée conjuguée d'une matrice (transposée hermitienne)*
conj(A)	*conjugué d'une matrice*
sqrt(A)	*racine carré des éléments de la matrice*
min(A)	*valeur et position du plus petit élément de A*
max(A)	*valeur et position du plus grand élément de A*
sum(A,1)	*somme des lignes de la matrice*
sum(A,2)	*somme des colonnes de la matrice*
abs(A)	*valeur absolue des éléments de A*
sign(A)	*prend le signe des éléments de A*
rem(x,y)	*reste après la division de x par y*
fix(x)	*rend la partie entière la plus proche de zéro*
mod(x,y)	*fonction x modulo y*
floor(A)	*arrondi vers l'entier inférieur*
ceil(A)	*arrondi vers l'entier supérieur*
round(A)	*arrondi vers l'entier le plus proche*
fliplr(A)	*permutation de gauche à droite des éléments d'une matrice*
flipud(A)	*permutation de haut en bas des éléments d'une matrice*
find(A<0.5)	*recherche des éléments de A inférieurs à 0.5*
sort(X)	*tri des éléments de X par ordre croissant*
mean(X)	*valeur moyenne sur X*
std(X)	*écart type sur X*
prod(X)	*produit des éléments de X*
cumsum(X)	*somme cumulée des éléments de X*
eig(A)	*valeurs propres de la matrice A*

Autres fonctions mathématiques ::
pi, sin, cos, tan, cot, sinc, sinh, cosh, tanh, coth, exp, log, log10, log2, erfc, ...

7. Les boucle et les instructions conditionnelles

Comme dans une programmation classique les boucles et les instructions conditionnelles existent. Elles commencent par l'instruction **for, if, while** et sont suivies d'un ensemble d'instructions (corps de boucle) qui seront exécutées si la condition est validée puis se terminent obligatoirement par **end**. Les opérateurs binaires sont "&", "|", "~".

if *condition*
corps de boucle
else
corps de boucle
end

for n=1:0.1:10
corps de boucle
end

Remarque : chaque boucle sous Matlab coûte très cher en temps de calcul et il est nécessaire de les supprimer dès que cela est possible.

8. Graphiques et figures

figure(n)	*ouvre ou sélectionne la fenêtre n pour affichage d'une figure*
plot(X,'x')	*trace le vecteur X avec un x pour chaque point*
hist(X)	*tracé d'un histogramme des valeurs de X*
semilogy(X)	*trace un graphe en semilog (utile pour les probabilités d'erreurs)*
clf	*efface la figure courante*
close	*fermeture de la fenêtre courante*
close all	*fermeture de l'ensemble des fenêtres associées aux figures*
subplot(211)	*permet de diviser une fenêtre en plusieurs figures*
axis([0 1 0 2])	*permet de redéfinir la fenêtre de visualisation*
hold	*maintient d'une figure pour superposition d'une autre figure*
grid	*positionnement d'une grille*
xlabel('texte')	*permet de labelliser l'axe des abscisses*
ylabel('texte')	*permet de labelliser l'axe des ordonnées*
title('texte')	*permet de nommer la figure*

9. Fonctions élémentaires pour les communications numériques

rand	*variables uniformement distribuées entre 0 et 1*
randn	*variables distribuées selon une loi normale N(0,1)*
fft	*transformée de Fourier rapide*
ifft	*transformée de Fourier inverse rapide*
conv	*convolution de deux vecteurs*
filter	*filtrage numérique avec traitement des effets de bords*
freqz	*trace la fonction de transfert d'un filtre numérique*
psd	*estimation de la densité spectrale de puissance*
roots	*racines d'un polynome*
abs	*module*
angle	*phase*

10. Création et appel d'une fonction Matlab

Soit **FonctionAlpha** le nom de la fonction que l'on souhaite créer. Dans le répertoire dans lequel le programme principal est situé (le cas échéant il faut préciser le chemin pour atteindre la fonction), écrire un fichier source appelé **FonctionAlpha.m** (le nom du fichier et le nom de la fonction sont identiques). Le fichier source possède alors la structure suivante :

function [y1 y2 ... yn]=FonctionAlpha(x1,x2,...,xp)
Corps de la fonction;

Les **xi** sont les paramètres d'entrée et les **yi** sont les paramètres de sortie. La fonction ainsi définie s'utilise alors comme une fonction prédéfinie de Matlab.