MATLAB EN BREF

I. MATLAB

I.1 Généralités

I.1.1 Introduction

Matlab = abréviation de « *Matrix Laboratory* ».

Environnement informatique conçu pour le calcul matriciel, parfaitement adapté à l'Automatique et au Traitement du Signal :

— facilité d'emploi,

— possibilités d'affichages graphiques,

— toolboxes : signal processing, image processing, optimization, ...

I.1.2 Pour lancer Matlab

Cliquer sur l'icône Matlab ou taper « Matlab » dans un shell, puis taper les instructions Matlab ou le nom d'un fichier les contenant (à la suite des chevrons >>).

I.1.3 Langage interprété

```
>> 2+2
ans =     4
>> 2*sin(pi/4)
ans =     1.4142
```

Résultat affiché et stocké dans la variable *ans*.
Fonctions mathématiques usuelles : *sin*, *cos*, *exp*, ...
Constantes (*pi*, ...).

I.1.4 Variables

```
>> x = pi/4
x =   0.7854
```

Les noms de variable (< 19 caractères) commencent par une lettre.

Attention, Matlab voit les majuscules (`x` ≠ `X`).

Point virgule « ; » pour ne pas afficher le résultat.

Plusieurs commandes par ligne, séparées par un point virgule ou une virgule :

```
>> x = pi/2; y = sin(x);
```

Pour une ligne trop longue, utiliser « ... ».

I.1.5 Variables complexes

Matlab indifférent aux nombres réels / complexes.

`i` et `j` initialisés à la valeur complexe `sqrt(-1)`.

```
>> z = 3 + 2*i  % et même z = 3 + 2i
                % dans la version 4
z =   3.0000 + 2.0000i
```

Fonctions usuelles prédéfinies dans Matlab : *real*, *imag*, *abs*, *angle* (en radian), *conj*.

```
>> r = abs(z);
>> theta = angle(z);
>> y = r*exp(i*theta);
```

I.1.6 Vecteurs, matrices et leur manipulation

- Toute variable Matlab est une matrice.

On peut entrer une matrice sous forme d'un tableau :

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

Les éléments de matrice peuvent être des expressions quelconques à évaluer :

```
>> x = [-1.3 sqrt(3) (1+2+3)*4/5]
x =      -1.3000 1.7321 4.8000
```

Éléments de matrice référencés par leurs indices :

```
>> x(2)
ans =    1.7321
>> x(5) = abs(x(1))
x =      -1.300 1.732 4.800 0.000 1.300
```

... taille ajustée, éléments non précisés à 0.

Ajout de lignes et de colonnes à une matrice :

```
>> r = [7 8 9]; A = [A ; r]
A =      1 2 3
         4 5 6
         7 8 9
>> r = [ 0; 0; 0]; A = [A, r]
A =      1 2 3 0
         4 5 6 0
         7 8 9 0
```

- Fonctions *zeros*, *ones* et *eye* :

```
>> eye(2)      % eye comme I : Identity
ans =    1 0
         0 1
>> x = ones(1,5)
ans =    1 1 1 1 1
```

- Taille d'une matrice : *size*, *length* :

```
>> size(x)
ans =    1 5
```

- Transposée et transposée conjuguée :

```
>> A'       % Transposée conjuguée de A
>> A.'      % Transposée de A
```

- Retournement horizontal, vertical et rotation :

```
>> flipud(A)    % Up - Down
>> fliplr(A)    % Left - Right
>> rot90(A)     % Rotation de 90°
```

1.1.7 L'opérateur d'énumération « : »

-
```
>> x = 0.5:0.1:0.85
x =      0.5000 0.6000 0.7000 0.8000
```

Le pas d'incrémentation par défaut vaut 1 :

```
>> x = 1:5
x =      1 2 3 4 5
```

(voir aussi `linspace(deb,fin,npts)`)

- Sélection d'éléments d'un vecteur ou d'une matrice :

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

Si l'on a :

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

alors

```
>> A(m,n)
ans =    7 9
         4 6
         1 3
```

Mais il faut remarquer que

```
>> m = [3, 0, 2];    % m(2) = 0 !
>> A(m,:)
ans =    1 2 3
         7 8 9
```

donne les lignes de A dont l'élément de m est non nul (cest-à-dire ici la 1° et la 3° ligne). Vecteur pris comme un booléen !

1.1.8 Chaînes de caractères

- Variables contenant des chaînes de caractères :

```
>> mess = 'Bienvenue sur Matlab';
```

Manipulations de même type que pour les vecteurs :

```
>> mess = [mess ' v 4.1'];
mess = Bienvenue sur Matlab v 4.1
```

- Conversion de nombres en chaînes de caractères : *num2str*, *int2str*, *sprintf* :

```
>> mess = ['pi vaut ', num2str(pi)]
mess = pi vaut 3.142
```

- Évaluation d'une expression Matlab : *eval* et *feval* :

```
>> nom_var = 'x';
>> eval([nom_var '1 = sqrt(-1)'])
x1 =     0.0000 + 1.0000i
>> nom_fct = 'sin';
>> feval(nom_fct,pi)
ans =    1.2246e-16  % sin(pi)  0 !
```

1.2 Opérations matricielles

- Opérations usuelles définies de façon naturelle :

```
>> 2*A
>> A*B          % Produit matriciel
>> A^p
>> inv(A)
```

Résolution de systèmes linéaires :

```
>> X = A\B      % solution de A*X = B
>> X = B/A      % solution de X*A = B
```

Attention :

```
>> A.*B         % Produit terme à terme
>> X = A./B     % Division terme à terme
>> A.^p         % Puissance terme à terme
```

- Autres fonctions utiles sur les matrices :

```
>> poly(A)      % Polyn. caract. de A
>> det(A)       % Déterminant de A
>> trace(A)     % Trace de A
>> [V,D] = eig(A)  % Valeurs et
                   % vecteurs propres
```

- Pour certaines fonctions, matrice = tableau de valeurs :

```
>> exp(A); log(A); sqrt(A);
% Arrondi, signe et modulo :
>> round(A); sign(A); rem(A,2);
```

Pour calculer l'exponentiel, le logarithme ou la racine carrée d'une matrice : *expm*, *logm* et *sqrtm*.

- Pour certaines fonctions, matrice = suite de vecteurs colonnes

```
>> min(A);     % Vecteur ligne des
               % minima des colonnes
>> max(A);       % Maximum
>> sum(A);       % Somme
>> prod(A);      % Somme
>> cumsum(A);    % Sommes cumulés
>> cumprod(A);   % Produits cumulés
>> sort(A);      % Tri des colonnes
>> median(A);    % Val. médiane des col
```

- Matrices creuses : Gain en mémoire et en coût de calcul, e.g.,

```
>> A = eye(2); B = sparse(A);
>> A*[1;1];      % 4 multiplications
>> B*[1;1];      % 2 multiplications
```

1.3 Affichages graphiques et alpha-numériques

1.3.1 Affichage alpha-numérique

```
>> disp(mess)
pi vaut 3.142
>> fprintf('pi vaut %e \n',pi)
pi vaut 3.141593e+000
>> rep=input('Valeur de lambda : ');
        % attend l'entrée d'une valeur.
Valeur de lambda :
```

1.3.2 Graphiques 1D : *plot*, *title*, *xlabel*, *ylabel*

```
>> x = (0:0.1:2)*pi; y = sin(x*pi);
>> plot(x,y);      % abscisse, ordonnée
>> title('Courbe y = sinus(x)')
>> xlabel('x'); ylabel('y')
```

1.3.3 Graphiques 2D : *mesh* et *meshgrid*

```
>> x = 1:0.1:2; y = -1:0.1:1;
>> [X,Y] = meshgrid(x,y);
>> mesh(X,Y,cos(pi*X).*sin(pi*Y))
```

Mais aussi *meshc* et *contour*, *image*, *surf* ...

1.3.4 Affichage de plusieurs courbes

- La commande *subplot* divise la fenêtre graphique :

```
subplot(nb_lig,nb_col,num_subdiv)
```

```
>> subplot(2,2,1)
>> plot(x,y)
>> subplot(2,2,2)
>> plot(x,y.^2)
```

fig. 1	*fig. 2*
fig. 3	*fig. 4*

- La commande *figure* crée une nouvelle fenêtre graphique

I.4 Environnement, scripts et fonctions

I.4.1 Répertoire de travail

path :

```
>> path(path,'mon_chemin');
```

Commandes *cd* et *dir* (même syntaxe que MS-Dos).

Définitions à l'initialisation dans le fichier "*startup.m*".

I.4.2 Sauvegarde et chargement de variables

- Sauvegarde :

```
>> save nom_fichier noms_variables
```

Exemple :

```
>> save test.mat A x y
```

Par défaut sauvegarde dans *matlab.mat*.

- Chargement :

```
load nom_fichier
```

Commandes *who* et *whos*.

Commandes *pack* et *clear*.

I.4.3 Scripts

- Script = suite de commandes écrite dans un fichier.

Commentaires à l'aide du caractère '%'.

Sauver le fichier avec une extension .m (*nomfich*.m).

Exécuter en tapant le nom du fichier sous Matlab.

- Les variables définies dans l'espace de travail sont accessibles pour le script.

Réciproquement, les variables définies (ou modifiées) dans le script sont accessibles dans l'espace de travail.

I.4.4 Fonctions

- Utilisation : comme les fonctions prédéfinies dans Matlab. Première ligne d'une fonction :

```
function var_in = nom_fct(var_out)
```

Exemple :

```
function y = sinuscardinal(x)
  z = sin(x);     % Variable de stockage
  y = z./x;       % Variable de sortie
                  % Attention en x=0 !
```

Fonction appelée par :

```
>> sincpi = sinuscardinal(pi);
```

Autre exemple :

```
function [min, max] = minetmax(x)
    min = min(x); max = max(x);
```

Fonction appelée par :

```
>> [miny, maxy] = minetmax(y);
```

- Les variables sont transmises *par valeur* seulement.
- Les nombres d'arguments en entrée et en sortie ne sont pas fixes et peuvent être récupérés par *nargin* et *nargout*.
- Les variables de l'espace de travail ne sont pas accessibles à la fonction, sauf les variables d'entrée.

Réciproquement, les variables définies dans une fonction sont locales, sauf les variables de sortie.

- Outils de Mise au point : « déboggage »

Le plus simple : *keyboard*.

Plus évolués : *dbstop, dbstep, dbcont, dbclear, ...*, pour poser ou enlever un point d'arrêt, exécuter pas à pas ou poursuivre l'exécution.

I.5 Boucles et contrôles

I.5.1 Test « *if* » :

-
```
if (expression logique)
    suite d'instructions 1;
else
    suite d'instructions 2;
end
```

- Expressions logiques :

— Opérateurs logiques : et (&), ou (|), égal (==), différent (~=), supérieur (>, >=), inférieur (<, <=), non (~), ou exclusif (*xor*).

— Fonctions logiques : *exist, any, find, isinf, isnan, ...*

— Une expression est considérée comme fausse si nulle, et vraie sinon.

I.5.2 Boucle « *for* » :

-
```
for i=1:10
    suite d'instructions;
end
```

Eviter au maximum les boucles dans Matlab, e.g.,

```
for i=1:N, x(i) = i; end
```

est équivalent à

```
x = 1:N;
```

Autre exemple :

```
r = 1:10; A =[];  % init. de A à vide
for i=1:5, A = [A ; r]; end
```

peut être écrit

```
r = 1:10; A = ones(5,1)*r;
```

et même, plus subtil...

```
r = 1:10; A = r(ones(5,1),:);
```

- Utiliser pleinement les fonctions de Matlab, *e.g.*,

```
maximum = max(A);
```

Au lieu de :

```
n = length(A(1,:));
for i=1:n, maximum(i) = max(A(:,i)); end
```

I.6 Help

- Commande *help*, *help nomfolder*, *help nomfile*.

Créer son propre help :

— pour les fonctions et scripts,

— pour les répertoires : "*Contents.m*".

Commande *lookfor*.

- Avant d'écrire une fonction, il est recommandé de consulter la doc ou le *help* afin de savoir si une fonction similaire n'existe pas déjà.

De même, avant d'utiliser une fonction Matlab pour la première fois, consulter la doc ou le *help* pour vérifier qu'elle réalise bien l'opération souhaitée.

II. MATLAB POUR LE SIGNAL

II.1 Construction de matrices particulières

- Matrices de Toeplitz :

```
>> c = [ 1  2  3  4  5 ];
>> l = [ 1.5  2.5  3.5  4.5  5.5 ];
>> toeplitz(c,l)
ans =      1.0  2.5  3.5  4.5  5.5
           2.0  1.0  2.5  3.5  4.5
           3.0  2.0  1.0  2.5  3.5
           4.0  3.0  2.0  1.0  2.5
           5.0  4.0  3.0  2.0  1.0
```

Les colonnes remportent le conflit sur la diagonale.

- Matrices de Vandermonde :

```
>> x = 2:5;
>>vander(x)
ans =     8     4     2     1
         27     9     3     1
         64    16     4     1
        125    25     5     1
```

- Matrices de Hankel, Hadamard, Hilbert, ...

II.2 Décomposition de matrices

- Décomposition en valeurs singulières

```
>> [U,S,V] = svd(X,0);
```

S: matrice diagonale de même dimension que X, avec les éléments diagonaux non nuls ordre décroissant, et U, V matrices unitaires telles que X = U*S*V'.

- Factorisation de Cholesky

```
>> R = chol(X);
```

Utilise uniquement la partie triangulaire supérieure de X (et considère que la partie triangulaire inférieure de X est la transposée conjuguée de la précédente). Message d'erreur si X n'est pas définie positive, sinon *R* triangulaire supérieure telle que R'*R = X.

- Factorisation QR :

```
>> [Q,R] = qr(X);
```

Matrice *R* triangulaire supérieure de même dimension que *X*, matrice unitaire Q, telles que X = Q*R.

- Factorisation LU (Lower, Upper)

```
>> [L,U] = lu(X);
```

U matrice triangulaire supérieure, L permutation d'une matrice triangulaire inférieure, telles que X = L*U.

II.3 Fonctions de filtrage et de convolution

- Fonction *filter*, filtrage par un filtre ARMA :

```
>> Y = filter(B,A,X);
```

construit le vecteur Y tel que :

$y(n) = b(1)\,x(n) + b(2)\,x(n-1) + \ldots + b(nb+1)\,x(n-nb) - a(2)\,y(n-1) - \ldots - a(na+1)\,y(n-na)$

- Fonction *conv*, convolution de deux vecteurs :

```
>> Z = conv(X,Y);
```

interprétable comme le produit de deux polynômes.

- En 2D : *filter2* et *conv2*.
- La fonction *roots*

```
>> p = [1 -6 -72 -27]; r=roots(p)
```

calcule les racines d'un polynôme donné par ses coefficients (ici $P(z) = z^3 - 6z^2 - 72z - 27$).

La fonction *polyval* calcule la valeur d'un polynôme en certains points :

```
>> polyval(p,[ 1, exp(j*pi/4) ])
ans =    1.0e+002 *
         -1.0400 -0.7862-0.5620i
```

La fonction *poly*

```
>> p=poly(r)
p =       1.000 -6.000 -72.000 -27.000
```

inverse la fonction *roots*. Si l'argument de *poly* est une matrice, *poly* renvoie les coefficients du polynôme caractéristique de cette matrice.

II.4 Transformation de Fourier discrète

Transf. de Fourier rapide : *fft*, fft inverse : *ifft*.

```
» fx=fft(x);
```

fréquences sur [0, 1]. Pour [–0.5, 0.5] utiliser *fftshift*.

En dimension 2 : *fft2* et *ifft2*.

II.5 Fonctions mathématiques particulières

Fonctions de *bessel* de première et deuxième espèces, fonction *gamma*, ...

II.6 Autres instructions

rand : génération pseudo-alétoire uniforme sur [0, 1]

randn : génération gaussienne $\mathcal{N}(0,1)$

mean : moyenne empirique des comp. d'un vecteur

std : écart-type empirique des comp. d'un vecteur

xcorr : corrélation empirique d'un vecteur

cov : matrice de covariance empirique

II.7 Exemple de fichier exécutable Matlab

```
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
% Etude d'un ARMA
%~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% Définition du filtre ARMA
z = [0.2+0.8i 0.2-0.8i]; % 2 zéros
p = [0.7+0.3i 0.7-0.3i]; % 2 pôles
a = poly(p);  % coefficients pôles
b = poly(z);  % coefficients zéros

% Génération d'une séquence
% BBG pseudo-aléatoire
N = 100;
l = [-N/2+1:N/2]'/N;
x = 2*randn(N,1); % N échantillons

% Filtrage du BBG par l'ARMA
y = real(filter(b,a,x));

% TF et pério de la réali. de BBG
fx = fftshift(fft(x));
Px = 1/N*abs(fx).^2;

% TF et Pério de la réali. filtrée
fy = fftshift(fft(y));
Py = 1/N*abs(fy).^2;

% Affichage des résultats
subplot(2,2,1),plot(0:N-1,x)
title('Sequence d''entree')
axis([0 N-1 min(x) max(x)])

subplot(2,2,2),plot(0:N-1,y)
title('Sequence filtree')
axis([0 N-1 min(y) max(y)])

subplot(2,2,3), semilogy(l,Px)
title('Periodo de l''entree')
axis([-.5 .5 min(Px) max(Px)])

subplot(2,2,4), semilogy(l,Py)
title('Periodo de la filtree')
axis([-.5 .5 min(Py) max(Py)])
```