

Eviter au maximum les boucles, *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

```
>> r=1 : 10; A=r([1 1 1 1 1], :);
```

ou

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

6 Signal et image

6.1 Matrices particulières

Matrices de Toeplitz :

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

Matrices de Hankel, Hadamard, Hilbert, *etc.*...

6.2 Filtrage et convolution

Fonction `filter` (2D : `filter2`) : filtrage rationnel par un filtre de coefficients dans A et B :

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

Convolution `conv` (2D : `conv2`) :

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

La fonction `roots` : racines (zéros, pôles). Calcul des racines d'un polynôme donné par ses coefficients (ici $P(z) = z^3 - 6z^2 - 72z - 27$) :

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

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` inverse la fonction `roots`.

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

6.3 Transformation de Fourier

`fft`, `ifft`. Calcul de la FFT du signal `x`, sur 1024 points régulièrement distribués dans `[0, 1]` :

```
>> TF=fft(x,1024);
```

Pour `[-0.5, +0.5]` utiliser `fftshift`. En dimension 2 : `fft2` et `ifft2`.

6.4 Un peu d'aléa

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

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

`mean` : moyenne empirique

`std` : écart-type empirique

`xcorr` : corrélation empirique

`cov` : covariance empirique

6.5 Fonctions mathématiques spéciales

`besselj`, `bessely`, `gamma`, `erf`, `erfinv`, *etc.*...

6.6 Exemple de fichier de commande

Problème : On possède N mesures (x_n, y_n) bruitées censées correspondre à des points d'une droite. On va estimer la pente a et la valeur à l'origine b de cette droite par moindres carrés, c'est-à-dire minimisant :

$$J(a, b) = \sum_{k=1}^N (y_k - ax_k - b)^2$$

Solution : En annulant les dérivées partielles par rapport à a et b on trouve (un seul minimum) :

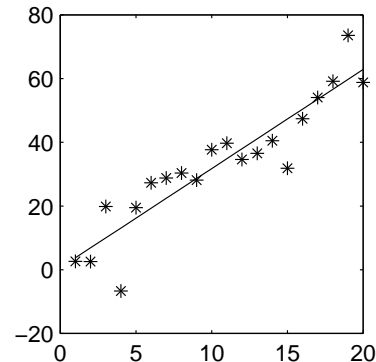
$$a = \frac{\sum_{k=1}^N x_k y_k - \frac{1}{N} (\sum_{k=1}^N x_k) (\sum_{k=1}^N y_k)}{\sum_{k=1}^N x_k^2 - \frac{1}{N} (\sum_{k=1}^N x_k)^2}$$

$$b = \frac{1}{N} \left(\sum_{k=1}^N y_k - a \sum_{k=1}^N x_k \right)$$

Script Matlab :

```
% Simulation
a = pi; b = 1/3;
% Tirage d'un ensemble de mesures
N = 20; x = 1 : N;
bruit = 10*randn(1,N); y = a*x+b+bruit;
% Estimation
aest = (sum(x.*y) - sum(x)*sum(y)/N)/...
      (sum(x.^2) - sum(x)^2/N);
best = (sum(y) - aest*sum(x))/N;
% Affichage
plot(x,y,'*',x,aest*x+best)
title(['Estimation a=',num2str(aest),...
      ' b=',num2str(best)])
```

Estimation a = 3.1107 b = 0.65773



— Matlab en bref —

Table des matières

1 Généralités	1
1.1 Introduction	1
1.2 Démarrer	1
1.3 Aide en ligne	1
1.4 Langage interprété	1
1.5 Variables	1
1.6 Variables complexes	1
1.7 Vecteurs, matrices et manipulations	1
1.8 L'opérateur d'énumération « : »	2
1.9 Chaînes de caractères	2
2 Opérations matricielles	2
3 Affichages	2
3.1 Affichage alpha-numérique	2
3.2 Graphiques 1D : <code>plot</code>	2
3.3 Graphiques 2D : <code>mesh</code>	3
3.4 Affichage de plusieurs courbes	3
4 Fichiers, scripts, fonctions	3
4.1 Répertoire de travail	3
4.2 Sauvegarde et chargement	3
4.3 Scripts	3
4.4 Fonctions	3
5 Boucles et contrôles	3
5.1 Test « if »	3
5.2 Boucle « for » :	3
6 Signal et image	4
6.1 Matrices particulières	4
6.2 Filtrage et convolution	4
6.3 Transformation de Fourier	4
6.4 Un peu d'aléa	4
6.5 Fonctions mathématiques spéciales	4
6.6 Exemple de fichier de commande	4

1 Généralités

1.1 Introduction

Matlab pour « Matrix Laboratory » : **tout est matrice**. Environnement de calcul matriciel, adapté à l'automatique et au traitement du signal et de l'image : facilité d'emploi, possibilités graphiques, boîtes à outils : signal processing, image processing, optimization, *etc.*...

1.2 Démarrer

Cliquez sur l'icône Matlab ou tapez `matlab` dans un shell. Tapez les commandes Matlab ou le nom d'un fichier de commandes à la suite des chevrons `>>`.

1.3 Aide en ligne

Commande `help` : `help NomFct`. Également `lookfor`. Avant d'écrire une fonction, assurez-vous qu'une fonction similaire n'existe pas déjà. De même, avant d'utiliser une fonction, vérifiez qu'elle réalise bien l'opération souhaitée.

Vérifiez également sa syntaxe. En cas d'erreur, lisez le message d'erreur que Matlab a la gentillesse de vous indiquer.

1.4 Langage interprété

Pas de compilation, ni de déclaration de variable, ni de réservation mémoire. Résultat affiché et stocké dans la variable `ans`.

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

Fonctions usuelles : `sin`, `cos`, `exp`, `log`, *etc.*...

1.5 Variables

Les noms de variable commencent par une lettre. Attention, Matlab fait la différence entre `X` et `x`.

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

Point virgule « ; » évite l'affichage du résultat. Plusieurs commandes par ligne : séparées par « ; » ou « , » :

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

1.6 Variables complexes

Matlab gère réels et complexes. `i` et `j` initialisés à $\sqrt{-1}$.

```
>> z = 3 + 2*i
z = 3.0000 + 2.0000i
```

Fonctions prédéfinies : `real`, `imag`, `abs`, `angle`, *etc.*...

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

1.7 Vecteurs, matrices et manipulations

Matrice sous forme d'un tableau :

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

Expressions à évaluer possible :

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

Éléments 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
```

Ajout de lignes à une matrice :

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

Ajout de colonnes : « , » à la place de « ; »

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, conjuguée, retournement, rotation :

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

1.8 L'opérateur d'énumération « : »

Pas à pas, incrément :

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

Incrémement par défaut : 1 (voir aussi linspace) :

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

Sélection d'éléments :

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

Si on a :

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

alors, extraction lignes 1 à 2 et toutes les colonnes :

```
>> A(1 : 2, :)
ans = 1 2 3
      4 5 6
```

1.9 Chaînes de caractères

Variables contenant des chaînes de caractères :

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

Manipulations de même type que pour les vecteurs :

```
>> mess = [mess ' v 5']
mess = Bienvenue sur Matlab v 5
```

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

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

Évaluation d'une chaîne : eval et feval :

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

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) % Val. et vect. propres
```

Matrices creuses : gain en mémoire et en coût de calcul

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

Pour certaines fonctions, matrice = tableau de valeurs :

```
>> exp(A); log(A); sqrt(A);
>> round(A); sign(A); rem(A,2);
```

Pour d'autres, matrice = suite de vecteurs colonnes

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

Décomposition de matrices :

```
>> [U,S,V] = svd(X,0); % Valeurs singulières
>> R = chol(X); % Cholesky
>> [Q,R] = qr(X); % QR
>> [L,U] = lu(X); % LU (Lower, Upper)
```

3 Affichages

3.1 Affichage alpha-numérique

Façon Matlab ou façon C

```
>> disp(mess)
pi vaut 3.142
>> fprintf('pi vaut %1.3e \n', pi)
pi vaut 3.141e+000
```

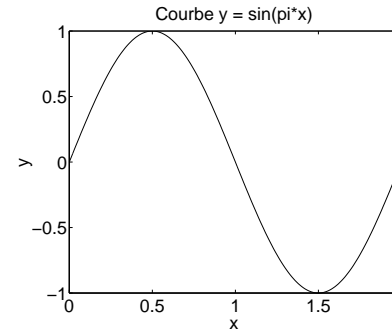
Saisie d'une valeur :

```
>> rep=input('Valeur de lambda : ');
```

3.2 Graphiques 1D : plot

Tracé d'une fonction avec axes et titre :

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

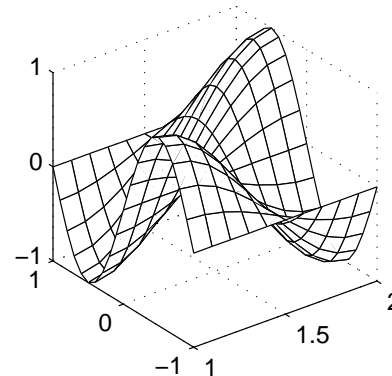


3.3 Graphiques 2D : mesh

Pour une fonction de deux variables :

```
>> 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, contour, image, surf, etc...



3.4 Affichage de plusieurs courbes

subplot divise la fenêtre graphique :

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



La commande figure crée une nouvelle fenêtre graphique. figure(2) permet d'adresser la figure n° 2.

4 Fichiers, scripts, fonctions

4.1 Répertoire de travail

Commandes cd et dir (idem DOS); cd et ls (idem Unix).

4.2 Sauvegarde et chargement

Sauvegarde :

```
>> save NomFichier NomsVariables
```

Exemple :

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

Par défaut sauvegarde dans matlab.mat.

Chargement :

```
>> load NomFichier
```

Voir aussi les commandes who, whos, pack et clear.

4.3 Scripts

Script : suite de commandes dans un fichier. Commentaires par « % ». Nom de fichier avec extension « .m » (nomfich.m). Exécution sous Matlab : nom du fichier, sans « .m ». Variables de l'espace de travail visibles dans le script et réciproquement.

4.4 Fonctions

Définition :

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

Appel identique aux fonctions Matlab :

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

Autre exemple :

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

Appel :

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

Passage par valeur seulement. Nombres d'arguments en entrée et en sortie accessibles dans nargin etargout. Variables de l'espace de travail invisibles dans les fonctions et réciproquement. Outils de mise au point (débuggage) : plus simple : keyboard; plus évolués : débogueur intégré...

5 Boucles et contrôles

5.1 Test « if » :

Structure générale :

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

Opérateurs logiques : et (&), ou (|, xor), égal (==), différent (~=), supérieur (>), inférieur (<), non (~). Fonctions logiques : exist, any, find, isinf, isnan, etc... Expression considérée fautive si 0, et vraie sinon.

5.2 Boucle « for » :

Structure générale :

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