Matlab

1 Présentation de Matlab

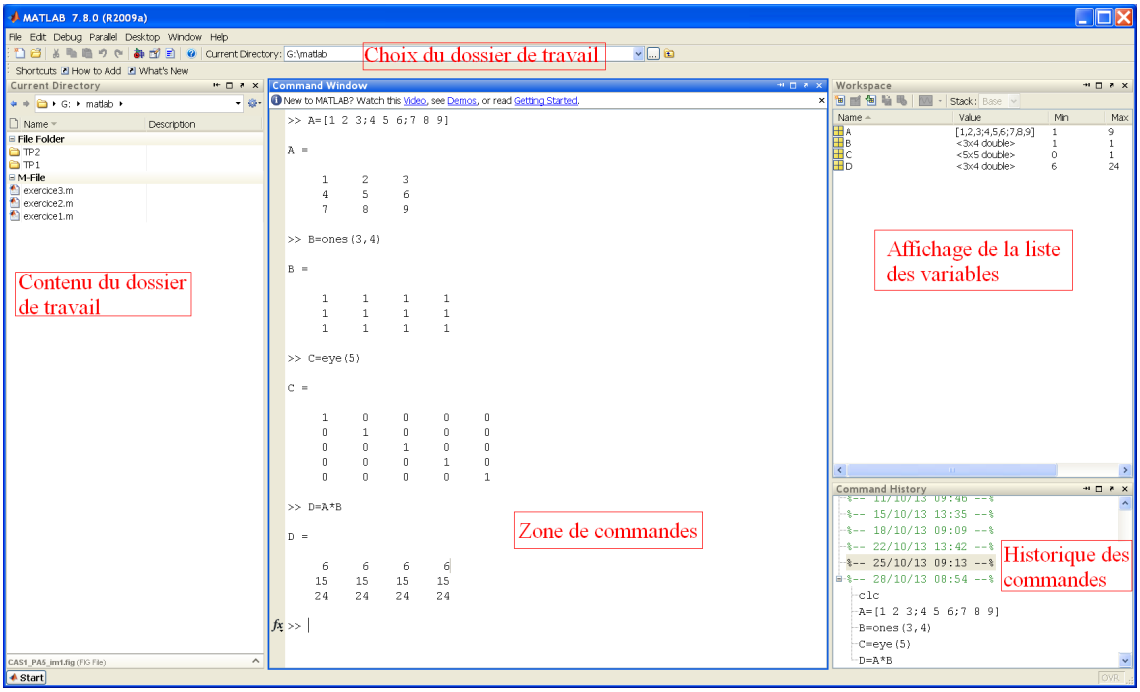
MATLAB, abréviation de MATrix LABoratory, est un langage très performant pour le calcul numérique. Les utilisations classiques sont : le calcul mathématique, le développement d'algorithmes, la modélisation et la simulation, l'analyse de données et la visualisation, le développement d'applications telles que les interfaces utilisateurs.

De nombreuses bibliothèques (*toolboxes*) complètent le noyau de base. Des produits concurrents et libres existent également, comme Scilab ou Octave.

Matlab est un interpréteur, il va exécuter les instructions saisies dans la fenêtre de commande (*command window*). On peut également regrouper ces instructions dans un fichier portant l'extension `.m` (par exemple : `ex1.m`). Ces instructions sont ensuite exécutées en tapant «`ex1`» dans la fenêtre de commande. L'interface de Matlab est composée de plusieurs parties :

— la fenêtre de commande (*command window*) où l'on tape les instructions ;
— le répertoire courant (*current directory*) qui affiche le contenu du dossier courant ;
— l'espace de travail (*workspace*) contenant la liste des variables existantes ;
— l'éditeur (*editor*) qui permet de créer des scripts ou des fonctions Matlab.

Ces différentes fenêtres peuvent être affichées ou non à l'aide du menu *Desktop*.



Quelques remarques générales avant de commencer :

— l'éxécution d'un programme peut être arrêtée à tout moment avec la combinaison de touche Ctrl+C ;

— pour éviter l'affichage de résultats lors de l'exécution d'instructions, il faut mettre un point-virgule à la fin de celles-ci.

2 Commandes d'aide

Diverses instructions seront très utiles par la suite :

— `lookfor mot_clé` qui permet de trouver une commande à l'aide de mots clés. Exemple : `lookfor filter`. Matlab donnera en réponse la liste des fonctions intégrées dont l'aide associée comporte ce mot-clé.

— `help nom_fonction` donne une aide sur la commande en question. Contrairement à `lookfor`, il est nécessaire de connaître le nom exact de la commande. Exemple : `help mean`.

— `help nom_toolbox` donne toutes les commandes de la toolbox par catégories. Exemple : `help signal; help stats;`

— la commande `demo` permet d'avoir accès aux démonstrations du logiciel.

3 Variables et Matrices

3.1 Mémoire

— Examen de la mémoire : `whos`

— Effacement de la mémoire : `clear` ou `clear all`

3.2 Variables

MATLAB n'a pas besoin de déclaration explicite de variables. Il réserve automatiquement de l'espace mémoire dès qu'il rencontre le nom d'une variable.

MATLAB fait la différence entre les minuscules et les majuscules.

Quand on omet de mentionner le nom d'une variable, MATLAB affecte la dernière valeur calculée à une variable temporaire générique 'ans' (pour answer).

La variable de base de MATLAB est une matrice, ce qui veut dire qu'un scalaire est considéré comme une matrice d'une ligne et d'une colonne.

3.3 Scalaire

Pour définir le réel $r = 2\pi$:

$r = 2 * pi$

On peut éviter l'affichage en faisant suivre la commande d'un point-virgule.

3.4 Vecteurs

3.4.1 Création d'un vecteur ligne

Pour obtenir le vecteur ligne $a = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$, il faut taper :

$a = [1\ 2\ 3]$ ou $a = [1, 2, 3]$
L'espace est un séparateur qui fait passer à la colonne suivante.

3.4.2 Création d'un vecteur colonne

Pour obtenir le vecteur colonne $a = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$, il faut taper :

$a = [1; 2; 3]$
Le point-virgule est un séparateur qui fait passer à la ligne suivante.

3.4.3 Vecteur transposé

$b = [5; 6; 7; 8]$
$c = b'$

3.4.4 Éléments d'un vecteur

$a = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10]$
$b = a(4) \Rightarrow 4^{\text{ème}}$ composante du vecteur
$b = a(4 : 9) \Rightarrow b =$ vecteur composé des éléments 4 à 9 du vecteur a

3.4.5 Opérateur d'énumération

Pour créer un vecteur dont les éléments constituent une suite arithmétique, on peut utiliser l'opérateur d'énumération :
`valeur initiale : pas : valeur finale`
le pas étant positif ou négatif. Ainsi, $e = 2 : 0.5 : 4$ correspond au vecteur $e = \begin{bmatrix} 2 & 2.5 & 3 & 3.5 & 4 \end{bmatrix}$

3.5 Matrices

3.5.1 Création d'une matrice

Dans MATLAB, toutes les variables représentent des matrices. On peut définir une matrice dans MATLAB de plusieurs façons :

— par la liste de ses éléments,
— en la générant par une suite d'instructions et de fonctions,
— en la lisant dans un fichier extérieur.

La construction élément par élément peut se faire de la façon suivante :

$$A = [1\ 2\ 3; 4\ 5\ 6; 7\ 8\ 9]$$

Cette instruction crée une matrice 3×3 et l'affecte à la variable A. Les éléments d'une ligne sont séparés par des espaces ou des virgules et les lignes sont séparées par des point-virgules. La matrice obtenue est la suivante : $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

Les éléments d'une matrice sont référencés de la façon suivante :

— $A(2,3)$ est l'élément de la deuxième ligne et troisième colonne de la matrice A.
— si on définit un vecteur B, alors $B(4)$ est le quatrième élément de ce vecteur.
— les indices des vecteurs débutent à 1 et non à 0.

On peut affecter une valeur scalaire à un élément d'une matrice comme suit :

$$A(2,3) = 8$$

qui va affecter la valeur 8 à l'élément de la deuxième ligne et troisième colonne de A.

Pour accéder à une colonne ou une ligne d'une matrice, il suffit d'utiliser l'opérateur :. Par exemple, $A(:,3)$ permet d'avoir tous les élements de la troisième colonne. $A(:,1:2)$ permet d'avoir les éléments des colonnes 1 à 2. $A([1\ 3],:)$ permet d'avoir les éléments des lignes 1 et 3. $A(end,:)$ permet d'avoir les éléments de la dernière ligne.

Certaines fonctions permettent de créer des matrices. Les fonctions matricielles les plus fréquemment utilisées sont :

`M=eye(n)`	renvoie la matrice identité, habituellement notée I_n en mathématiques
`M=ones(n,m)`	renvoie une matrice à n lignes et m colonnes dont tous les coefficients sont des 1
`M=zeros(n,m)`	matrice à n lignes et m colonnes dont tous les coefficients sont des 0 ; elle sert beaucoup pour les initialisations
`M=linspace(a,b,n)`	crée une matrice ligne de n points régulièrement espacés sur l'intervalle $[a,b]$ (bornes comprises)
`M=rand(n,m)`	crée une matrice aléatoire à n lignes et m colonnes dont les éléments sont uniformément distribués entre 0 et 1
`M=[a:b;c:d]`	crée une matrice de 2 lignes dont les éléments de la première ligne sont les entiers de a à b et les éléments de la seconde ligne sont les entiers de c à d
`M=[a:x:b;c:y:d]`	crée une matrice de 2 lignes dont les éléments de la première ligne sont les nombres de a à b, 2 nombres consécutifs étant séparés d'un pas de x, et les éléments de la seconde ligne sont les nombres de c à d, 2 nombres consécutifs étant séparés d'un pas de y.

Exemple : M=[3:7; 1.5:0.5:3.5] donne la matrice $M = \begin{bmatrix} 3 & 4 & 5 & 6 & 7 \\ 1.5 & 2 & 2.5 & 3 & 3.5 \end{bmatrix}$

3.5.2 Création d'une matrice complexe

$a = [i \quad 2 + 3 * j; 4 * i \quad 5]$ crée la matrice $a = \begin{bmatrix} i & 2+3j \\ 4i & 5 \end{bmatrix}$

i et j sont des variables prédéfinies. Si on écrit $i = 4$ et $j = 5$, i et j ne sont plus considérés comme des complexes. Il faut donc éviter d'utiliser i et j comme variables.

3.5.3 Concaténation de matrices

Des matrices peuvent être composées à partir d'autres matrices par concaténation.
$M = [10\ 12; 14\ 16]$;
$N = [M, ones(2,3); eye(3), zeros(3,2)]$

On obtient la matrice N suivante :

$$N = \begin{bmatrix} 10 & 12 & 1 & 1 & 1 \\ 14 & 16 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

3.5.4 Opérations usuelles

MATLAB permet d'effectuer les opérations usuelles en étendant leur définition aux matrices selon les règles de l'algèbre linéaire. Pour l'addition et la soustraction, les tailles des matrices doivent être égales.

MATLAB autorise que les dimensions ne soient pas égales lorsqu'une des variables est un scalaire. `A + 1` (où A est une matrice de taille 3*3) fournit la même valeur que `A + ones(3)`.

La multiplication de deux variables sera possible si les matrices qu'elles représentent respectent les règles de concordance des dimensions. MATLAB autorise que les dimensions ne concordent pas lorsqu'une des variables représente un scalaire.

MATLAB autorise le produit et les divisions par un scalaire. L'élévation à une puissance donnée est définie par le symbole ^.

3.5.5 Opérations élément par élément

Les opérations $*$, ^, \, / peuvent aussi s'appliquer élément par élément sur une matrice ou un vecteur en les faisant précéder d'un point. Par exemple, [1 2 3 4].*[1 2 3 4] ou [1 2 3 4].^2 donnent [1 4 9 16].

3.5.6 Fonctions élémentaires

Les fonctions élémentaires disponibles sous Matlab s'appliquent aussi bien à des matrices qu'à des scalaires. Les plus courantes sont :

— `sin`, `cos`, `tan`, `asin`, `acos`, `atan`

— `exp`, `log`, `log10`

— `abs`, `real`, `imag`, `sqrt`, `sign`

D'autres fonctions renvoient un scalaire lorsqu'elles s'appliquent sur des vecteurs. Dans le cas où on les applique à des matrices; elles agissent généralement sur chacune des colonnes indépendamment, le résultat est alors un vecteur. Il s'agit des fonctions :

— `max`, `min`

— `sort`

— `sum`, `prod`, `median`, `mean`, `std`

— `any`, `all`

Par exemple, l'élément maximal d'une matrice A est donné par `max(max(A))` ou par `max(A(:))`, et non pas par `max(A)` qui renvoie un vecteur.

3.5.7 Fonctions matricielles

Les plus courantes sont:

`eig`	valeurs propres
`poly`	polynôme caractéristique
`det`	déterminant
`inv`	inverse
`expm`	matrice exponentielle
`sqrtm`	matrice racine carrée
`size`	taille d'une matrice

Taille d'une matrice :

— `[m,n] = size(M)` permet d'obtenir le nombre de lignes (m) et le nombre de colonnes (n) de la matrice M. Si on pense avoir affaire à une matrice en 3 dimensions, on écrit:

— `[m,n,p] = size(M)`, p reçoit alors le nombre de composantes (ou de couches) de la matrice.

4 Gestion de l'espace de travail

Les fonctions suivantes permettent d'avoir une vue globale de l'espace de travail :

`who`	liste les variables
`whos`	liste les variables, leur taille et leur type
`what`	liste les fichiers d'extension `.m` et `mat`

Les variables peuvent être supprimées à l'aide de la commande `clear nom_variable`. Attention à la commande `clear` qui utilisée seule efface toutes les variables de l'espace de travail.

Lorsqu'on quitte Matlab, toutes les variables sont perdues. On peut les enregistrer en utilisant la commande `save nom_fichier nom_variables_a_enregistrer`. Le format du fichier est alors en `*.mat`. Lorsqu'on relance Matlab, on peut les recharger avec `load nom_fichier`.

Certaines commandes DOS ou Unix classiques peuvent être utilisées dans Matlab :

`pwd`	indique le chemin du répertoire courant
`cd`	change de répertoire
`dir`	liste tous les éléments d'un répertoire
`path`	obtention ou initialisation du chemin

D'autres commandes DOS ou UNIX peuvent être exécutées si elles sont précédées de "!". Par exemple : `!del nom_fichier`

5 Boucles et tests

Les principales instructions de contrôle proposées par MATLAB sont `for`, `while` et `if` ; elles fonctionnent à peu près comme leurs équivalents dans les autres langages de programmation.

5.1 La boucle for

La boucle `for` doit respecter la syntaxe suivante :

```
for compteur = expression
   instructions
end
```

Généralement, expression est un vecteur de la forme `début : increment : fin` et compteur prend successivement toutes les valeurs de expression pour exécuter instructions.

Il faut essayer d'éviter les boucles quand c'est possible. Pour cela on peut utiliser des boucles implicites. Pour construire la matrice $H = (h_{i,j})$ avec $h_{i,j} = \frac{1}{i+j-1}$, on peut utiliser les instructions suivantes :

$n = 3$;
$H = zeros(3)$;
for $i = 1 : n$
    for j $= 1 : n$
        $H(i, j) = 1/(i + j - 1)$;
    end;
end;

Notez que l'on a initialisé la variable H. C'est recommandé lorsque c'est possible. Notez également que l'on a indenté les boucles : c'est recommandé pour une meilleure lisibilité ! On peut aussi construire H de la façon suivante :

$J = 1 : n$;
$J = repmat(J, n, 1)$;
$I = J'$;
$E = ones(n)$;
$H = E./(I + J - E)$;

5.2 La boucle while

L'instruction `while` respecte la syntaxe suivante :

```
while expression
   instructions
end
```

expression désigne le résultat d'une opération logique. Elle est construite en utilisant des opérateurs relationnels et logiques.

5.3 Test : instruction if

Cette instruction permet d'exécuter une suite d'instructions conditionnelles. Sa syntaxe est la suivante :

`if` *condition*
   instructions
`end`

Le traitement va être exécuté uniquement si la condition est vraie. Des branchements multiples sont aussi possibles :

`if` *condition1*
    instructions1
`elseif` *condition2*
    instructions2
`else`
    instructions
`end`

5.4 Relations

Les principaux opérateurs relationnels sont les suivants :

$<$	inférieur
$>$	supérieur
$<=$	inférieur ou égal
$>=$	supérieur ou égal
$==$	égal ("=" est une affectation)
$\tilde{}=$	différent

Ils peuvent être combinés avec les opérateurs logiques suivants : & et, | ou, ~ non

Les fonctions `any` et `all` permettent de réduire les relations entre matrices à des vecteurs ou des scalaires. Consulter l'aide à leur sujet. Voir aussi la fonction `find` qui renvoie l'indice des éléments obéissant à une condition. Par exemple, soit y un vecteur, alors $find(y > 2)$ va renvoyer le rang de tous les éléments de y qui sont strictement supérieurs à 2.

6 Fichiers .m

Matlab peut exécuter un ensemble d'instructions contenues dans un programme. Ces programmes ont une extension `.m`. Il y a 2 types de programmes `.m` : les scripts et les fonctions. Pour ouvrir l'éditeur, cliquez sur le bouton "New M-file" ou tapez la commande `edit`.
% permet d'insérer des commentaires.

6.1 Scripts

Un fichier script est composé d'un ensemble d'instructions Matlab. Si le fichier a le nom `prog.m`, alors la commande `prog` va l'exécuter. Les variables d'un programme script sont globales, et son exécution va changer leurs valeurs dans l'espace de travail. Il est préférable de commencer chaque programme par les deux instructions suivantes :

— `clear;` supprime toutes les variables du workspace, libère la mémoire
— `close all;` ferme les fenêtres ouvertes par l'exécution des précédents programmes.

6.2 Fonctions

Les variables d'une fonction sont locales, mais peuvent aussi être déclarées comme globales. La première ligne du script d'une fonction doit obligatoirement être de la forme :

`function [resultat1, resultat2, ...] = nom_fonction(argument1, argument2, ...)`

Pour pouvoir exécuter une fonction, il faut l'enregistrer dans un fichier avec l'extension .m. On peut ensuite appeler la fonction en tapant

— `variable_résultat = nom_fonction(paramètres)` si la fonction renvoie un seul résultat (qui peut être une matrice),

— ou `[liste_variables_résultats] = nom_fonction(paramètres)` si la fonction renvoie plusieurs résultats. Les noms des variables résultats doivent être entre crochets et séparés par des virgules.

6.3 Textes, messages d'erreurs

Les textes doivent être encadrés par des quotes : `s='texte'`. Ils peuvent être affichés sur la fenêtre de commande Matlab : `disp('bonjour')`. On peut également afficher le contenu d'une variable : `disp(nom_variable)`.
Les messages d'erreur peuvent être affichés à l'aide de la commande `error` qui arrête aussitôt l'exécution du programme : `error('erreur!')`.
On peut également saisir des paramètres lors de l'exécution : `iter = input('Nombre d iterations?')`.

6.4 Pause et break

Lors de l'exécution d'un programme, on peut suspendre son exécution en utilisant l'instruction `pause` et la relancer si l'utilisateur appuie sur une touche.

7 Graphiques

Matlab permet de réaliser des graphiques 2-D et 3-D. Soient deux vecteurs x et y de même taille. La commande `plot(x,y)` va tracer y en fonction de x. Si on souhaite tracer un deuxième graphique sur une figure différente, taper `figure` ou `figure(i+1)` si la figure courante porte le numéro i.

Pour tracer un deuxième graphique sur la même figure, taper `hold on`. L'effet de cette commande s'annule avec `hold off`.

L'écriture de texte sur les axes d'un graphe ou sur le graphe s'effectue à l'aide des commandes :

`title`	titre du graphique
`xlabel`	titre de l'axe des abscisses
`ylabel`	titre de l'axe des ordonnées
`gtext`	placement d'un texte avec la souris
`text`	positionnement d'un texte spécifié par des coordonnées

La commande `grid` place une grille sur un graphe. Les commandes précédentes doivent être saisies après la commande `plot`.

Par défaut les courbes sont tracées par des lignes continues, mais il y a d'autres choix : - -, :, -., ., +, *, o et x. De même, on peut préciser les couleurs avec : y, m, c, r, g, b, w et k. Par exemple, `plot(x,y,'r*')` va tracer en rouge et avec des * la courbe de y en fonction de x.

La commande `subplot` permet de partitionner un graphe en plusieurs graphes : `subplot(m,n,p)` divise la fenêtre de la figure en une matrice $m \times n$ de petits sous-graphes et sélectionne le $p^{ème}$ sous-graphe pour la figure courante.

8 Manipulation d'images

8.1 Chargement d'une image

```
I = imread('nom_image')
```

Lorsque cette commande est exécutée, la matrice associée à l'image nom_image est chargée dans la variable I.

Si l'image est une image indexée, il faudra aussi charger la palette associée. Sinon, I contiendra uniquement les codes couleur de chaque pixel de l'image et les composantes des couleurs associées aux codes seront perdues. On écrira donc :

```
[I,P] = imread('nom_image')
```

La palette (composantes des couleurs présentes dans l'image) sera alors stockée dans P et la matrice des codes couleurs dans I.

8.2 Affichage d'une image

`imshow(I)` permet d'afficher l'image associée à la matrice I. Dans le cas d'une image indexée, on écrira `imshow(I,P)`, I étant la matrice image et P la palette associée.

`imshow` affiche l'image contenue dans le fichier image ou la matrice correspondante. `imshow` appelle `imread` pour lire l'image depuis le fichier, mais les données de l'image ne seront pas stockées dans le workspace.

`image` affiche la matrice en argument comme une image (n'accepte pas de fichier image en paramètre). L'affichage repose sur les valeurs de la matrice qui sont :

— associées aux index de la carte des couleurs activée (colormap)

— ou directement interprétées comme valeurs RGB (true color)

`imagesc` affiche la matrice en argument comme une image. Contrairement à `image`, elle effectue une remise à l'échelle des valeurs de la matrice avant l'affichage, de manière à utiliser pleinement la carte des couleurs

8.3 Enregistrement d'une image

`imwrite(I,'nom_image', 'fmt')` où I est la matrice représentant l'image, nom_image le nom du fichier dans lequel l'image sera enregistrée et fmt le format d'enregistrement souhaité (BMP, GIF, JPEG, PNG, TIFF...).

S'il s'agit d'une image indexée, il faudra aussi enregistrer la palette P associée et on écrira `imwrite(I,P,'nom_image', 'fmt')`

8.4 Extraction des composantes rouge, verte et bleue d'une image en vraies couleurs

Une image en vraies couleurs est associée à un tableau de dimension 3. Pour chaque pixel (i,j), le dosage de rouge se trouve en I(i,j,1), le dosage de vert en I(i,j,2) et le dosage de bleu en I(i,j,3).
`R = I(:,:,1);` permet d'extraire la matrice des composantes rouges de l'image I.
`V = I(:,:,2);` permet d'extraire la matrice des composantes vertes de l'image I.
`B = I(:,:,3);` permet d'extraire la matrice des composantes bleues de l'image I.

8.5 Conversion d'une image couleurs en une image d'intensités

```
IG = rgb2gray(I)
```

IG est la matrice d'une image en niveaux de gris (donc une image à 1 seule composante) calculée à partir des trois images de composantes de I.
Si I est une image couleur indexée, l'instruction de conversion sera : `IG = ind2gray(I,P)`

8.6 Conversion image indexée $\Leftrightarrow$ image en vraies couleurs

`I2 = ind2rgb(I1,P)` convertit l'image indexée I1, associée à la palette P, en image en vraies couleurs, dont la matrice est I2.
`[Iind,P] = rgb2ind(I,x)` convertit l'image en vraies couleurs I en image indexée Iind, associée à la palette P. x est la taille maximale souhaitée pour la palette P (en général une puissance de 2).

9 Fonctions principales

Instructions particulières

% *commentaires*	commentaires
`clear all`	supprime toutes les variables
`close all`	ferme toutes les fenêtres graphiques
`clc`	efface la fenêtre de commande
`format`	format d'écriture des nombres

Opérations

$+, -, *, /, \hat{}$	addition, soustraction, multiplication, division, puissance.
$.*, ./, .\hat{}$	multiplication, division, puissance élément par élément

Variables spéciales et constantes

`pi`	π
`i` ou `j`	nombre complexe i
`Inf`	nombre infini
`NaN`	« not a number » : exprime une indétermination
`ans`	dernière réponse

Graphiques

`figure`	nouvelle fenêtre graphique
`plot`	trace une courbe en reliant les points
`hold on`	permet de tracer plusieurs courbes l'un sur l'autre
`subplot`	divise la fenêtre en plusieurs graphiques
`xlabel, ylabel`	légende de l'axe des abscisses, des ordonnées
`title`	titre du graphique

Manipulation de fichiers

`save, load`	enregistre ou charge les données d'un fichier
`fopen, fclose`	ouvre, fermer un fichier
`fprintf`	écrit des données formattées dans un fichier

Opérations sur les matrices

' (apostrophe)	transposée
`linspace`	génère un vecteur dont les éléments sont régulièrement espacés
`ones`	crée une matrice remplie de 1
`zeros`	crée une matrice remplie de 0
`size`	taille d'une matrice
`length`	longueur d'un vecteur
`det`	déterminant

Fonctions mathématiques

`exp`	exponentielle
`cos, sin, tan`	cosinus, sinus, tangente
`sinc`	sinus cardinal (défini par $sinc(x) = sin(\pi x)/\pi x$)
`log`	logarithme népérien (ln)
`log10`	logarithme en base 10 (log_{10})
`exp`	exponentielle
`sqrt`	racine carrée
`real, imag`	partie réelle, imaginaire
`abs`	valeur absolue, module
`angle`	argument
`conj`	conjugué

Temps

`tic, toc`	chronomètre.

Aide

`help`	aide simple (documentation en ligne)
`doc`	aide détaillée (documentation hypertexte)
`lookfor`	recherche dans les mots-clés de l'aide
`demo`	démonstrations.

Programmation

`function`	fonction Matlab
`if, else, elseif`	condition
`end`	fin d'un boucle ou d'une condition
`for`	boucle (sur un nombre déterminé de fois)
`while`	boucle (sur un nombre indéterminé de fois)
`break`	termine l'exécution d'une boucle
`error`	affiche un message d'erreur et arrête l'exécution.

Interaction avec l'utilisateur

`disp`	affiche une matrice ou un texte
`input`	pose une question à l'utilisateur
`pause`	pause dans l'exécution d'un programme

Conversion en chaîne de caractères

`num2str`	convertit un nombre en chaîne de caractères
`mat2str`	convertit une matrice en chaîne de caractères