Introduction à MATLAB

N. Kerioui
Centre Universitaire Abdelhafid Boussouf, Mila, Algérie,
Département de mathématiques et informatique,
$2^{ème}$ année licence MATH.

Table des matières

1- Introduction.....2

2- L'environnement MATLAB.....2

3- L'aide en MATLAB.....3

4- Première interaction avec MATLAB.....3

5- Les nombres en MATLAB.....4

6- Les chaines de caractères.....5

7- Les fonctions en MATLAB.....6

8- Les vecteurs.....7

9- Les matrices.....7

10- Éléments de programmation.....8

11- Polynômes.....11

12- Graphiques.....11

13- Calcul symbolique.....15

1. Introduction

Le nom MATLAB est la contraction du terme anglais (**MAT**rix **LAB**oratory). Ce logiciel est un environnement de programmation interactif pour le calcul scientifique et de la visualisation des données produit par Mathworks.

À l'origine MATLAB était conçu pour faire principalement des calcules sur les vecteurs et les matrices d'où son nom '**MAT**rix **LAB**oratory', mais par la suite il a était utilisé pour exécuter des fonctions, d'attribuer des valeurs à des variables, etc. Il permet aussi d'effectuer des opérations mathématiques, de manipuler des matrices, de faire des graphiques, etc.

Une caractéristique de MATLAB est que les variables n'ont pas à être déclaré, leur nature se déduisant automatiquement lors de l'affectation.

2. L'environnement MATLAB

La version 7 de MATLAB affiche au démarrage plusieurs fenêtres notamment :

i) ***Command Window*** : nous l'utilisons pour formuler nos expressions et interagir avec MATLAB, et c'est la fenêtre la plus utilisée.

ii) ***Workspace*** : indique toutes les variables existantes avec leurs types et valeurs.

iii) ***Current Folder*** : indique le répertoire courant ainsi que les fichiers existants.

iv) ***Command History*** : garde la trace de toutes les commandes entrées par l'utilisateur.

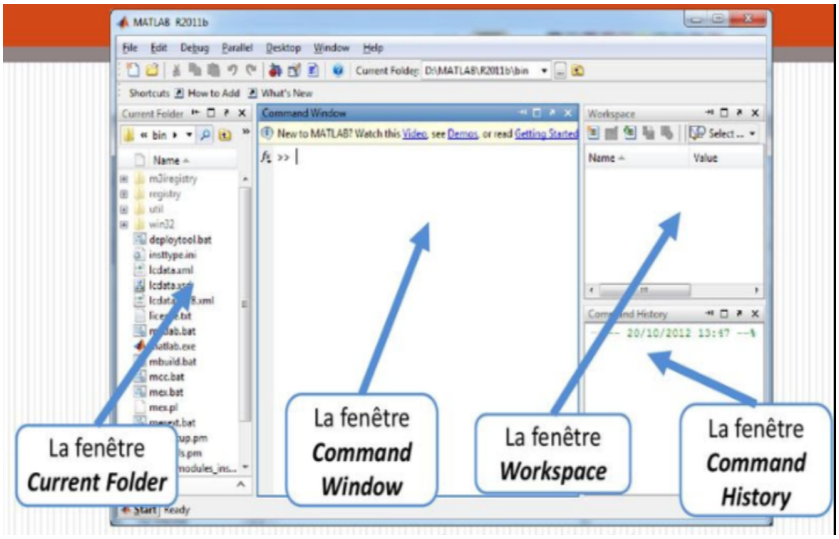

FIGURE 1 – Fenêtre typique de MATLAB.

3. L'aide en MATLAB

help : donne de l'aide sur une fonction ou un toolkit (help help)

helpdesk : documentation en hypertexte (requiert Netscape ou autre)

helpwin : aide en ligne dans une fenêtre séparée

lookfor : recherche d'un mot clé (lent)

which : localise fonctions et fichiers

what : liste des fichiers MATLAB dans le répertoire courant

exist : check si une fonction ou une variable existe dans le workspace

who, whos : liste des variables dans le workspace

4. Première interaction avec MATLAB

a- Le moyen le plus simple pour utiliser MATLAB est d'écrire directement dans la fenêtre de commandes (*Command Window*) juste après le curseur (*prompt*) >>

b- Toutes les variables sont considérées comme des matrices plutôt que de simples structure (caractère, réel,...etc). Par exemple, une variable scalaire est vue par MATLAB comme une matrice 1×1 (une ligne, une colonne).

c- Les opérations élémentaires dans MATLAB sont : $+$ l'addition, $-$ la soustraction, * la multiplication, / la division, \ la division inverse, ^ la puissance et $'$ le transposé.

d- La variable **ans** contient la valeur du dernier calcul si celui-ci n'a pas été attribué à une autre variable :

$$>> \quad 5+6 \qquad \rightarrow \qquad \text{ans} = 11$$

e- Pour créer une variable on utilise la structure simple : >> nomvariable = valeur

sans se préoccuper du type de la variable. Par exemple

$$\begin{aligned} &>> a = \quad 5; \\ &>> b = \quad 'bonjour'; \\ &>> c = \quad false; \end{aligned}$$

Remarque 1 : Le nom d'une variable ne doit contenir que des caractères alphanumériques ou le symbole (underscore) (les symboles d'opération sont interdits), et doit commencer par un alphabet, nous devons aussi faire attention aux majuscules car le MATLAB est sensible à la casse (**A** et **a** sont deux variables différentes.)

Remarque 2 : L'interpréteur de MATLAB traite les instructions d'affectation selon deux cas :

– Si c'est une nouvelle variable, il lui crée son entrée et lui réserve automatiquement un nouvel espace de stockage approprié (par défaut type double).

– Si la variable existe déjà, il modifie sa valeur et lui ajoute/réduit l'espace de stockage, si nécessaire.

f- Les variables sont déclarées lorsqu'elles sont initialisées :

$x = 8$	variable scalaire (1×1)
$y = 1.26485978$	variable scalaire (1×1)
$vectl = [-1, 0, 3, 1.2, 8]$	vecteur ligne (1×5)
$vectc = [0; 8; 24; -2]$	vecteur colonne (4×1)
$z =$ 'Mila university center'	chaine de caractère (1×20)

g- Le point décimal s'exprimer par . et on écrit par exemple $>> \quad A = 4.5$

h- Le point virgule ; est utilisé pour terminer une linge et pour éviter l'affichage, par exemple $>> \quad x = 2; \qquad \rightarrow \qquad >>$. Pour afficher une valeur de manière propre, on préférera la commande **disp()**.

Remarque 3 : il est possible d'écrire plusieurs expressions dans la même ligne en les faisant séparées par des virgules ou des points virgules. Par exemple :

$$>> \quad 5+6, 2*5-1, 12-4 \quad \text{ou} \quad >> \quad 5+6; 2*5-1, 12-4;$$

i- La pourcentage % est utilisé pour faire un commentaire.

j- Les parenthèses ( ) sont utilisés pour marquer la priorité d'une expression arithmétique.

k- Les accolades [ ] sont utilisés pour former les vecteurs lignes, les vecteurs colonnes et les matrices :

$$>> \quad vl = [1 \;\; 6 \;\; 2.9] \quad \rightarrow \quad vl = 1 \;\; 6 \;\; 2.9$$

$$>> \quad vc = [1; \;\; 6; \;\; 2.9] \quad \rightarrow \quad vc = \begin{matrix} 1 \\ 6 \\ 2.9 \end{matrix}$$

$$>> \quad mat = [1 \;\; 2; 3 \;\; 4] \quad \rightarrow \quad mat = \begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}$$

l- Les opérateurs booléennes en MATLAB sont : & (et logique), | (ou logique), $\sim$ (non logique).

m- Les opérateurs relationnels en MATLAB sont : $<$ (inférieur), $<=$ (inférieur ou égale), $>$ (supérieur), $>=$ (supérieur ou égale), $==$ (égale), $\sim=$ (différent).

5. Les nombres en MATLAB

i- MATLAB utilise toujours les nombres réels (double précision) pour faire les calcules, ce qui permet d'obtenir une précision de calcule allant de jusqu'aux 16 chiffres significatifs. Généralement, le résultat d'une opération de calcule est par défaut affiché avec quatre chiffres après la virgule. Pour de grandes variables, il est donc conseillé de changer le type. Les types existants sont mentionnés dans la Figure 2.

ii- MATLAB utilise les signes + et − pour les nombres signés et utilise la lettre 'e' pour spécifier la facteur d'échelle en puissance de 10, par exemple :

$$1.60210e-20 \quad \Leftrightarrow \quad 1.60210 \times 10^{-20}$$

iii- Les nombres complexes utilise les caractères 'i' et 'j' pour designer la partie imaginaire :

$$>> \ c1 = 1 - 2i \ \rightarrow \ 1.0000 - 2.0000i$$

iv- MATLAB, en tant que langage scientifique, a prévu des constantes prédéfinies : **pi, inf, i ou j, realmin, realmax, eps, ans, flops.**

$$>> \ eps \ \rightarrow \ ans = 2.2204e-16 \qquad >> \ realmax \ \rightarrow \ ans = 1.7977e+308$$

v- Pour changer l'affichage d'un nombre, on utilise la commande format tels que :

format long : affiche les nombres avec 04 chiffres après la virgule

format short : affiche les nombres avec 14 chiffres après la virgule

format bank : affiche les nombres avec 02 chiffres après la virgule

format rat : affiche les nombres sous forme d'une ration a/b

Types de données	Taille (bytes)	Gammes de valeurs	Fonctions de conversion
Signed 8-bit integer	1	-2^7 à 2^7-1	`int8()`
Signed 16-bit integer	2	-2^{15} à $2^{15}-1$	`int16()`
Signed 32-bit integer	4	-2^{31} à $2^{31}-1$	`int32()`
Signed 64-bit integer	8	-2^{63} à $2^{63}-1$	`int64()`
Unsigned 8-bit integer	1	0 à 2^8-1	`uint8()`
Unsigned 16-bit integer	2	0 à $2^{16}-1$	`uint16()`
Unsigned 32-bit integer	4	0 à $2^{32}-1$	`uint32()`
Unsigned 64-bit integer	8	0 à $2^{64}-1$	`uint64()`
Double-Precision Floating Point (64 bits)	8	$-1.79769e^{+308}$ à $-2.22507e^{-308}$ et $2.22507e^{-308}$ à $1.79769e^{+308}$	`double()`
Single-Precision Floating Point (32 bits)	4	$-3.40282e^{+038}$ à $-1.17549e^{-038}$ et $1.17549e^{-038}$ à $3.40282e^{+038}$	`single()`
Char	2	0 à 65535	`char()`
Logical (8 bit)	1	0 à 1	`logical()`

FIGURE 2 – Types de données.

6. Les chaines de caractères

- Il est possible de concaténer les mots (i.e. de construire les phrases) avec deux méthodes : soit en utilisant l'espace entre les mots ou bien à l'aide des parenthèses carrées [ ]. Par exemple :

>> phrase1= 'Mila university center' → phrase1= Mila university center

>> mot1= 'Mila' ; mot2= 'university' ; mot3= 'center' ; phrase2=[mot1, ' ', mot2, ' ', mot3]

- Il n'est pas possible de mélanger directement les variables numériques et les variables textes. Pour afficher une valeur numérique dans un texte, il est nécessaire de la transformer en texte (string) à l'aide de la fonction **num2str()** :

```
>> a=50 ; b=a/2 ; moitié=['La moitié de', num2str(a) , 'est', num2str(b)]
```

7. Les fonctions en MATLAB

Parmi les fonctions les plus fréquemment utilisées, on peut citer :

- $sin(x)/cos(x)$ le sinus et le cosinus de x (en radian)
- $tan(x)/atan(x)$ le tangent et l'arc tangent de x (en radian)
- $asin(x)/acos(x)$ l'arc sinus et l'arc cosinus de x (en radian)
- $sqrt(x)$ la racine carrée de x
- $abs(x)$ la valeur absolue de x
- $exp(x) = e^x$
- $log(x)$ logarithme naturel de x
- $log10(x)$ logarithme à base 10 de x
- $imag(x)$ la partie imaginaire du nombre complexe x
- $real(x)$ la partie réelle du nombre complexe x
- $round(x)$ arrondi un nombre vers l'entier le plus proche
- $ceil(x)$ arrondi un nombre vers l'entier suivant
- $factor(x)$ les diviseurs de x sauf 1 et lui même
- $isprime(x)$ Tester si le nombre x est premier ou non (le résultat est de type logique)
- $gcd(x, y)$ le PGCD de x et y
- $lcm(x, y)$ le PPCM de x et y

8. Les vecteurs

Par défaut, le vecteur est une ligne à plusieurs colonnes ou est une colonne à plusieurs lignes. Les composantes sont séparées par des blancs ou des virgules pour les vecteurs lignes et par des points-virgules pour les vecteurs colonnes. Des messages erreurs sont affichés si une opération impossible est tentée (par exemple l'addition de vecteurs de longueurs différentes).

Opérations vectorielles. Le tableau suivant résument certaines commandes couramment utilisées :

n : m	nombres de n à m par pas de 1
n : p : m	nombres de n à m par pas de p
linspace(n,m,p)	p nombres de n à m
lenght(A)	longueur de A
A(i)	i-ème coordonnée de A
A(i1 : i2)	coordonnées i1 à i2 de A
A(i1 : i2)=[ ]	supprimer les coordonnées i1 à i2 de A
[A,B]	concaténer les vecteurs A et B
A*B'	produit scalaire des vecteurs lignes A et B
A'*B	produit scalaire des vecteurs colonnes A et B
reshape(A,u,v)	crée une matrice de taille [u,v], à partir de A

• Généralement, ( ) représente l'ensemble d'arguments tandis que [ ] représente l'énumération d'éléments.

9. Les matrices

Une matrice est une ensemble de lignes comportant toutes le même nombre de colonnes. Les matrices suivent la même syntaxe que les vecteurs. Les composantes des lignes sont séparées par des virgules et chaque ligne est séparée de l'autre par un point virgule.

size(A)	nombre de lignes et de colonnes de A
A(i,j)	coéfficient d'ordre i,j de A
A(i1 : i2, :)	lignes i1 à i2 de A
A(i1 : i2, :) =[ ]	supprimer les lignes i1 à i2 de A
A(:, j1 : j2)	colonnes j1 à j2 de A
A(:, j1 : j2)=[ ]	supprimer les colonnes j1 à j2 de A
A(:)	concaténation des vecteurs colonnes de A
A(i)	coéfficient d'ordre i dans l'indexation linéaire
diag(A)	coéfficients diagonaux de A

Les opérations matricielles

A'	transposée de A
inv(A)	inverse de A
expm(A)	exponentielle de A
det(A)	déterminant de A
trace(A)	trace de A
poly(A)	polynome caractéristique de A
eig(A)	valeurs propres de A
[U,D]=eig(A)	vecteurs propres et valeurs propres de A
A B	solution de Ax = B
B/A	solution de xA = B

Quelques matrices particulières

zeros(m,n)	matrice nulle de taille m,n
ones(m,n)	matrice de taille m,n dont tous les coéfficients valent 1
eye(n)	matrice identité de taille n
diag(A)	matrice diagonale dont la diagonale est le vecteur A
magic(n)	carré magique de taille n
rand(m,n)	matrice de taille m,n de nombres aléatoires entre 0 et 1
rank(A)	nombre de colonnes ou lignes linéairement indépendantes
flipud(A)	basculer les lignes de haut en bas
fliplr(A)	basculer les colonnes de gauche à droit
rot90(A)	rotations de 90 degres (sens trigo)
triu(A)	extrait le triangle supérieur de A
tril(A)	extrait le triangle inférieur de A

10. Éléments de programmation

a. Instructions de contrôle

a.1. L'instruction For

- La syntaxe est :
  for valeur initiale : pas : valeur finale
  bloc d'instruction
  end

a.2. L'instruction while

- La syntaxe est :
  while expression

les commandes
end

a.3. L'instruction if

- La syntaxe est :

if l'expression 1
commandes à exécuter si l'expression 1 est "vrai"
elseif l'expression 2
commandes à exécuter si l'expression 2 est "vrai"
else
commandes à exécuter si aucune expression est "vrai"
end

a.4. L'instruction switch...case

- La syntaxe est :

switch expression logique
case valeur1
bloc 1
case valeur2
bloc 2
... ...
case valeurn
bloc n
otherwise
bloc n+1
end

b. Scripts et fonctions

MATLAB permet de créer des fichiers de code utilisateurs dits **fichier.m**. Ils contiennent des suites d'instructions MATLAB qui sont exécutées les unes après les autres. Pour exécuter ces fichiers, ils doivent être stockés sous le répertoire courant ou inclus dans des répertoires identifiés sur le chemin de recherche.
Ces **fichiers.m** sont classés en deux types : **les scripts** et **les fonctions**. Ils sont employés comme toute autre fonction ou commande de MATLAB.

b.1. Scripts

Les scripts sont des fichiers de code qui n'utilisent aucune argument d'appel en entrée et ne retournent aucun argument en sortie. Ces fichiers manipulent les variables de l'espace de travail, en crée de nouvelles et en supprime d'autres de manières permanente. Voici un exemple de script :

```
beautemps = 1;
if beautemps ~= 0
disp('Hello, il fait beau')
end
return
```

- Le script modifie les valeurs des variables du workspace.
- Pour créer un script : File → New → M-File, ou d'une manière équivalente :

```
>> edit
```

- L'utilisation des scripts est très simple : il suffit de saisir leur nom devant le prompt pour que MATLAB exécute leurs instructions internes.

b.2. Fonctions

Les fonctions, contrairement aux scripts, acceptent des arguments en entrée et permettent le retour d'arguments en sortie. Ils manipulent leurs propres variables indépendamment de celles présentent sur l'espace de travail. La syntaxe de déclaration des fonctions est comme suit :

```
function [y1, ..., ym] = nomfonction (x1, ..., xn)

{ Corps de la fonction

end
```

- On n'exécute jamais un programme de type fonction. On l'appelle depuis un autre programme ou depuis la console. Voici un exemple de fonction :

```
function y = temps(x)
if length(x) > 1
error('x doit être un scalaire ')
end
if x ~= 1
disp('Hello, il fait beau')
else
disp('espérons que demain sera meilleur !')
end
return
```

- les variables internes d'une fonction sont locales et n'entrent pas dans le workspace. Vous pouvez, dans une fonction, déclarer une ou plusieurs variables globales afin de pouvoir les utiliser depuis l'extérieur de la fonction. Par exemple :

```
global x y
```

- MATLAB offre plusieurs moyens de vérifier les arguments d'entrées et de sorties d'une fonction :

nargin	retourne le nombre d'arguments d'entrée
nargout	retourne le nombre d'arguments de sortie
nargchk	vérifie le nombre d'arguments d'entrée
error	affiche un message d'erreur
inputname	retourne le nom d'un argument d'entrée

11. Polynômes

MATLAB représente un polynôme sous forme d'un vecteur ligne contenant les coefficients classes dans l'ordre des puissances décroissances. Par exemple le polynôme P d'expression $P(x) = x^2 - 6x + 9$ est représenté par : >> $P = [1\ -6\ 9]$. Le tableau suivant représente quelque commandes MATLAB pour manipuler des polynômes :

y=polyval(p,x)	y = valeurs de P(x)
z=roots(p)	z = racines de P (i.e. telles que P(z) = 0)
p=conv(p1,p2)	p = coefficients du polynôme P1P2
[q,r]=deconv(p1,p2)	q = coefficients de Q, r = coefficients de R tels que P1 = QP2 + R
y=polyder(p)	y = coefficients de P'(x)
y=polyint(p)	y = coefficients de $\int_0^x P(t)dt$

12. Graphiques

a. Les figures : figure

Les graphiques se mettent dans des fenêtres appelées "figures". Lorsque plusieurs graphiques sont tracés, ils peuvent être mis dans des fenêtres différentes, les figures sont alors numérotées.

b. Le tracé de courbes dans le plan : plot

La fonction plot permet de placer des points de coordonnées (x, y) sur un graphique. Nous illustrons diverses possibilités à travers quelques exemples :

Exemple 1 :

x=[1 2 3 4] ; % définit le vecteur ligne de coordonnées $(1,2,3,4)$ et le nomme x

y=[5 1 2 1] ; % définit le vecteur ligne de coordonnées $(5,1,2,1)$ et le nomme y

plot(x,y) % trace y en fonction de x et joint les points (x,y) par des segments de droites.

Exemple 2 :

x=[1 2 3 4] ; % définit le vecteur ligne de coordonnées $(1,2,3,4)$ et le nomme x

y=[5 1 2 1] ; % définit le vecteur ligne de coordonnées $(5,1,2,1)$ et le nomme y

plot(x,y,'+') % Positionne les points de coordonnées (x,y) définis ci-dessus au moyen du symbole '+'. D'autres symboles peuvent être utilisés (+,.,o).

x=-5 :0.1 :5 ; % définit le vecteur ligne dont la première coordonnée est -5, la dernière est 5 et la différence entre deux coordonnées successives est 0.1

y=sin(x) ; % définit le vecteur y dont les coordonnées sont les sinus des coordonnées de x.

z=cos(x) ; % définit le vecteur z dont les coordonnées sont les cosinus des coordonnées de x.

plot(x,y,'r',x,z,'b') % trace le graphe de la fonction sinus entre -5 et 5, en rouge et sur le même graphique, trace le graphe de la fonction cosinus entre -5 et 5 en bleu.

c. Plusieurs graphiques sur la même figure : subplot

Il est possible de tracer plusieurs graphiques sur la même figure au moyen de l'instruction subplot (sous-graphique). L'exemple suivant illustre le fonctionnement de cette instruction.

Exemple :

```
x=-5 :0.1 :5 ; % définit le vecteur ligne dont la première coordonnée est -5, la dernière est 5 et la différence entre deux coordonnées successives est 0.1
y=sin(x) ; % définit le vecteur y dont les coordonnées sont les sinus des coordonnées de x.
z=cos(x) ; % définit le vecteur z dont les coordonnées sont les cosinus des coordonnées de x.
subplot(2,1,1) % définit deux sous-graphiques (2 lignes de graphiques et 1 colonne) et sélectionne le 1er des deux sous-graphiques.
plot(x,y,'r') % trace le graphe de la fonction sinus entre -5 et 5, en rouge.
subplot(2,1,2) % définit deux sous-graphiques (2 lignes de graphiques et 1 colonne) et sélectionne le 2ème des deux sous-graphiques.
plot(x,z,'g') % trace le graphe de la fonction cosinus entre -5 et 5 en vert.
```

d. Le tracé de courbes en dimension 3 : plot3

Pour tracer une courbe paramétrée avec 3 coordonnées, on utilisera l'instruction **plot3**. L'instruction **grid on** permettra de placer un quadrillage utile pour se repérer en dimension 3. L'instruction **grid off** permet de supprimer le quadrillage.

Exemple :

```
u=-10 :0.1 :10 ; % définit le vecteur ligne dont la première coordonnée est -10, la dernière est 10 et la différence entre deux coordonnées successives est 0.1
x=sin(u) ; % définit le vecteur x dont les coordonnées sont les sinus des coordonnées de u.
y=cos(u) ; % définit le vecteur y dont les coordonnées sont les cosinus des coordonnées de u.
z=u ; % définit le vecteur z dont les coordonnées sont celles de u.
plot3(x,y,z) ; % trace la courbe en trois dimensions.
```

e. Les surfaces : surf et surfc

- Une fonction de deux variables peut être représentée par un graphe : ce graphe est

une surface, à chaque couple (x, y) est associé un nombre z. Attention, la dimension de z est liée à celles de x et de y. Le nombre de lignes de x et de z doivent être identiques, le nombre de lignes de y doit être égal au nombre de colonnes de z.

Exemple :

```
x=linspace(-5,5,100) ;
y=linspace(-5,5,100) ;
z=cos(x)'*sin(y) ;
surf(x,y,z) ;
```

- L'instruction **surfc** a les mêmes fonctionnalités que la fonction **surf** décrite précédemment mais elle trace également les courbes de niveau, le lecteur pourra regarder ce que fait la liste d'instructions précédente dans laquelle on remplacera l'instruction **surf** par **surfc**.

f. Titres de figures et étiquettes sur les axes

- L'instruction title ('écrire le titre de la figure ici') permet de mettre un titre sur une figure.
- L'instruction xlabel ('écrire l'étiquette de l'axe des abscisses ici') permet de mettre une étiquette sur l'axe des abscisses.
- L'instruction ylabel ('écrire l'étiquette de l'axe des abscisses ici') permet de mettre une étiquette sur l'axe des ordonnées.
- L'instruction zlabel ('écrire l'étiquette de l'axe des cotes ici') permet de mettre une étiquette sur l'axe des cotes.

Remarque : Si on souhaite ajouter une courbe à un graphique déjà existant par exemple, il faut utiliser l'instruction **hold on**. Pour désactiver le graphique, on utilise l'instruction **hold off**. Dans l'exemple suivant, on trace d'abord le graphe de la fonction *sin* entre -5 et 5, puis on trace sur le même graphique le graphe de la fonction *cos*.

Exemple :

```
x=-5 :0.1 :5 ;
plot(x,sin(x)) ;
hold on
plot(x,cos(x))
hold off
```

13. Calcul symbolique

La toolbox symbolic de MATLAB contient les commandes **diff**, **int** et **taylor** qui fournissent respectivement l'expression analytique de la dérivée, de l'intégrale indéfinie (i.e. une primitive) et le polynôme de Taylor d'une fonction donnée. En particulier, si on a défini une fonction avec la chaîne de caractères f, **diff**(f,n) donne sa dérivée à l'ordre n, **int**(f) son intégrale indéfinie, et **taylor**(f,x,$n+1$) son polynôme de Taylor de degré n en $x_0 = 0$. La variable x doit être déclarée comme symbolique en utilisant la commande **syms x**. Cela permettra de la manipuler algébriquement sans avoir à spécifier sa valeur. Pour appliquer ceci à la fonction

$$f(x) = \frac{x^2 + 2x + 2}{x^2 - 1}$$

on procède ainsi :

```
>> f = '(x^2+2*x+2)/(x^2-1)';
>> syms x
>> diff(f)
  (2*x+2)/(x^2-1)-2*(x^2+2*x+2)/(x^2-1)^2*x
>> int(f)
  x+5/2*log(x-1)-1/2*log(1+x)
>> taylor(f,x,6)
  -2-2*x-3*x^2-2*x^3-3*x^4-2*x^5
```

Notons que la commande **simple** permet de réduire les expressions générées par **diff**, **int** et **taylor** afin de les rendre aussi simples que possible. La commande **funtool** aide à la manipulation symbolique de fonctions à l'aide d'une interface graphique.