Maîtrise Chimie Année 2005

Initiation à MATLAB

Kimathy ELISE

Sommaire

Chapitre 1

Généralités et prise en main

1.1 Démarrage, quitter

Pour lancer le programme, tapez matlab dans une fenêtre de commandes. Une fenêtre logo fait une breve apparition, puis dans la fenêtre de commande, le symbole apparaît : C'est l'invite de MATLAB qui attend vos commandes.
Vous pourrez quitter la session avec la commande quit.

1.2 Aide

L'aide en ligne peut être obtenue directement dans la session en tapant help nom de commande. La commande help toute seule vous indiquera les différents thèmes abordes dans la documentation. En tapant help directement, MATLAB liste l'ensemble des boites à outils (ToolBox) disponibles et donc l'ensemble des fonctions que l'on peut utiliser, c'est peut être la le plus gros atout de MATLAB.

1.3 Calculs élémentaires

Commençons par les opérateurs les plus courants : +, -, *, /, ^. Le dernier signifie «puissance», et on retiendra qu'il est différent de celui du FORTRAN. Les parenthèses s'utilisent de manière classique.
Nous avons tout pour effectuer un premier calcul : tapez une expression mathématique quelconque et appuyez sur «Entrée». Par exemple :

```
>> (3*2)/ (5+3)
ans =
0.7500
```

Le résultat est mis automatiquement dans une variable appelée ans (answer). Celle-ci peut être utilisée pour le calcul suivant, par exemple :

```
>> ans*2
ans =
1.5000
```

Ensuite, vous remarquerez que le résultat est affiche avec 5 chiffres significatifs, ce qui ne signifie pas que les calculs sont faits avec aussi peu de précision. La précision utilisée par MATLAB pour stocker les réels est celle de la double précision FORTRAN. Si vous voulez afficher les nombres avec plus de précision, tapez la commande format long. Pour revenir au comportement initial : format short.

1.4 Historique des commandes

Toutes les commandes que vous aurez tapé sous MATLAB peuvent être retrouvées et édites grâce aux touches de direction. Appuyez sur * pour remonter dans les commandes précédentes, + pour redescendre et utilisez), et la touche «Backspace» pour éditer une commande.
Pour relancer une commande, inutile de remettre le curseur à la fin, vous appuyez directement sur la touche «Entrée».
Vous pouvez retrouver toutes les commandes commencant par un groupe de lettres. Par exemple pour retrouver toutes les commandes commencent par «plot», tapez plot, puis appuyez plusieurs fois sur *.

Chapitre 2

Variables et fonctions prédéfinies

2.1 Variables

Gros avantage sur les langages classiques : on ne déclare pas les variables. Leur type (entier, réel, complexe) s'affectera automatiquement en fonction du calcul effectue.
Pour affecter une variable, on dit simplement à quoi elle est égale. Exemple :

```
>> a=1.2
a =
1.2000
```

On peut maintenant inclure cette variable dans de nouvelles expressions mathématiques, pour en définir une nouvelle :

```
>> b = 5*a^2+a
b =
8.4000
```

et ensuite utiliser ces deux variables :

```
>> c = a^2 + b^3/2
c =
297.7920
```

Pour voir le contenu d'une variable, on tape son nom :

```
>> b
b =
8.4000
```

On peut aussi faire des calculs en complexe. $p-1$ s'écrit indifféremment i ou j, donc pour définir un complexe :

```
>> a+ b*i
ans =
1.2000 + 8.4000i
```

Le symbole * peut etre omis si la partie imaginaire est une constante numerique. Tous les opérateurs précédents fonctionnent en complexe. Par exemple :

```
>> (a+b*i)^2
ans = -69.1200 +20.1600i
```

La commande **clear** permet d'effacer une partie ou toutes les variables définies jusqu'à présent. Syntaxe :
clear var1 var2 var3 . . .
Si aucune variable n'est spécifiée, toutes les variables seront effacées.
La commande **who** affiche les noms de toutes les variables en cours.

2.2 Fonctions prédéfinies

Toutes les fonctions courantes et moins courantes existent. La plupart d'entre elles fonctionnent en complexe. On retiendra que pour appliquer une fonction à une valeur, il faut mettre cette dernière entre parenthèses. Exemple :

```
>> sin (pi/12)
ans =
0.16589613269342
```

Voici une liste non exhaustive :
– fonctions trigonométriques et inverses : sin, cos, tan, asin, acos, atan
– fonctions hyperboliques (on rajoute «h») : sinh, cosh, tanh, asinh, acosh, atanh
– racine, logarithmes et exponentielles : sqrt, log, log10, exp
– fonctions erreur : erf, erfc
– fonctions de Bessel et Hankel : besselj, bessely, besseli, besselk, besselh. Il faut deux paramètres : l'ordre de la fonction et l'argument lui-même. Ainsi J1(3) s'écrira besselj(1,3)
La notion de fonction est plus générale dans MATLAB, et certaines fonctions peuvent avoir plusieurs entrées (comme besselj par exemple) mais aussi plusieurs sorties.

Chapitre 3

Matrices et tableaux

3.1 Vecteurs, Matrices et opérations associées

L'élément de base dans MATLAB est une matrice dont les dimensions n'ont pas à être fixés explicitement. N'importe quelle variable peut être vue comme une matrice : dans le cas d'un vecteur il s'agit d'une matrice avec 1 seule ligne ou 1 seule colonne et dans le cas d'un scalaire il s'agit d'une matrice de taille 1×1

La saisie d'un vecteur ou d'une matrice s'opère très facilement en effectuant par exemple :

```
» V=[1 0 3*2 sqrt(2)]
V =
1.0000 0 6.0000
1.4142
» A=[4 1 6 ; 3 2 7 ; 4 6 2]
A =
4 1 6
3 2 7
4 6 2
```

On peut extraire un élément du vecteur ou de la matrice en utilisant la commande

```
» V(3)
ans =
6
» A(2,3)
ans =
7
```

L'opérateur « : » joue un rôle important dans MATLAB car il est possible d'extraire tout ou partie de la ligne ou colonne d'un vecteur ou une matrice. Exemple :

```
» A(:,1)
ans =
4
3
4
» A(2,:)
ans =
3 2 7
» V(1:3)
ans =
1 0 6
```

MATLAB étant spécialement dédié au calcul matriciel, celui ci propose donc beaucoup de fonctions propres aux matrices permettant:
• De connaître la taille : length et size
» length (V)
ans =
4
» size (A)
ans =
3 3
• De créer des matrices (ou vecteurs) particuliers : ones, zeros, eye
» U=ones(1,4)
U =
1 1 1 1
» Z=zeros(2,3)
Z =
0 0 0
0 0 0

3.2 Opérations sur les tableaux

3.2.1 Addition et soustraction

Les deux opérateurs sont les mêmes que pour les scalaires. A partir du moment ou les deux tableaux concernes ont la même taille, Le tableau résultant est obtenu en ajoutant ou soustrayant les termes de chaque tableau.

3.2.2 Multiplication, division et puissance terme a terme

Ces opérateurs sont notés .*, ./ et .^ (attention a ne pas oublier le point).
Ils sont prévus pour effectuer des opérations termes a terme sur deux tableau de même taille. Ces symboles sont fondamentaux lorsque l'on veut tracer des courbes.

3.2.3 Multiplication, division et puissance au sens matriciel

Puisque l'on peut manipuler des matrices, il paraıt intéressant de disposer d'un multiplication matricielle. Celle-ci se note simplement ***** et ne doit pas etre confondu avec la multiplication terme a terme. Il va de soi que si l'on écrit **A*B** le nombre de colonnes de A doit etre égal au nombre de lignes de B pour que la multiplication fonctionne.
La division a un sens vis-à-vis des inverses de matrices. Ainsi A/B représente A multiplie (au sens des matrices) a la matrice inverse de B. ATTENTION : même si la matrice B est régulière, elle peut etre mal conditionnée, et la méthode numerique utilisée pour calculer son inverse peut renvoyer des résultats faux.

Chapitre 4

Graphique 1D, 2D

4.1. Tracés graphiques

MATLAB dispose d'une instruction `plot` très complète qui permet d'effectuer rapidement toutes sortes de tracés graphiques.

4.2 Tracer une courbe simple

L'utilisation la plus simple de l'instruction plot est la suivante.
plot (vecteur d'abscisses, vecteur d'ordonnées)
[x1 x2 . . . xn] [y1 y2 . . . yn]
Les vecteurs peuvent être indifféremment ligne ou colonne, pourvu qu'ils soient tous deux de même type. En général ils sont lignes car la génération de listes de valeurs fournit par défaut des vecteurs ligne.

4.2 Affichage graphique des courbes 1D

Matlab permet un grand nombre de types d'affichage 1D et 2D, seuls les plus courant seront décrits ici.

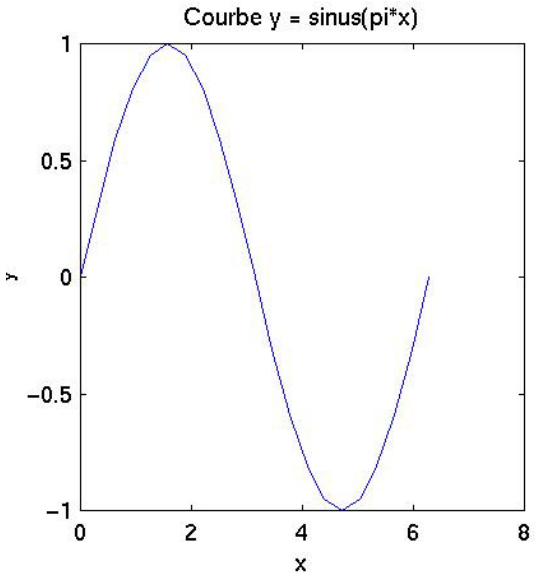
Exemple

```
>> x = 0:0.1:2; y = sin(x*pi);

>> plot(x*pi,y)  % plot(abscisse,ordonnée)
```

On aurait pu tracer la courbe en semilog ou en log avec les fonctions semilogx, semilogy et loglog.

On peut ajouter un titre aux figures ainsi que des labels aux axes avec les commandes title, xlabel, ylabel:

```
>> title('Courbe y = sinus(pi*x)')

>> xlabel('x'); ylabel('y')
```

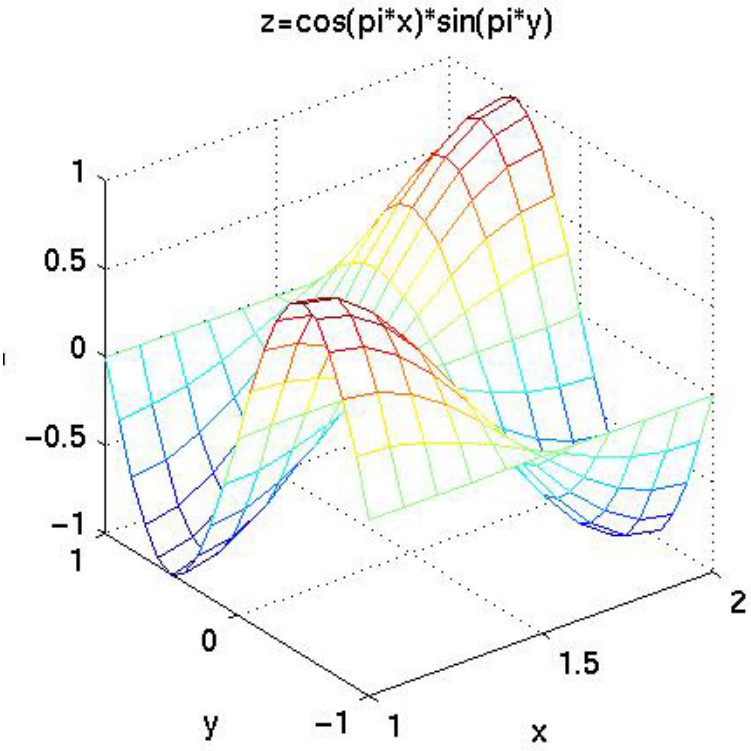


4.3 Affichage graphique des courbes 2D

Avec la commande mesh on peut aisément avoir une représentation d'une fonction 2D. Cependant il faut construire la grille des coordonnées des points en lesquels on a les valeurs de la fonction à l'aide de la fonction 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))  % Coordonnées en x et en y et % valeurs de la
fonction : Z

>> xlabel('x');ylabel('y');zlabel('z');

>> title('z=cos(pi*x)*sin(pi*y)')
```

Mais on peut aussi visualiser les matrices avec les fonctions image, meshc, contour...



4.4 Affichage de plusieurs courbes

On peut bien évidemment vouloir afficher plusieurs courbes à l'écran. Pour cela deux solutions s'offrent à nous :

On peut effectuer plusieurs affichages sur une même figure en utilisant la commande subplot qui subdivise la fenêtre graphique. Sa syntaxe est :

subplot(nombre_lignes, nombre_colonnes, numéro_subdivision)

Les subdivisions sont numérotés de 1 à nombre_lignes*nombre_colonnes, de la gauche vers la droite puis de haut en bas.

Exemple :

```
>> subplot(3,2,1)

>> plot(x,y)

>> subplot(3,2,2)

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

1	2
3	4
5	6

On peut aussi ouvrir une deuxième fenêtre graphique à l'aide de la commande figure.

Le passage d'une fenêtre graphique à une autre pourra alors se faire à la souris ou en précisant le numéro correspondant dans la commande figure(n).

Chapitre 5

Messages et Dialogues

MATLAB autorise l'utilisation de chaînes de caractères :

```
» mesg='IUP1 Antilles Guyane';
» disp(mesg)
IUP1 Antilles Guyane
 mesg1=[mesg ' PI1F'];
» disp(mesg1)
IUP1 Antilles Guyane PI1F
```

On peut convertir des nombres en suite de caractères grâce aux fonctions num2str, int2str et sprintf

```
» y2k=2005;
» disp([mesg1 ,' ',num2str(y2k)])
IUP1 Antilles Guyane PI1F 2005
```

La fonction eval exécute une chaîne de caractère contenant une commande MATLAB .
Exemple

```
» eval(['x','= sqrt(2)'])
x =
1.4142
```

On retrouve le même principe avec la fonction feval mais pour l'évaluation d'une fonction

```
» feval('cos',pi)
ans =
-1
```

L'utilisateur d'un programme peut s'il le souhaite être interrogé en utilisant la commande :

```
» rep_fo=input('Quelle est la fréquence d''accord ? ')
Quelle est la fréquence d'accord ? 2
rep_fo =
2
```

Mais aussi il existe également certaines fonctions comme fprintf, sprintf, etc., Voir l'aide.

Chapitre 6

Environnement, Script, Fonctions

6.1. Environnement et scripts

Il est impératif de travailler dans votre répertoire et non dans celui de matlab ! Après avoir créer votre propre répertoire sous le format suivante c:\users\formation_année\PI1F_TP_Matlab\.

Lors de l'exécution de MATLAB vous pouvez obtenir le répertoire courant en tapant l'instruction cd , et utiliser cette même instruction pour se déplacer vers votre répertoire. L'instruction dir permet de lister le contenu de votre répertoire.

Vous pouvez sauvegarder l'ensemble ou une partie des variables crées lors de votre session MATLAB en utilisant la commande **save** nom_fichier nom_variables
Si vous voulez sauver l'ensemble des variables, il faut supprimer le champ nom_variables dans la commande précédente.

Pour charger les variables dans votre espace de travail il suffit d'exécuter l'instruction : **load** nom_fichier Il n'est pas besoin de spécifier l'extension du fichier car celui ci est un « .mat » par défaut.

MATLAB peut exécuter des suites d'instructions (programme informatique ….) contenu dans des fichiers de type texte ayant pour extension « .m ». Ce *m-file* (fichiers MATLAB) peut être édités avec un simple éditeur de texte comme NOTEPAD sous l'environnement WINDOWS. Pour exécuter ce script il suffit de se placer dans le répertoire où se trouve le fichier .m (Donc **forcement** dans votre répertoire) et de taper le nom du fichier sans son extension dans la fenêtre de commande MATLAB.

Il est donc fortement conseillé de créer ce genre de fichiers pour les problèmes que vous aurez à résoudre car cela augmente la rapidité de développement et permet une grande souplesse.

Bien entendu des fichiers .m peuvent s'appeler l'un l'autre mais en sachant que les variables sont par défaut locales au script en cours. Pour résoudre ce problème il suffit de changer le script .m en une fonction MATLAB portant la même extension.

En résumé, pour que vos fonctions et scripts soient accessibles à partir de la ligne de commande Matlab, il faut au début de chaque session Matlab, déclarer vos répertoires de travail. Ceci se fait à l'aide de la commande path :

path (path,'c:\ users\formation_année \ PI1F_TP_Matlab\path_personnel) %

Ceci rajoute aux chemins déjà existants (que l'on obtient à l'aide de path) le chemin mis entre quotes.

6.2 Fonctions

Pour créer une fonction il suffit d'utiliser le mot clef function comme ceci :

```
function R=reqv(Ra,Rb)
%*************************************
%* Calcul de la résistance équivalente *
%* d'un circuit de 2 résistances en // *
%* syntaxe R=Reqv(Ra,Rb) *
%*************************************
R=(Ra*Rb)/(Ra+Rb);
```

Ce fichier doit obligatoirement être enregistrer dans un fichier de même nom que la fonction, c'est à dire ici : reqv.m

Les lignes qui commencent par un symbole % sont des lignes de commentaires. On peut les placer n' importe où dans le fichier, mais dans notre cas, elles ont un rôle bien particulier puisque le texte situé derrière le symbole % apparaît lorsque l'on tape l'instruction help reqv

C'est la méthode la plus généraliste, et elle permet notamment de réaliser des fonctions ayant plusieurs sorties. Exemple d'une fonction (sincos).

L'ordre des opérations est le suivant :

5.2 Fonctions 35

1. Editer un nouveau fichier appele sincos.m
2. Taper les lignes suivantes :

```
function s = sincos(x)
s = sin(x)-x.*cos(x);
```

3. Sauvegardez

Le résultat est le même que précédemment :

```
>> sincos(pi/12)
ans =
0.0059
```

On remarquera plusieurs choses :

– l'utilisation de .* pour que la fonction soit applicable à des tableaux.

– la variable s n'est là que pour spécifier la sortie de la fonction.

– l'emploi de ; à la fin de la ligne de calcul, sans quoi MATLAB afficherait deux fois le résultat de la fonction.

Donc encore une règle :

Lorsque l'on définit une fonction dans un fichier, il est préférable de mettre un ; `a la fin de chaque commande constituant la fonction. Attention cependant `a ne pas en mettre sur la première ligne.

Un autre point important méritant un cadre :

Le nom du fichier doit porter l'extension .m et le nom du fichier sans suffixe doit être exactement le nom de la fonction (apparaissant après le mot-cle function

En ce qui concerne l'endroit de l'arborescence où le fichier doit être place, la règle est la même que pour les fichiers de commande (section 5.1.2).

Il est permis de comparer la concision de MATLAB pour la déclaration d'une fonction à la lourdeur du FORTRAN ou de tout autre langage, où les paramètres formels et toutes les variables locales doivent être déclarées.

Voyons maintenant comment définir une fonction comportant plusieurs sorties. On veut réaliser une fonction appelée cart2pol qui convertit des coordonnées cartésiennes (x, y) (entrées de la fonction) en coordonnées polaires (r, _) (sorties de la fonction). Voila le

contenu du fichier cart2pol.m3 :
3Cette fonction est ici extrêmement mal écrite (trouvez pourquoi), mais de toutes fa¸cons elle existe déjà
toute faite sous MATLAB. . .
36 5.2 Fonctions
function [r, theta] = cart2pol (x, y)
r = sqrt(x.^2 + y.^2);
theta = atan (y./x);
On remarque que les deux variables de sortie sont mises entre crochets, et séparées par une virgule.
Pour utiliser cette fonction, on écrira par exemple :
>> [rr,tt] = cart2pol(1,1)
rr =
1.4142
tt =
0.7854
On affecte donc simultanément deux variables rr et tt avec les deux sorties de la fonction, en mettant ces deux variables dans des crochets, et séparés par une virgule. Les crochets ici n'ont bien sur pas le même sens que pour les tableaux.
Il est possible de ne récupérer que la première sortie de la fonction. MATLAB utilise souvent ce principe pour définir des fonctions ayant une sortie «principale» et des sorties «optionnelles»4.
Ainsi, pour notre fonction, si une seule variable de sortie est spécifiée, seule la valeur du rayon polaire est renvoyée. Si la fonction est appelée sans variable de sortie, c'est ans qui prend la valeur du rayon polaire :
>> cart2pol(1,1)
ans =
1.4142

6.3 Portée des variables

A l'intérieur des fonctions comme celle que nous venons d'écrire, vous avez le droit de manipuler trois types de variables :
– Les variables d'entrée de la fonction. Vous ne pouvez pas modifier leurs valeurs.
– Les variables de sortie de la fonction. Vous devez leur affecter une valeur.
– Les variables locales. Ce sont des variables temporaires pour découper des calculs par exemple. Elles n'ont de sens qu'à l'intérieur de la fonction et ne seront pas «vues» de l'extérieur

6.4 Boucles et contrôles

MATLAB comme tous les langages de programmation propose des instructions de boucle : for et de contrôle if else que vous allez pouvoir utiliser en utilisant bien entendu la commande help !
Remarque importante sur l'utilisation d'une boucle for :
L'instruction
» for i=1:100,x(i)=i;end;
peut être avantageusement remplacé par
» x=1:100;
En conséquence on veillera à ne pas trop utiliser cette instruction, et l'on préférera utiliser les ressources de base de MATLAB !