INTRODUCTION A MATLAB

Année académique 2008-2009

Cours de 1re année du grade de bachelier en sciences de l'ingénieur, orientation ingénieur civil

Professeurs Quentin Louveaux, Olivier Bruls, et Frédéric Nguyen

Table des matières

1 QU'EST CE QUE MATLAB ?

MATLAB est un logiciel interactif qui fournit à l'utilisateur un environnement lui permettant de réaliser un grand nombre de calculs, en particulier ceux où les matrices interviennent. L'élément de base est un tableau qui ne demande pas de dimensionnement préalable. Ceci vous permet de résoudre de nombreux problèmes techniques de calcul numérique bien plus rapidement que si vous deviez écrire un programme dans un langage tel que C ou Fortran. MATLAB s'opère depuis une session de commandes en ligne (voir figure ci-dessus). Celles-ci peuvent être exécutées une à une ou bien être sauvées dans un script afin de l'exécuter comme un programme. Il existe un grand nombre de fonctions et commandes MATLAB qui vous permettent de réaliser :

- des opérations vectorielles ou matricielles
- des calculs statistiques
- de la visualisation de données et images
- …

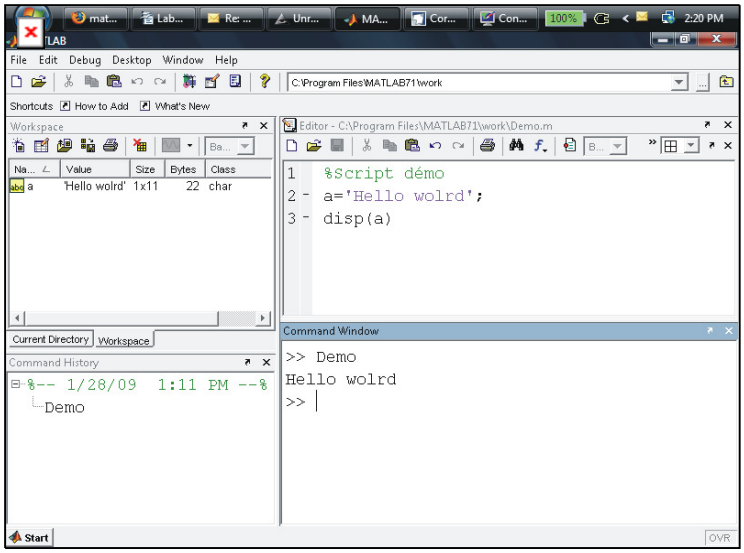

Illustration 1: Le logiciel MATLAB s'utilise généralement avec les fenêtres suivantes: A données en mémoire, B éditeur de script, C lignes de commandes, D historique des commandes, E adresse de navigation

Le nom MATLAB est en fait l'abréviation de MATrix LABoratory. Le but initial de Cleve Moler, concepteur de ce logiciel en 1981 était de fournir un accès aisé aux logiciels de haute qualité issus des projets LINPACK (LIN pour « linear » = linéaire) et EISPACK (EIS pour « eigenvalues » = valeurs propres). Aujourd'hui, la philosophie de base est restée la même et MATLAB incorpore les bibliothèques LAPACK (Linear Algebra) et BLAS (Basic Linear Algebra Subroutines) qui sont ce qu'il y a de mieux en matière de logiciel de calcul matriciel. La version dont vous disposez sur ces PC est commercialisée par la société :

The Math Works, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098
USA.

URL : http://www.mathworks.com

fondée et dirigée par Cleve Moler.

2 COMMENT DEBUTER ET TERMINER UNE SESSION MATLAB ?

Depuis les salles de laboratoire

Pour accéder à l'ordinateur de type PC, il suffit d'introduire votre *login* et votre *password*, suivi de la touche <RETURN>. Pour sortir de session, il faut appuyer sur le bouton rouge de la barre en haut de l'écran et choisir l'option « log out ».

Depuis internet

Utilisez les informations se trouvant à l'adresse suivante :

http://www.montefiore.ulg.ac.be/services/ail/cours/algo1/putty.html

Lors de votre première connexion, ouvrez un terminal et lancer la commande

/home/algo/LAM/idstudent

Si vous vous connectez de chez vous, il est conseillé de lancer MATLAB avec la commande

MATLAB -nojvm

Informations générales

Pour commencer, ouvrez un terminal en choisissant ce programme dans le menu. Ensuite, pour débuter une session MATLAB il faut introduire la commande

MATLAB (suivi de <RETURN>)

Il apparaît alors à l'écran une fenêtre dont la « sous-fenêtre » **Command Window** constitue l'espace de travail interactif. Celui-ci contient un message d'identification suivi du symbole :

```
>>
```

qui vous indique que c'est à vous de jouer : MATLAB est prêt à exécuter toute commande que vous allez lui fournir.

Si par exemple vous taper dans la fenêtre de commandes

```
a = 1          (suivi de  <RETURN>)
```

le programme vous répond

```
a    =

     1
```

La variable `a` s'enregistre dans l'espace de travail (workspace) et la valeur 1 lui est assignée.

Remarquez que pour être exécutée, toute commande doit être suivie de <RETURN> ; nous ne répéterons donc plus cette exigence dans la suite.

On termine une session en allant dans le menu <File> en haut à gauche et en cliquant sur <Exit MATLAB>. Cette commande provoque le retour au système d'exploitation ou au programme qui a appelé MATLAB.

EXERCICES

EX 2-1

Entrer en session MATLAB.
Exécuter successivement les commandes

```
a  =  -1

B  =  LOG(a)

B=log(a)

C  =  exp(B)
```

Qu'en pensez-vous ?

Maintenant vous devez avoir remarqué que MATLAB est sensible aux majuscules et minuscules. Le programme accepte indifféremment majuscules et minuscules, mais les fonctions prédéfinies (comme `log` ou `exp`) doivent être écrites en minuscules. Remarquez aussi que les noms de ces fonctions prédéfinies ne sont pas réservés.

3 AIDE INTERACTIVE (votre meilleur ami)

Une aide interactive est disponible pour toutes les commandes MATLAB. Pour obtenir la liste des commandes reprises dans cette aide, exécuter la commande

`help`

Pour obtenir l'aide relative à une commande, exécuter HELP *commande*. Par exemple

`help abs`

fournit la description de la fonction ABS (valeur absolue).

`help \`

nous fournit l'utilisation du caractère '\'.

Finalement, vous vous détendrez un moment si vous suivez la suggestion contenue dans la réponse à la commande

help why

helpwin

Ouvre une fenêtre d'aide générale

4 COMMANDES ET VARIABLES

MATLAB est un langage d'expression. Les expressions introduites par l'utilisateur sont interprétées et évaluées par le système. Les commandes MATLAB sont de la forme

```
variable = expression
```

ou simplement

```
expression
```

Les expressions sont composées d'opérateurs et autres caractères spéciaux de fonctions et de noms de variables. L'évaluation de l'expression produit une matrice, qui est alors écrite sur l'écran et stockée pour une utilisation future.

Si le nom de la variable et le signe = sont omis (cas 2), une variable de nom « ans » est automatiquement créée.

Si la commande

1900/81

est introduite, elle produit l'affichage :

```
ans    =

   23.4568
```

Le symbole ';' à la fin de la commande supprime l'affichage du résultat à l'écran.

Plusieurs commandes peuvent être écrites sur une ligne. Elles seront alors séparées par ';'ou par ','

Le symbole '%' dans une ligne a pour effet que le reste de la ligne ne sera pas exécuté (ceci permet d'insérer des commentaires dans un fichier d'exécution).

Si une commande ne peut être écrite sur une seule ligne, il suffira d'ajouter à la fin de la première ligne au moins trois '.' et MATLAB concaténera cette ligne et la suivante (jusqu'à un maximum de 1024 caractères). Ces points de « concaténation » et un nombre éventuel, doivent être séparés par un blanc (espace vide).

Par exemple, la commande

S = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 …
-1/8 + 1/9 - 1/10 + 1/11 -1/12;

évalue la valeur de cette somme, assigne cette valeur à la variable s mais n'imprime rien du tout sur l'écran. Si l'on veut connaître la valeur de s, il suffit de taper

S

pour obtenir

```
S      =

    0.6532
```

Les noms des variables sont formés par une lettre suivie par une suite de n'importe quel nombre ou lettre.

5 ESPACE DE TRAVAIL

Les commandes effectuées jusqu'à maintenant ont créé des variables qui sont stockées dans *l'espace de travail* de MATLAB. La commande

`who`

donne la liste des variables contenues dans cet espace de travail. Pour en savoir plus, il faut exécuter la commande :

`whos`

Si l'on souhaite en savoir plus sur une variable A contenue dans l'espace de travail, la commande

`size (A)`

fournit les dimensions (nombre de lignes nombre de colonnes) de cette variable.

La commande

`clear`

détruit toutes les variables de l'espace de travail.

EXERCICES

EX 5-1

Exécuter les commandes

who, whos, e = sqrt(-1), f=4*atan(1)

Combien d'octets occupe un nombre « réel », un nombre complexe ?

Exécuter les commandes

clear, who, whos

Que constatez-vous ?

6 INTRODUCTION DE MATRICES

MATLAB ne travaille qu'avec une seule sorte d'objet : une matrice numérique rectangulaire dont les éléments peuvent être complexes. Dans certains cas, une signification particulière peut être attribuée à des matrices 1 x 1, qui sont des scalaires et à des matrices ne comportant qu'une seule ligne ou une seule colonne, qui sont des vecteurs.

Les matrices peuvent être introduites dans MATLAB des quatre façons suivantes

- liste explicite des éléments
- utilisation des commandes du programme
- lecture d'un fichier extérieur
- lecture d'un fichier de données (commandes `SAVE, LOAD`)

Le langage MATLAB ne contient aucune commande de dimension ou de déclaration de type. Le stockage est alloué automatiquement tant qu'il reste de la mémoire disponible.

La liste explicite est la façon la plus simple d'introduire des matrices de petite taille. Cette liste est précédée du symbole '[' et suivie du symbole ']'. Elle est constituée ligne par ligne, la fin d'une ligne étant indiquée par le symbole ';'. Les éléments d'une ligne doivent être séparés par un espace ou une virgule.

Ainsi, l'entrée de la ligne

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

donnera comme résultat à l'écran

```
 A      =

      1     2     3
      4     5     6
      7     8     9
```

La matrice A est alors stockée pour une utilisation future.

Les éléments des matrices peuvent être toute expression reconnue par MATLAB, par exemple

B = [-2.5; log(1); 4*atan(1)]

```
 B      =

   -2.5000
         0
    3.1416
```

```
C = [ sqrt(2)   10/2*5   (1+2+3)/2   exp(1) ]

 C      =

    1.4142   25.0000    3.0000    2.7183
```

Les matrices peuvent également être définies sur plusieurs lignes d'entrée, le <RETURN> remplaçant le symbole ';'. Ainsi A aurait pu également être introduite par

```
  A     =     [  1     2     3          <RETURN>
           4     5     6          <RETURN>
           7     8     9     ]          <RETURN>



A      =

      1     2     3
      4     5     6
      7     8     9
```

Certaines commandes de MATLAB permettent la création de matrices spéciales. Ainsi par exemple, la commande

```
E3  =  eye (3)
```

crée une matrice unité d'ordre 3 et la stocke sous le nom E3 ;

la commande

```
ONEA  =  ones (3,3)
```

crée une matrice d'éléments tous égaux à 1 et de dimensions 3 X 3 et la stocke sous le nom ONEA.

Les matrices peuvent aussi être introduites par lecture d'un fichier extérieur. Si par exemple, le fichier ‘xyz' contient les cinq lignes

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

alors, la commande xyz lit le fichier et assigne donc à A les valeurs indiquées. L'introduction de matrice par lecture de fichier de données sera expliquée plus loin.

7 ELEMENTS DE MATRICES

Un élément individuel d'une matrice est déterminé par ses indices entre parenthèses. Ainsi la commande

A(2,3)

fournit

```
ans    =

     6
```

Mais on peut également obtenir un bloc d'éléments d'une matrice. Par exemple,

A(1, :)	fournit la 1ere ligne de A
A(:, 2)	fournit la 2eme colonne de A
A([1 2], :) ou A(1:2, :)	fournit la matrice composée des 2 premières lignes de A
A(:, [2 3]) ou A(:, 2:3)	fournit la matrice composée des colonnes 2 et 3 de A
A(:)	spécial ! Fournit quoi ?

Le symbole ':' se révèle donc particulièrement intéressant. Il est aussi utilisé dans les expressions j:k et j:i:k qui permettent de créer des vecteurs :

j : k	est identique à [j, j+1, j+2, ..., k]; et est vide si j>k
j : i : k	est identique à [j, j+i, j+2i, ..., k]; et est vide si i>0 et j>k ou si i<0 et j<k

Les valeurs de i,j,k ne doivent pas être nécessairement entières.

Le dimensionnement des matrices est réalisé automatiquement dans MATLAB. Ainsi, la commande

D = [1 2 3; 4 5 6];

suivie de la commande

D = [1 0; 0 1]

Fournit

```
D      =

     1     0
     0     1
```

En d'autres mots, le système en « sait assez » pour reconnaître que la matrice D a été changée d'une matrice 2 x 3 à une matrice 2 x 2. Un autre aspect de ce dimensionnement automatique apparaît si l'on exécute la commande

D(2,4) = 5

qui fournit

```
D      =

     1     0     0     0
     0     1     0     5
```

Cette fois, MATLAB rend D suffisamment grande pour que la commande ait un sens. Les « trous » sont remplis par des éléments nuls.

Des matrices peuvent aussi être formées à partir de plus petites matrices considérées comme éléments. Ainsi la suite de commandes

```
B = [10;11;12];   C = [13 14 15 16];   D = [ A B; C]
```

fournit

```
D      =

     1     2     3    10
     4     5     6    11
     7     8     9    12
    13    14    15    16
```

Il faut évidemment que les dimensions des blocs consécutifs soient compatibles ! Par exemple, la commande

```
d = [ A  C;  B]
```

donne lieu à un message d'erreur :

```
??? Error using ==> horzcat
All matrices on a row in the bracketed expression must have
the same number of rows.
```

EXERCICES

EX 7-1

Vider la mémoire de travail (commande **clear**)
Créer par introduction directe la matrice A :

$$A = \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$$

Créer au moyen d'une commande du type i : j : k le vecteur ligne :

```
b= [ 10  12  14 ]
```

Créer au moyen de la commande **ones** (taper **help ones**) le vecteur colonne :

$$c = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}$$

Créer les matrices

$$D = \begin{vmatrix} A & c \\ b & 10 \end{vmatrix}$$

$$E = \begin{vmatrix} A & b \\ c & 10 \end{vmatrix}$$

EX 7-2

Remplacer l'élément (2,3) de la matrice D par le nombre 7.
Extraire de la matrice D la matrice formée par l'intersection des lignes 1, 2, 3 et des colonnes 1,2.

Extraire de la matrice D la matrice formée par l'intersection des lignes 1, 2, 3 et des colonnes 1,4
Échanger les lignes 1 et 4 de la matrice D.

8 NOMBRES ET EXPRESSIONS ARITHMETIQUES, OPERATIONS MATHEMATIQUES

La notation décimale conventionnelle avec le point décimal optionnel et le signe moins en tête est utilisée pour l'introduction et l'affichage des nombres. Une puissance de 10 peut être ajoutée en suffixe ; dans ce cas tout espace doit absolument être évité.

Exemples corrects

```
4          -27        0.007           .789
5.82       7.2325     82.895E2   0.85672-12
```

Exemples incorrects

```
4,62       8 E5       9.82 e12
```

Des expressions peuvent être construites en utilisant les opérateurs arithmétiques et règles de priorité usuels. Ces opérateurs sont

+	addition	/	division à droite
-	soustraction	\	division à gauche
*	multiplication	^	puissance (obtenu avec les touches AltGr-6)

L'opérateur de division à gauche nous sera utile dans le cadre des opérations matricielles. Dans le cas scalaire les expressions 1/2 et 2\1 ont la même valeur numérique 0.5000.

Les parenthèses sont utilisées de façon standard pour altérer les règles de priorité usuelles des opérateurs arithmétiques.

9 NOMBRES ET MATRICES COMPLEXES

Si le résultat d'une commande est un nombre complexe à partie imaginaire non nulle, l'affichage de ce résultat par MATLAB utilise toujours la lettre i. Ainsi la commande

```
sqrt(-4)
```

fournit

```
ans   =

    0.0000 + 2.0000i
```

Des nombres complexes peuvent être introduits; par exemple

```
z = 6 + 8*i
```

Des matrices complexes peuvent être introduites au moins de deux façons différentes illustrées ci-dessous

```
A =   [ 1   2;  3  4 ]    +   i*  [ 5   6;  7  8 ]
```

ou

```
A =   [ 1+5*i  2+6*i;   3+7*i  4+8*i ]
```

Lorsque des nombres complexes sont introduits comme éléments de matrice entre crochets ([]), il est important d'éviter tout espace blanc de part et d'autre du signe + , car l'expression 1 + 5*i représente l'addition de deux nombres distincts.

EXERCICES

EX 9-1

Introduire le nombre complexe 3+4*i sous forme algébrique et sous forme exponentielle. Faire la différence des deux résultats.

10 FORMAT D'AFFICHAGE

Le résultat de toute commande MATLAB, à moins que celle-ci ne soit suivie du caractère ';', est affiché à l'écran. Le format de cet affichage est contrôlé par la commande format suivie (éventuellement) de

`short, short e, short g, long, long e, long g, hex, bank, rat…`

Ces commandes n'influencent aucunement la précision des calculs ou les valeurs des éléments de l'espace de travail, seul l'affichage est affecté.

Par défaut, en entrée de session, le format est « `short` », c'est à dire un format fixe à cinq chiffres. Ainsi les commandes

```
format short,    x = [ 1   1.00001 ]
```

produisent

```
 X      =

    1.0000    1.0000
```

Rassurons nous, rien n'a été perdu, mais dans ce cas-ci, le format utilisé est trompeur. Pour en savoir plus, nous exécutons les commandes

```
format short e,    x
```

qui fournissent

```
 X      =

    1.0000e+00   1.0000e+00
```

soit pas grand chose de plus que précédemment ! Enquêtons donc plus avant avec

```
format long,    x
```

qui cette fois fournit le résultat

```
 X      =

   1.00000000000000   1.00001000000000
```

mieux conforme à ce que nous avions en tête au départ.

EXERCICES

EX 10-1

Pour vous convaincre que les formats d'affichage n'affectent en rien les calculs ou les valeurs des éléments, effectuez successivement les séquences de commande suivante

```
format short,    x = [ 1.00001    1 ]

x(1) - x(2)

format long,    x(1) - x(2)

format short,    ans
```

Surpris ? Pourquoi ? Quelle explication ?

11 OPERATIONS MATRICIELLES

Les opérations matricielles sont évidemment fondamentales pour MATLAB. En voici une description succincte

'	transposition d'une matrice
+	addition matricielle
-	soustraction matricielle
*	multiplication matricielle
/	division matricielle à droite
\	division matricielle à gauche
^	puissance matricielle (introduit à l'aide des touches AltGr-6)

Les dimensions des matrices intervenant dans ces opérations doivent évidemment être compatibles pour l'opération envisagée.

Si tel n'est pas le cas, un message d'erreur sera affiché par MATLAB.

Le caractère spécial (') (apostrophe) effectue la transposition d'une matrice. Par exemple, la séquence de commande

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

fournit

```
A      =

     1     2     3
     4     5     6
     7     8     9

 B      =

     1     4     7
     2     5     8
     3     6     9
```

et la séquence

```
Y = [ 1 -2 1 ],   X = Y'
```

fournit

```
 Y      =

     1    -2     1

 X      =

     1
    -2
     1
```

Attention ! Lorsque A est complexe, A' est la transposée conjuguée !

L'addition et la soustraction dénotée respectivement par + et - n'a de sens que si les dimensions des deux matrices sont les mêmes.

La multiplication X * Y n'a de sens que si le nombre de colonnes de X est égal au nombre de lignes de Y. La multiplication d'une matrice par un vecteur colonne ou d'un vecteur ligne par une matrice n'est qu'un cas particulier de multiplication matricielle. Ainsi, la commande

```
A*X
```

fournit

```
 ans    =

     0
     0
     0
```

ce qui indique évidemment que la matrice A est singulière.

La multiplication d'une matrice par un scalaire ou d'un scalaire par une matrice est tout à fait licite. Par exemple

```
2*Y
```

fournit

```
ans    =

     2    -4     2
```

Il y a deux symboles de « division matricielle » dans MATLAB le caractère ' / ' et le caractère '\'. D'une façon générale

X = A\B est une solution de A*X = B

X = B/A est une solution de X*A = B

La résolution de systèmes n'ayant pas encore été abordée au cours théorique, ne vous en faites pas si les notions suivantes vous semblent un peu floue. La division à gauche est définie dès que B a autant de lignes que A. Si A est carrée le calcul de A\B est effectué par l'élimination de Gauss. Si A est proche de la singularité, un message d'avertissement est affiché. Si A est non carrée, les équations sont résolues au sens des moindres carrés, qu'elles soient sous-déterminées ou sur-déterminées. Une façon simple de retenir la signification de la division à gauche est de penser comme suit :

X = A\B est formellement équivalent à $X = A^{-1}B$

La division à droite B/A est définie en terme de la division à gauche par B/A = (A'\B')'.

Lorsque l'on veut résoudre un système linéaire Ax = b, on utilisera donc l'opérateur \ de division à gauche.

En guise d'exemple, revenons à la matrice A introduite précédemment. Modifions-là afin qu'elle ne soit plus singulière, définissons un vecteur x composé d'éléments tous égaux à 1, calculons le second membre correspondant b = Ax ainsi que la solution xsol de ce système. Tout ceci s'écrit au moyen de la séquence de commandes

```
A(3,3) = 9.1;    X = [ 1 1 1 ]';    B = A*X;    XSOL = A\B
```

qui fournit

```
XSOL  =

    1.0000
    1.0000
    1.0000
```

Vérifions la solution en exécutant

```
X - XSOL
```

qui fournit

```
ans    =

    1.0e-13 *
```

```
-0.2798
 0.5596
-0.2798
```

Le fait que l'on n'obtienne pas un vecteur nul est évidemment dû aux erreurs d'arrondi. Nous aurons l'occasion de revenir sur ce point plus tard.

Si A est une matrice carrée (et seulement dans ce cas), l'opération A^p où p est un scalaire élève la matrice A à la puissance p.

EXERCICES

EX 11-1

Il se pourrait que l'exécution de la commande X – XSOL ne donne pas sur votre ordinateur le résultat ci-dessus. Comment peut-on l'expliquer?

12 OPERATIONS ELEMENT PAR ELEMENT

Il existe également des opérations de multiplication et division élément par élément. Elles sont définies par les commandes

.* ./ .\

C = A.*B fournit c(i,j) = a(i,j)*b(i,j)

C = A./B fournit c(i,j) = a(i,j)/b(i,j)

C = A.\B fournit c(i,j) = b(i,j)/a(i,j)

On vérifie par exemple que la suite de commandes

```
A = rand(4);   A./A
```

fournit bien une matrice dont tous les éléments sont égaux à un.

13 CREATION ET MANIPULATION DE MATRICES

Un certain nombre de commandes sont particulièrement intéressantes pour la création et la manipulation de matrices. En voici une description succincte.

`eye` matrice unité,

`ones` matrice dont tous les éléments sont égaux à un,

`rand` matrice d'éléments aléatoires,

`diag` matrice diagonale,

`tril` extraction de la partie triangulaire inférieure d'une matrice,

`triu` extraction de la partie triangulaire supérieure d'une matrice,

Pour plus de détails, consulter le « help »

Voici un exemple de la facilité de création d'une matrice au moyen de ces commandes : une matrice d'ordre 6 dont tous les éléments de la diagonale sont égaux à 4 et les éléments au-dessus et en dessous de la diagonale sont égaux à 1, tandis que tous les autres éléments sont nuls peut être créée au moyen de la commande.

```
tril ( triu ( ones ( 6 ), -1 ), 1 ) + 3*eye(6)
```

EXERCICES

EX 13-1

Créer une matrice « unité » 5X7. Créer une matrice 3X4 d'éléments nuls.
Créer la matrice diag { π, 2π,..., 7π }.

EX 13-2

Créer en une seule commande une matrice triangulaire supérieure d'ordre 7 dont tous les éléments de la $j^{ème}$ colonne sont égaux à j.

EX 13-3

Créer en une seule commande une matrice triangulaire supérieure d'ordre 7 dont tous les éléments de la $i^{ème}$ ligne sont égaux à i.

EX 13-4

Créer en une seule commande une matrice dont les éléments en dessous de la diagonale valent 1, les éléments au-dessus de la diagonale valent -1 et les éléments de la diagonale sont nuls.

14 FONCTIONS MATHEMATIQUES ELEMENTAIRES

Un certain nombre de fonctions mathématiques de base sont implémentées dans MATLAB

`abs` valeur absolue ou module complexe,

`conj` complexe conjugué,

`imag` partie imaginaire,

`real` partie réelle,

`norm` norme, vectorielle ou matricielle (diverses options...),

`prod` produit de tous les éléments d'un argument vectoriel,

`sum` somme de tous les éléments d'un argument vectoriel,

`round` arrondit tous les éléments aux entiers les plus proches.

EXERCICES

EX 14-1

Comment construire une matrice 4X4 d'éléments 0 ou 1 "aléatoires" ?

EX 14-2

Calculer norm(A, 1) et norm(A, 'inf') où A est une matrice symétrique. Que constatez-vous?

EX 14-3

Calculer la somme des dix premiers nombres entiers.

15 POSSIBILITÉS GRAPHIQUES

Les possibilités graphiques de MATLAB sont innombrables !

Pour créer le graphe de la fonction sinus par exemple, nous commençons par définir deux vecteurs

```
t = 0:pi/20:2*pi;  y = sin(t);
```

ensuite nous construisons le graphe au moyen de la commande

```
plot(t,y)
```

on peut superposer un quadrillage

```
grid
```

mettre un titre, mettre des étiquettes aux axes

```
title('sinus'),  xlabel('temps t'), ylabel('sinus(t)')
```

On peut aussi représenter des fonctions de deux variables. Par exemple, pour représenter la fonction

$z = x\ e^{-x^2-y^2}$

on commence par définir une grille dans le plan (x,y) et une matrice représentant z(x,y)

```
[X,Y] = meshgrid(-2:.2:2, -2:.2:3);  Z=X.*exp(-X.*X - Y.*Y);
```

ensuite on crée le graphe par la commande

```
mesh(Z)
```

MATLAB accepte aussi de traîter des images grâce entre autres aux commandes <image> et <imread>. La commande <A=imread('filename')> lit une image (couleur ou en niveaux de gris) à partir du fichier spécifié par 'filename'. Les données de cette image sont placées dans la matrice A. La taille de cette matrice A sera évidement déterminée par le type d'image utilisée.

Ainsi par exemple :

```
street1=imread('street1.jpg');
```

La commande whos permet de déterminer la taille de cette image.

```
whos street1
```

Une fois l'image stockée dans une matrice on peut l'afficher en utilisant la commande <image>. Chaque élément de la matrice spécifie la couleur (ou le niveau de gris) d'une zone de l'image.

```
image(street1);
axis equal; axis off
```

A quoi les commandes axis servent-elles?
Taper les commandes suivantes:

```
street2=imread('street2.jpg');
streets=street1+street2;
image(streets);
axis equal; axis off
```

Que constatez-vous? Quelle explication pouvez-vous donner à ce que vous voyez?

Pour plus d'informations, consultez la documentation.

16 INSTRUCTIONS DE CONTROLE DE FLUX

L'instruction <for> permet à des commandes d'être répétées un certain nombre de fois. La forme globale est

for variable = expression , commande,..., commande, end

En général, l'expression peut être une matrice, auquel cas les colonnes sont stockées une à une dans la variable et les commandes sont exécutées jusqu'au <end>. Plus simplement, l'expression sera souvent du type i : j : k.

Si par exemple, nous voulons construire un vecteur de composantes 2, 4,..., 12, on exécutera l'une ou l'autre des trois commandes suivantes

```
for   I  =  1 : 6,   T(I) = 2*I, end

for   I  =  1 : 6,   U(I) = 2*I ; U, end

for   I  =  1 : 6,   V(I) = 2*I ; end,   V
```

Quelles différences observez-vous ?

Des boucles <for> peuvent être imbriquées. Ainsi, la commande

```
for   I  =  1 : 3,   for   J  =  1 : 3,   A(I,J) = I+J-1 ;
end, end,   A
```

fournit

```
 A      =

      1     2     3
      2     3     4
      3     4     5
```

La commande <while> permet de répéter des commandes un nombre indéfini de fois. La forme générale est

while expr oprel expr, commande,..., commande, end

où <oprel> est défini comme suit :

== égal

< inférieur

> supérieur

<= inférieur ou égal

>= supérieur ou égal

~= non égal

Par exemple, la commande

```
E  =  1;   while  1+E > 1,   E=E/2 ;   end,   E=2*E
```

fournit

```
 E      =

    2.2204e-16
```

Quelle interprétation peut-on donner à ce résultat ?

Il existe également un <if> dont la forme générale est

```
if     expr  oprel  expr,
       commande,..., commande,
elseif       expression,
       commande,..., commande,
else
       commande,..., commande,
end
```

A titre d'exemple, essayez (plusieurs fois) la commande

```
A = rand(5) ; if sum(A(:))/25  >= 0.5, B = round(A), else B = A, end
```

À ce stade, il convient de se rappeler que MATLAB est un interpréteur de commandes Par conséquent, dans une boucle, chaque fois que le groupe de commande est exécuté, il est retraduit. Ceci ralentit évidemment l'exécution. Les boucles sont donc à éviter autant que possible, ce qui est souvent facile étant donné la richesse des commandes de MATLAB. Ainsi par exemple, pour créer un vecteur dont la $i^{ème}$ composante est égale à i, on préférera à la commande

```
for   i  =  1 : n,   t (i) = i; end
```

la commande

```
t = [ 1 : n ] ;
```

qui fait exactement la même chose, mais beaucoup plus rapidement.

EX 16-1

Les intégrales

$$y(n) = \int_0^1 (x^n)/(x+5)\, dx$$

peuvent être calculées " simplement " si l'on observe qu'elles satisfont la récurrence

$$y(n) + 5y(n-1) = 1/n, \qquad y(0) = \ln(6/5)$$

Calculez les valeurs successives de y(n) jusqu'à y(25). Tracez graphiquement la solution. Qu'observez-vous?
Utilisez maintenant la formule suivant

$$y(n-1) = (1/n - y(n)) / 5$$

et partez de y(25) = 0 pour calculer les y(n) jusqu'à y(0). Tracez graphiquement la solution. Qu'observez-vous?

17 VALEURS ET VECTEURS PROPRES

Les fonctions utiles pour la recherche de valeurs et vecteurs propres sont

eig calcul des valeurs et vecteurs propres,

poly polynôme caractéristique.

Si par exemple on veut calculer les valeurs propres de la matrice

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

```
 A      =
```

```
     1     2     3
     4     5     6
     7     8     9
```

il suffit d'exécuter la commande

eig(A)

```
 ans   =

   16.1168
   -1.1168
   -0.0000
```

Le vecteur ans contient les valeurs propres de A. Changeons de format d'affichage afin d'avoir une meilleure idée de la dernière valeur propre

format short e, ans

```
 ans   =

    1.6117e+01
   -1.1168e+00
   -1.0307e-16
```

Si l'on s'intéresse également aux vecteurs propres on exécutera

format short, [V, D] = eig(A)

```
 V     =

   -0.2320   -0.7858    0.4082
   -0.5253   -0.0868   -0.8165
   -0.8187    0.6123    0.4082


 D     =

   16.1168         0         0
         0   -1.1168         0
         0         0   -0.0000
```

La matrice D contient les valeurs propres sur la diagonale et la matrice V contient les vecteurs propres, de sorte que l'on a A*V = V * D. Contrôlons ces valeurs propres en calculant A - V*D*V^{-1}, soit

A - V*D*inv(V)

```
 ans   =

    1.0e-14 *

    0.1110    0.1110    0.0888
```

```
    0.1332    0.4441    0.3553
    0.2665    0.4441    0.5329
```

Nous pouvons aussi obtenir les coefficients du polynôme caractéristique au moyen de la commande <poly>

poly(A)

```
 ans   =

   1.0000  -15.0000  -18.0000   -0.0000
```

Le polynôme caractéristique est donc égal à

$\det(\mu I - A) = \mu^3 - 15\,\mu^2 - 18\,\mu$

dont les racines sont

$\mu = 0$ et $\mu = (16.1168, \quad -1.1168)$

18 POLYNOMES

Il existe plusieurs fonctions MATLAB qui vous permettent de traiter les vecteurs comme si leurs entrées étaient des coefficients ou des racines d'équations polynomiales. Par exemple, tapez:

c = [1 1 1 1]

et puis

r = roots(c)

Les racines du polynôme $x^3+x^2+x+1 = 0$ devraient s'afficher à l'écran et être stockées dans le vecteur `r`. Les coefficients du polynôme peuvent être calculés à partir des racines avec la fonction `poly`

poly(r)

Un polynôme peut être évalué pour une valeur de `x` donnée. Par exemple,

polyval(c, 1.32)

Si un autre polynôme, $2x^2-0.4x-1$ est représenté par le vecteur d,

d = [2 -0.4 -1]

les deux polynômes peuvent être multipliés symboliquement avec la fonction de convolution, conv, pour obtenir les coefficients du produit:

cd = conv(c,d)

La fonction de déconvolution, deconv, peut être utilisée pour diviser un polynôme par un autre, par ex.

[q,r] = deconv(c,d)

Le résultat q est le quotient, et r le reste.

19 ANALYSE STATISTIQUE

Les calculs statistiques de base peuvent être effectués avec MATLAB. Pour des calculs plus complexes, il existe une librairie spécialisée : la Statistics Toolbox.

Les fonctions 'rand' et 'randn' vous permettent de générer une série de nombres (pseudo)aléatoires respectivement suivant une distribution uniforme ou normale :

num = randn(10000,1)

Remarquez l'importance du deuxième argument. Pour visualiser le signal que vous venez de créer, entrez

plot(num)

Vous pouvez vérifier que cette distribution est normale en calculant sa moyenne et sa déviation standard:

mean(num)

std(num)

Afin de visualiser l'histogramme, tapez

hist(num,20)

Ici, 20 correspond au nombre d'intervalles utilisés.

20 RESOLUTION DE SYSTEMES LINEAIRES

Une grande partie de la puissance de MATLAB réside dans les fonctions déjà construites dans le langage et relatives aux problèmes de l'algèbre linéaire. La résolution de systèmes n'ayant pas encore été abordée au cours théorique, cette section est présente à titre informatif.

En ce qui concerne la résolution de systèmes linéaires, nous avons déjà parlé des opérations / et \. Il existe encore d'autres fonctions dont nous donnons une description succincte ci-dessous

`lu` décomposition LU,

`det` déterminant,

`inv` inverse d'une matrice carrée,

`rank` rang d'une matrice

`cond` nombre de conditionnement en norme 2,

`rcond` estimation de l'inverse du nombre de conditionnement en norme 1,

Pour plus de détails, consulter le « help »

21 MATRICES CREUSES

Certaines matrices contiennent un grand nombre d'éléments nuls, on les appelle des matrices creuses (sparse matrix). MATLAB contient certaines fonctions pour traiter ces matrices particulières. Commençons par créer une matrice et un vecteur de taille relativement importante :

```
a = zeros(10000); a(50,50) = 1;

j = zeros(10000,1); j(50) = 1;
```

Nous allons calculer le temps nécessaire à la multiplication de ces deux éléments grâce aux commandes tic et toc :

```
tic, a*j ; toc
```

Utilisons maintenant la commande de MATLAB pour spécifier que notre matrice et notre vecteur sont creux, et répétons l'opération de multiplication :

```
b = sparse(a);

k = sparse(j);

tic, b*k; toc
```

Nous observons que la multiplication s'effectue bien plus rapidement (nous gagnons 3 ordres de grandeur). Nous pouvons également vérifier l'utilisation de la mémoire :

```
whos
```

De nouveau, nous voyons l'avantage d'utiliser la commande `sparse`. Libérons maintenant la mémoire utilisée :

```
clear
```

22 TRIER ET TROUVER

Deux opérations particulièrement utiles à effectuer sur des vecteurs sont les opérations de tri et de recherche. Dans MATLAB, elles se nomment respectivement sort et find. Examinons leur comportement.

```
a = rand(1,10)

sort(a)
```

```
[s i] = sort(a)
```

Que représente la valeur de i ?

```
a>0.5

j = find(a>0.5)

a(j)
```

Lors de l'utilisation de find, les opérateurs | et & sont également utiles. Ils signifient respectivement ou et et. Regardons comment ils fonctionnent :

```
a < 0.2 | a > 0.8

a(find(a < 0.2 | a > 0.8))

b = round(a*10)

b(find(b>5 & mod(b,2)))
```

Exercices

EX 22-1

– Nous avons un groupe de 30 personnes qui doit passer un interview. Déterminer un ordre de passage aléatoire pour ces 30 personnes.

– Soit le vecteur `x = [-3 9 3 4 -1 -4 7 7 3 10 -7 5 4 10 0 -2 -8 -5]`

Construisez un vecteur m contenant les 4 plus petits éléments positifs de `x` dans leur ordre d'apparition dans `x`. Déterminez également leur indices correspondant dans `x`.

– Créez une matrice aléatoire de taille `10x10`. Triez ensuite les valeurs propres de cette matrice par rapport à la valeur de leur angle (ou argument). Pour rappel, l'angle d'un nombre complexe a+ib est l'angle entre l'axe des ordonnées et le vecteur (a,b). Utilisez les commandes atan, real, et imag mais pas la commande angle (sauf pour vérifier vos résultats).

23 EQUATIONS DIFFERENTIELLES ET GESTION DE FICHIERS

Nous voulons résoudre l'équation différentielle suivante :

$$\frac{\partial y_1}{\partial t} = y_2 y_3$$
$$\frac{\partial y_2}{\partial t} = -y_1 y_3$$
$$\frac{\partial y_3}{\partial t} = -0.51\, y_1 y_2$$

Avec les conditions initiales $y_1(0) = 0$, $y_2(0) = 1$ et $y_3(0) = 1$.

Pour cela nous allons créer un fichier avec la fonction qui décrit l'équation différentielle (sans les conditions initiales). Si vous utilisez l'interface graphique de MATLAB, cliquez sur l'icône représentant une petite feuille blanche en haut à gauche de la fenêtre. Sinon, utilisez un programme tel que gedit, emacs, vi ou autre pour éditer le fichier. Le contenue à insérer dans ce fichier est le suivant :

```
function dy = rigid(t,y)

dy = zeros(3,1);

dy(1) = y(2) * y(3);

dy(2) = -y(1) * y(3);

dy(3) = -0.51 * y(1) * y(2);
```

Sauvez ce fichier dans le répertoire actuel de MATLAB sous le nom rigid.m (le répertoire actuel de MATLAB peut être obtenu en tapant la commande pwd dans MATLAB ; et il peut être changé en utilisant la commande cd). Quand vous créez un fichier avec une fonction, comme nous venons de le faire, veuillez à toujours appeler le fichier de la même manière que le nom de la fonction

auquel vous ajoutez '.m'.

Retournons maintenant sous MATLAB pour résoudre l'équation différentielle entre 0 et 12, puis traçons le résultat à l'écran :

```
[T,Y] = ode45(@rigid,[0 12],[0 1 1]);

plot(T,Y(:,1),'-',T,Y(:,2),'-.',T,Y(:,3),'.')
```

24 POUR VOUS EXERCER

– Ecrire une fonction recevant en argument n points differents x_1, ..., x_n. Cette fonction doit permettre de calculer la valeur du produit $(x - x_1)...(x - x_n)$ et dessiner le graphe de la fonction entre x_1 et x_n.

Remarque : Essayez d'écrire la fonction sans boucles et en utilisant les commandes prod et repmat.

– Soit le nuage de points suivant :

$(x_1, y_1) = (0.29, 4.98)$; $(x_2, y_2) = (1.87, 0.08)$; $(x_3, y_3) = (2.09, 2.19)$;

$(x_4, y_4) = (2.82, 0.74)$; $(x_5, y_5) = (6.08, 3.10)$; $(x_6, y_6) = (8.97, 3.77)$;

$(x_7, y_7) = (9.78, 1.40)$

Utilisez la fonction `polyfit` pour trouver le meilleur polynôme de degré 1, 2, ..., 5, 6 approximant ces points.

Utilisez les fonctions `plot` et `polyval` pour donner le graphe de ces polynômes. Utilisez la commande `subplot` pour afficher les 6 polynômes les uns à côté des autres. Utilisez la fonction spline pour déterminer la spline passant par ces points puis la représenter graphiquement.

– On vous demande de résoudre un système d'équations différentielles en utilisant MATLAB, puis d'afficher vos résultats de manière graphique.

Pour cela, vous devez créer un fichier MATLAB nomdevotrefichier.m vous permettant de résoudre l'équation différentielle grâce à la commande `ode45`.

On vous demande de définir le système d'équations différentielles au moyen d'une fonction qui sera définie dans un second fichier MATLAB appelé nomdevotrefonction.m.

Lorsque vous aurez terminé, la résolution s'effectuera en tapant nomdevotrefichier dans la fenêtre principale de MATLAB.

Pour obtenir des informations sur la manière d'utiliser `ode45`, tapez `help ode45` ou référez-vous à l'aide de MATLAB en cliquant sur 'Help',

puis sur 'MATLAB Help', puis en tapant 'ode' dans l'ongle 'Index'. La deuxième méthode permet d'obtenir une aide plus détaillée que la première.

De la même façon vous pouvez obtenir de l'aide au sujet des fonctions `function`, `plot`, `plot3`, ou de toute autre fonctions qui vous serait utile.

Voici le problème que vous devez résoudre :

On considère un système écologique simple constitué de lapins disposant d'une réserve de nourriture illimitée et de renards dont les lapins sont la seule nourriture.

Un modèle mathématique classique dû à Volterra décrit ce système par une paire d'équations différentielles ordinaires :

$$\frac{\partial L}{\partial t} = 2\,L - \alpha\,L\,R, \quad L(0) = L_0$$
$$\frac{\partial R}{\partial t} = -R + \alpha\,L\,R, \quad R(0) = R_0$$

où t est le temps, L = L(t) est le nombre de lapins, R = R(t) est le nombre de renards et α est une constante positive. Lorsque $\alpha = 0$, il n'y a pas d'interaction : les lapins croissent sans limite et les renards disparaissent. Lorsque $\alpha > 0$, les renards rencontrent les lapins avec une probabilité proportionnelle au produit de leur nombre. Il en résulte une diminution des lapins et une augmentation des renards.

1. Etudiez, dans le plan (L,R), les trajectoires pour = 0.01 pour les trois couples de conditions initiales suivantes

• $L_0 = 10$ et $R_0 = 10$

• $L_0 = 1$ et $R_0 = 10$

• $L_0 = 10$ et $R_0 = 1$

2. Dans chaque cas, mettez en évidence l'évolution temporelle en utilisant un affichage 3D.

25 COMMUNICATION AVEC LE MONDE EXTERIEUR

MATLAB est à même d'échanger (recevoir ou envoyer) des fichiers avec le monde extérieur.

Voici une description succincte des diverses possibilités

`save filename`	sauver dans le fichier *filename* toutes les variables de l'espace de travail
`load filename`	recharger dans l'espace de travail toutes les variables stockées dans le fichier *filename*
`print`	imprimer la figure courante
`diary filename`	faire une copie de la session MATLAB dans le fichier *filename*