Chapitre 1

Prise en main

1.1 Démarrage et aide

Sous Linux (ou Unix), le démarrage de Matlab se fait simplement en tapant `matlab` dans une fenêtre de commande (shell). Sous Windows ou MacOS, il suffit de double-cliquer sur l'icône Matlab. Dans tous les cas, on obtient une fenêtre avec une interface graphique qui permet de gérer les fichiers, les variables et les applications associées à MATLAB, figure 1.1 (pour la version 7) et figure 1.2 (pour la version 8). A noter que dans la version 8, l'éditeur de texte est intégré à l'espace de travail. Dans la fenêtre de commande (`Command Window`) apparaît le prompt `>>` caractéristique de MATLAB. C'est dans cette fenêtre qu'on exécute les commandes, les fonctions et les scripts MATLAB.

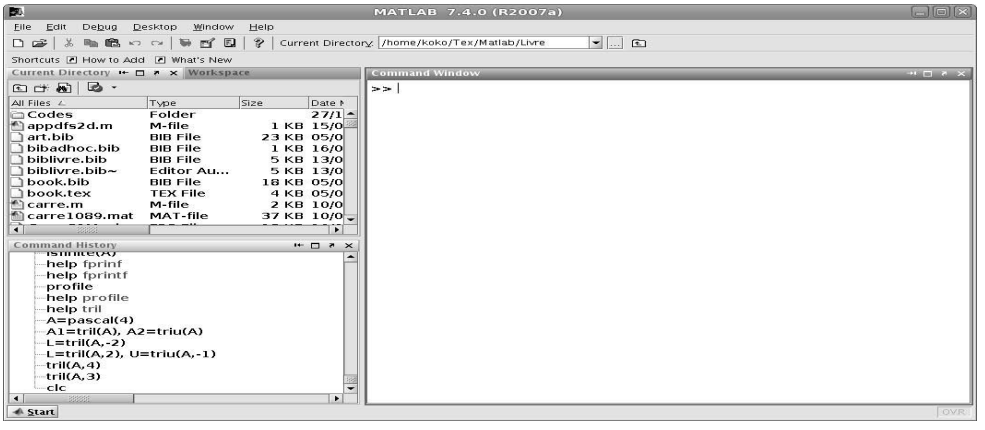

FIGURE 1.1 – Espace de travail MATLAB 7

L'aide en ligne s'obtient en tapant la commande `help`. Une longue liste de sujets, pour lesquels l'aide est disponible, apparaît alors:

```
>> help

HELP topics:

matlab/general       -  General purpose commands.
matlab/ops           -  Operators and special characters.
matlab/lang          -  Programming language constructs.
matlab/elmat         -  Elementary matrices, matrix manipulation.
matlab/elfun         -  Elementary math functions.
```

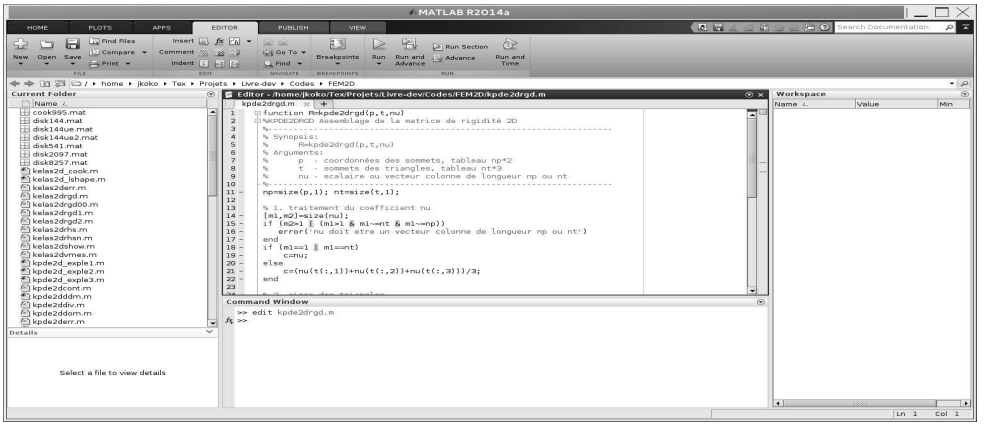

FIGURE 1.2 – Espace de travail MATLAB 8

```
matlab/specfun      -  Specialized math functions.
matlab/matfun       -  Matrix functions, numerical linear algebra.
matlab/datafun      -  Data analysis and Fourier transforms.
matlab/audio        -  Audio support.
matlab/polyfun      -  Interpolation and polynomials.
matlab/funfun       -  Function functions and ODE solvers.
matlab/sparfun      -  Sparse matrices.
matlab/graph2d      -  Two dimensional graphs.
matlab/graph3d      -  Three dimensional graphs.
matlab/specgraph    -  Specialized graphs.
matlab/graphics     -  Handle Graphics.
matlab/uitools      -  Graphical user interface tools.
matlab/strfun       -  Character strings.
matlab/iofun        -  File input/output.
matlab/timefun      -  Time and dates.
matlab/datatypes    -  Data types and structures.
matlab/verctrl      -  Version control.
matlab/demos        -  Examples and demonstrations.
toolbox/local       -  Preferences.
toolbox/optim       -  Optimization Toolbox.
toolbox/pde         -  Partial Differential Equation Toolbox.

For more help on directory/topic, type "help topic".
```

Après, il suffit d'exécuter

`>>help` *`sujet`*

pour avoir l'aide sur le sujet concerné. Pour avoir la description d'une fonction MATLAB, il suffit d'exécuter la commande `help` avec le nom de la fonction

`>>help` *`fonction`*

Par exemple, si on veut la description de la fonction `plot`, qui permet de tracer des courbes 2D, on tape simplement

```
>>help plot

 PLOT   Linear plot.
```

```
PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
then the vector is plotted versus the rows or columns of the matrix,
whichever line up.  If X is a scalar and Y is a vector, length(Y)
disconnected points are plotted.

PLOT(Y) plots the columns of Y versus their index.
If Y is complex, PLOT(Y) is equivalent to PLOT(real(Y),imag(Y)).
In all other uses of PLOT, the imaginary part is ignored.

Various line types, plot symbols and colors may be obtained with
PLOT(X,Y,S) where S is a character string made from one element
from any or all the following 3 columns:

      y     yellow      .     point              -     solid
      m     magenta     o     circle             :     dotted
      c     cyan        x     x-mark             -.    dashdot
      r     red         +     plus               --    dashed
      g     green       *     star
      b     blue        s     square
      w     white       d     diamond
      k     black       v     triangle (down)
                        ^     triangle (up)
                        <     triangle (left)
                        >     triangle (right)
                        p     pentagram
                        h     hexagram

For example, PLOT(X,Y,'c+:') plots a cyan dotted line with a plus
at each data point; PLOT(X,Y,'bd') plots blue diamond at each data
point but does not draw any line.

PLOT(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...) combines the plots defined by
the (X,Y,S) triples, where the X's and Y's are vectors or matrices
and the S's are strings.

For example, PLOT(X,Y,'y-',X,Y,'go') plots the data twice, with a
solid yellow line interpolating green circles at the data points.

The PLOT command, if no color is specified, makes automatic use of
the colors specified by the axes ColorOrder property.  The default
ColorOrder is listed in the table above for color systems where the
default is yellow for one line, and for multiple lines, to cycle
through the first six colors in the table.  For monochrome systems,
PLOT cycles over the axes LineStyleOrder property.

PLOT returns a column vector of handles to LINE objects, one
handle per line.

The X,Y pairs, or X,Y,S triples, can be followed by
parameter/value pairs to specify additional properties
of the lines.

See also SEMILOGX, SEMILOGY, LOGLOG, GRID, CLF, CLC, TITLE, XLABEL,
YLABEL, AXIS, AXES, HOLD, COLORDEF, LEGEND, SUBPLOT, and STEM.
```

Il est possible de retrouver toutes les fonctions contenant un mot clé. Pour cela on utilise la commande `lookfor`. Par exemple, si on recherche les fonctions relatives à la factorisation des matrices

```
>> lookfor factorization
CHOL   Cholesky factorization.
CHOLUPDATE Rank 1 update to Cholesky factorization.
LU     LU factorization.
QRDELETE Delete column from QR factorization.
QRINSERT Insert column in QR factorization.
QRUPDATE Rank 1 update to QR factorization.
QZ     QZ factorization for generalized eigenvalues.
CHOLINC  Sparse Incomplete Cholesky.
LUINC  Sparse Incomplete LU factorization.
SYMBFACT Symbolic factorization analysis.
```

Remarque 1.1 *Si dans l'aide en ligne, les fonctions et commandes MATLAB apparaissent en lettres majuscules c'est seulement pour les distinguer des autres mots. Les fonctions et commandes Matlab doivent être codées en minuscules.*

Les commandes d'éditions de Matlab sont simples:

$\uparrow$ remonter dans l'historique de commandes;

$\downarrow$ descendre dans l'historique de commandes;

$\leftarrow$, $\rightarrow$ déplacement sur la ligne;

Backspace, Delete modifications sur la ligne de commande.

1.2 Calculatrice

En mode interactif, MATLAB peut être utilisé comme une calculatrice scientifique disposant d'un très grand nombre de fonctions de calcul numérique et de visualisation graphique. Une ligne de commande de MATLAB est de la forme [1]

```
{variable=}expression{;}
```

Lorsque le point-virgule (`;`) est absent en fin de ligne, la valeur de l'expression est affichée à l'écran. Les sorties MATLAB sont donc en grande partie gérées par le point-virgule. Lorsqu'il n'y a pas de variable pour récupérer la valeur de l'expression, c'est la variable par défaut de MATLAB, `ans`, qui est utilisée. Voici quelques exemples de ligne de commande

```
>> 2*pi/9
ans =
    0.6981

>> x=6^10
x =
    60466176
```

Par défaut les réels sont affichés [2] en simple précision (5 chiffres significatifs). On peut modifier l'affichage avec la commande `format`:

```
>> pi
ans =
    3.1416
```

1. En ligne de comande, tout ce qui apparaît entre `{}` est optionnel.
2. Tous les calculs dans MATLAB sont en double précision

```
>> format long
>> pi
ans =
   3.14159265358979

>> format short
>> pi
ans =
    3.1416
```

Pour obtenir un affichage de réels en virgule flottante, il suffit de rajouter `e` dans à la commande `format`

```
>> format short e
>> pi^6
ans =
   9.6139e+02
```

Enfin, `format short g` ou `format long g` laisse le choix à MATLAB d'utiliser le «meilleur» affichage (virgule fixe ou virgule flottante) en fonction du réel. D'autres formes d'affichages existent (`format bank`, `format rat`, `format hex`, `format compact`), voir l'aide en ligne.

Les nombres complexes se codent de façon naturelle avec `i` ou `j` comme imaginaire pure

```
>> 1.5+i
ans =
   1.5000 + 1.0000i
```

Le passage en complexes est automatique et sans avertissement.

```
>> c=sqrt(-1)
c =
   0 + 1.0000i

>> pi+acos(pi)
ans =
   3.1416 + 1.8115i
```

Ce qui peut être fâcheux en cas d'erreur de programmation. Les opérateurs pour les complexes sont les mêmes que pour les nombres réels:

```
>> c1=exp(i)
c1 =
   0.5403 + 0.8415i

>> c2=sqrt(-2)
c2 =
    0 + 1.4142i

>> c1+c2                    % somme de deux complexes
ans =
   0.5403 + 2.2557i

>> c1*c2                    % produit de deux complexes
ans =
    -1.1900 + 0.7641i
```

```
>> c1/c2                    % quotient de deux complexes
ans =
    0.5950 - 0.3821i

>> c1^(1/2)                 % racine carrée d'un complexe
ans =
   0.8776 + 0.4794i
```

1.3 Ponctuation, commentaires, interruption

Une ligne de commande MATLAB peut comporter plusieurs commandes séparées par des virgules (`,`) ou des points-virgules (`;`)

```
>> x=3/8, y=2.2678/37.80; z=(x+y)/sqrt(-4)
x =
   0.3750
z =
   0 - 0.2175i
```

Une ligne de commande MATLAB peut comporter des commentaires signalés par le symbole `%`. Tout ce qui suit ce symbole est alors ignoré par l'interpréteur MATLAB:

```
>> x=pi        % nombre réel
x =
    3.1416

>> z=sqrt(-1)  % nombre complexe
z =
        0 + 1.0000i
```

Si une expression est trop longue pour tenir sur une ligne de commande, alors on termine la ligne par `...` (3 points):

```
>> y=cos(2*pi/3)*(sin(5.78*pi/4)^2+...
     cos(pi/5)^2)
y =
   -0.8125
```

Un traitement MATLAB peut être à tout moment interrompu en appuyant simultanément sur les touches **Ctrl** et **C**.

1.4 Variables

Les noms des variables MATLAB sont des mots de 63 caractères (alphanumériques) maximum, sans espace et sans symbole de ponctuation. Le nom d'une variable doit obligatoirement commencer par une lettre de l'alphabet. Il y a une distinction entre majuscules et minuscules pour les noms de variables:

```
>> x1=3.8675, X1=pi^2
x1 =
    3.8675
X1 =
    9.8696
```

Comme tous les langages, MATLAB dispose de mots clés. Certains mots clés ne peuvent être utilisés comme variables. Ce sont les mots réservés du langage. La liste des mots réservés de MATLAB est donnée dans le tableau 1.1. Une tentative d'utilisation provoque une erreur

```
>> for=2.16
??? for=2.16
       |
Error: "identifier" expected, "=" found.
```

La liste de ces mots réservés est fournie par la fonction `iskeyword`. D'autres mots clés, non réservés, représentent des variables ou des constantes prédéfinies, voir tableau 1.2. Ces mots-clés peuvent être utilisés comme variables définies par l'utilisateur:

```
>> pi, realmax
ans =
     3.1416
ans =
     1.7977e+308

>> pi=278, realmax=.2568
pi =
    278
realmax =
         0.2568
```

En particulier les imaginaires purs `i` et `j`, vues au § 1.2, peuvent être utilisés comme indices dans les boucles.

`break case catch continue else elseif end for function`
`global if otherwise persistent return switch try while`

TABLE 1.1 – Listes de mots réservés

`ans`	*answer* variable MATLAB par défaut
`eps`	précision numérique relative, `eps=2.220446049250313e-016`
`inf`	l'infini mathématique ∞, *e.g.* `1.0/0.`
`nan`	littéralement *Not a Number*, *e.g.* `0/0`
`pi`	constante $\pi = 3.14159265358979$
`i ou j`	nombre complexe $\sqrt{-1}$
`nargin`	nombre d'arguments d'entrée d'une fonction
`nargout`	nombre d'arguments de sortie d'une fonction
`realmin`	plus petit réel utilisable
`realmax`	plus grand réel utilisable
`bitmax`	plus grand entier utilisable, stocké en double précision
`varargin`	nombre variable d'arguments d'entrée d'une fonction
`varargout`	nombre variable d'arguments de sortie d'une fonction

TABLE 1.2 – Constantes et variables prédéfinies

1.5 Gestion de la mémoire

La commande `who` donne la liste des variables actives dans l'espace de travail. Mais elle est moins précise. Pour avoir plus d'informations, il est préférable d'utiliser la commande `whos` qui

donne non seulement les variables mais aussi leur type, leur taille (comme tableau) et l'espace mémoire occupé. Voici un exemple

```
>> whos
  Name      Size          Bytes  Class

  A         50x50         20000  double array
  x          1x10001      80008  double array
  y          1x10001      80008  double array
  zz       100x100        80000  double array
```

Il est possible de supprimer une ou plusieurs variables avec la commande `clear`. Par exemple

```
>> clear x zz
```

supprime les variables `x` et `zz` de l'espace de travail. En particulier `clear all` efface toutes les variables de l'espace de travail.

1.6 Répertoire de travail

Avant de commencer à travailler, il faut indiquer à Matlab le répertoire dans lequel on veut travailler. Sous Unix, le répertoire de travail est le répertoire courant. Sous Windows, le répertoire par défaut s'appelle `works` et se trouve dans le répertoire d'installation de Matlab. Pour le changer il suffit d'entrer le nom de votre répertoire de travail dans la fenêtre `Current Directory` de la barre de menu.

1.7 Sauvegarde de l'environnement de travail

La fonction `save` permet de sauvegarder tout (par défaut) ou partie de l'espace de travail, qui se récupère grâce à la commande `load`. Par exemple si on veut sauvegarder les variables `X` et `Y` dans le fichier `FichXY` il suffit d'exécuter la ligne de commande

```
>> save FichXY X Y
```

Par défaut le fichier de sauvegarde est en binaire et comporte l'extension `.mat`.

La fonction `load` est l'inverse de la fonction `save`. Par exemple, pour récupérer les variables `X` et `Y` sauvegardées dans le fichier `FichXY`, il suffit de faire

```
>> load FichXY X Y
```

ou

```
>> load FichXY
```

pour charger toutes les variables sauvegardées dans le fichier `FichXY`.

La commande `diary` permet de sauvegarder l'intégralité d'une session de travail. Elle est très utile lorsqu'on tâtonne. On démarre la sauvegarde avec `diary on` et on arrête avec `diary off`. Dans tous les cas lorsqu'on tape `diary`, on passe d'un état à un autre, *i.e.* on passe de `diary on` à `diary off` ou *vice versa*. Il est possible d'utiliser la forme fonctionnelle suivante

```
>> diary('nomfich')
```

Alors toute la session sera sauvegardée dans le fichier `nomfich`.