# MATLAB

Michaël Thémans

20 mars 2003

## 1 Brève présentation

MATLAB est un langage de programmation pour lequel une interface spéciale a été conçue. Il s'agit en gros d'une *calculatrice programmable interactive* qui permet de manipuler des vecteurs, des matrices, des fonctions, ... et de dessiner ou représenter ces objets ou les concepts qu'ils permettent de modéliser.

Il s'agit d'un **langage interprété**, c'est-à-dire que les programmes sont "compilés" et exécutés au cours d'une seule et même opération.

L'avantage indéniable de MATLAB réside en une syntaxe assez simple et intuitive. Celle-ci a d'ailleurs été reprise dans le livre de Golub & Van Loan, intitulé *Matrix computations*, pour présenter les algorithmes.

## 2 Documentation

### 2.1 Sites web

On pourra consulter le site officiel qui permet de télécharger quelques petits manuels de référence ou de petits programmes disponibles gratuitement :

`http ://www.mathworks.com/products/matlab`

Pour les amateurs de la langue de Molière, une version française est également disponible à l'adresse suivante :

`http ://www.fr.mathworks.com/products/matlab`

### 2.2 Aide MATLAB

Pour obtenir de l'aide sur une instruction ou sur une fonction, il suffit de taper `help` suivi du nom de l'instruction ou de la fonction souhaitée dans l'invite MATLAB. On peut également utiliser l'aide fournie par l'interface graphique de MATLAB.

## 3 Interface et mode de fonctionnement

Pour démarrer MATLAB, on tape `matlab` dans une fenêtre `xterm`. On arrive alors dans l'interface graphique de MATLAB. On y trouve trois cadres qui ont

chacun un rôle précis :

- La fenêtre **Launch pad** permet d'avoir accès aux outils, aux exemples et à la documentation des produits MATLAB dont on dispose.
- La fenêtre **Command history** contient l'historique des sessions les plus récentes. On peut exécuter à nouveau les commandes qui ont été tapées en double-cliquant sur leur nom. On peut également utiliser le copier-coller afin de recopier une commande dans un fichier.
- La fenêtre **Command window** permet de taper des commandes MATLAB et de travailler en mode interactif.

Une autre façon d'utiliser le logiciel MATLAB consiste à créer des fichiers contenant des programmes écrits en MATLAB comme on le fait habituellement avec les autres langages de programmation. Le nom de ces fichiers doit obligatoirement se terminer par le suffixe `.m`, par exemple `optimisation.m`. Si on tape dans la fenêtre **Command window** `optimisation`, les instructions contenues dans le fichier `optimisation.m` seront exécutées une à une. Cette façon de travailler est évidemment recommandée si l'on tape de nombreuses instructions ! Ainsi, on pourra sauvegarder son travail dans un ou plusieurs fichiers.

On peut écrire un programme MATLAB avec n'importe quel éditeur de texte (`emacs` par exemple) ou en l'éditant â l'aide de l'interface graphique fournie par le logiciel (aller dans `File` → `New` → `M-File`).

Il est également possible de créer des fonctions qui peuvent être utilisées dans les expressions mathématiques ou dans les instructions MATLAB. Une fonction MATLAB est un fichier `.m` particulier dont la première ligne commence par « function ». Afin de pouvoir utiliser une fonction, il faut que le nom du fichier `.m` corresponde exactement au nom de la fonction et que ce fichier se trouve dans le répertoire de travail !

# 4 Sélection de commandes et fonctions MATLAB

## 4.1 Commandes de base

### 4.1.1 Instructions de contrôle

- L'instruction conditionnelle `if` :

```
if {expression logique}
   {instruction(s)}
elseif {expression logique}
   {instruction(s)}
        else
   {instruction(s)}
        end
```

- L'instruction `switch` :

```
switch {expression}
  case {valeur#1}
 {instruction(s)#1}
  case {valeur#2}
 {instruction(s)#2}
```

```
.
.
.
otherwise
{instruction(s)}
end
```

Si `expression` a la `valeur` $i$, le programme exécute la ou les `instruction(s)` $\#i$. Si `expression` ne prend aucune des valeurs spécifiées derrière un `case`, la ou les `instruction(s)` sous `otherwise` est(sont) exécutée(s).

- Boucle `while`

```
while {expression logique}
{instruction(s)}
end
```

- Boucle `for`

```
for {indice} = {debut} : {increment} : {fin}
{instruction(s)}
end
```

- L'instruction `continue` passe la main à la prochaine itération d'une boucle `for` ou `while`, en ignorant les instructions restantes dans le corps de la boucle pour l'itération en cours.
- L'instruction `break` force la fin d'une boucle `for` ou `while` en court-circuitant les instructions restantes dans la boucle et en passant à la première instruction exécutable trouvée après la boucle.
- L'instruction `return` termine la séquence de commandes en cours et rend la main à la fonction appelante ou à l'écran/clavier.
- L'instruction `pause` permet de stopper l'exécution d'un programme en attendant que l'utilisateur appuie sur une touche du clavier.
  Variante : `pause(n)` permet de stopper l'exécution d'un programme pendant `n` secondes.

### 4.1.2 Entrées/sorties

- `input` permet d'afficher un message pour poser une question à l'utilisateur et de stocker sa réponse dans une variable, par exemple

```
n = input( 'Donner un entier : ' )
```

- Impression de messages à l'écran : `disp('message')`

### 4.1.3 Affichage d'instructions

Lors de la frappe d'instructions dans un fichier ou à l'écran, le mode de fonctionnement par défaut affiche le résultat de chaque ligne de commande, ce qui peut parfois donner lieu à un affichage trop long et difficile à interpréter. Cela peut également ralentir considérablement l'exécution du programme en cours. Si on ne veut pas que MATLAB affiche le résultat d'une ligne/instruction, il suffit de rajouter un ; à la fin de la ligne concernée.

## 4.2 Opérations de base sur les matrices et les vecteurs

Considérons les objets mathématiques suivants :

$$A = \begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix} \in \mathbb{R}^{3\times 3},$$

$$I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{3\times 3},$$

$$b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \in \mathbb{R}^{3\times 1},$$

$$c = (c_1\ c_2\ c_3) \in \mathbb{R}^{1\times 3}.$$

Le tableau ci-dessous reprend les principales commandes et fonctions de base permettant de manipuler les vecteurs et les matrices de MATLAB :

| Opération réalisée | Expression correspondante en MATLAB |
|---|---|
| Création de $A$ | $A = [\ a_1\ a_2\ a_3\ ;\ a_4\ a_5\ a_6\ ;\ a_7\ a_8\ a_9\ ]$ |
| Création de $I$ | $I = eye(3)$ |
| Création de $b$ | $b = [\ b_1\ ;\ b_2\ ;\ b_3\ ]$ |
| Création de $c$ | $c = [\ c_1\ \ c_2\ \ c_3\ ]$ |
| $rg(A)$ | $rank(A)$ |
| $det(A)$ | $det(A)$ |
| Valeurs propres de $A$ | $eig(A)$ |
| $A^{-1}$ | $inv(A)$ |
| $A^T$ | $A'$ |
| $b^T$ | $b'$ |
| $tr(A)$ | $trace(A)$ |
| $M := A + I \in \mathbb{R}^{3\times 3}$ | $M = A + I$ |
| $M := A - I \in \mathbb{R}^{3\times 3}$ | $M = A - I$ |
| $v := b + c^T \in \mathbb{R}^{3\times 1}$ | $v = b + c'$ |
| $w := b^T + c \in \mathbb{R}^{1\times 3}$ | $w = b' + c$ |
| $\alpha = cb \in \mathbb{R}$ | $alpha = c * b$ |
| $v := Ab \in \mathbb{R}^{3\times 1}$ | $v = A * b$ |
| $w := cA \in \mathbb{R}^{1\times 3}$ | $w = c * A$ |
| $M := bc \in \mathbb{R}^{3\times 3}$ | $M = b * c$ |
| $M := AI \in \mathbb{R}^{3\times 3}$ | $M = A * I$ |
| $\|A_1\|$ | $norm(A,1)$ |
| $\|A_2\|$ | $norm(A,2)$ |
| $\|A_\infty\|$ | $norm(A,inf)$ |
| $\|b_1\|$ | $norm(b,1)$ |
| $\|b_2\|$ | $norm(b,2)$ |
| $\|b_\infty\|$ | $norm(b,inf)$ |

Les opérations « élément par élément » sur les vecteurs et les matrices sont effectués en ajoutant . avant les opérateurs, par exemple :

```
A.*I
```

### 4.3 Utilisation des indices

Les éléments d'un vecteur ou d'une matrice peuvent être adressés en utilisant les indices sous la forme suivante :

| | |
|---|---|
| $t(10)$ | $10^{\text{e}}$ élément du vecteur $t$ |
| $t(1:5)$ | 5 premiers éléments du vecteur $t$ |
| $A(2,9)$ | élément se trouvant à la $2^{\text{e}}$ ligne et à la $9^{\text{e}}$ colonne 9 de la matrice $A$ |
| $B(:,7)$ | $7^{\text{e}}$ colonne de la matrice $B$ |
| $C(3,:)$ | $3^{\text{e}}$ ligne de la matrice $C$ |

### 4.4 Systèmes linéaires généraux

| Fonction | Emploi | Opération réalisée |
|---|---|---|
| $\backslash$ | $x = A\backslash B$ | résout les systèmes d'équations $AX = B$ (où $A$ est une matrice carrée) |
| | $X = A\backslash B$ | idem mais avec plusieurs membres de droite regroupés dans $B$ |
| *lu* | $[L,U] = lu(A)$ | calcule la factorisation $PA = LU$ (avec pivotage partiel) |
| | $[L,U,P] = lu(A)$ | idem avec la matrice de permutations $P$ donnée explicitement en sortie |
| *cond* | $cond(A)$ | calcule $\kappa(A)$ |
| *cond* | $cond(A,1)$ | calcule $\kappa_1(A)$ |
| *cond* | $cond(A,inf)$ | calcule $\kappa_\infty(A)$ |
| *cond* | $cond(A,fro)$ | calcule $\kappa(A)$ pour la norme de Frobenius |

### 4.5 Systèmes linéaires particuliers

| Fonction | Emploi | Opération réalisée |
|---|---|---|
| *chol* | $G = chol(A)$ | calcule la factorisation de Cholesky $A = GG'$ (où $A$ est $n$ x $n$ symétrique définie positive) |
| *tril* | $tril(A)$ | donne la partie triangulaire inférieure de $A$ |
| | $tril(A,k)$ | donne les éléments sur et sous la $k$-ième diagonale de $A$ |
| *triu* | $triu(A)$ | donne la partie triangulaire supérieure de $A$ |
| | $triu(A,k)$ | donne les éléments sur et au dessus de la $k$-ième diagonale de $A$ |

### 4.6 Orthogonalisation et moindres carrés

| Fonction | Emploi | Opération réalisée |
|---|---|---|
| *qr* | $[Q,R] = qr(A)$ | calcule la factorisation $A = QR$ (où $A$ est rectangulaire $m\ x\ n$ avec $m > n$)) |
| | $[Q,R] = qr(A,0)$ | idem mais en faisant l'économie des $m\ -\ n$ dernières colonnes de $Q$ |
| | $[Q,R,P] = qr(A)$ | idem mais avec permutations (regroupées dans la matrice $P$) |
| | $[Q,R,p] = qr(A,0)$ | combinaison des deux méthodes précédentes (avec un vecteur $p$ de permutations par économie de place) |
| *svd* | $[U,S,V] = svd(A)$ | calcule la décomposition en valeurs singulières $A\ =\ U\Sigma V^T (m\ x\ n)$ |
| | $[U,S,V] = svd(A,0)$ | idem avec économie d'une ligne entièrement nulle dans $\Sigma$ |
| *orth* | $Q = orth(A)$ | construit une base orthonormale de $Im(A)$ |