Initiation MATLAB – Séance 1

–

Manipulation de matrices, structures de contrôle et affichage 2D

Guillaume Revy / Chu-Duc Nguyen

03/12/2008

1 Introduction à MATLAB

MATLAB est un système interactif pour le calcul numérique (langage de programmation, environnement de travail et de résolution de problèmes), qui propose des outils performants pour la manipulation de matrices, l'édition de graphiques, le debuggage, le profiling, ...

1.1 Interface graphique générale

L'interface graphique générale de MATLAB se compose de quatre blocs.

Command line (console d'exécution) : permet à l'utilisateur d'exécuter des fonctions MATLAB, d'attribuer des valeurs à des variables, d'effectuer des opérations sur ces variables...

Current directory (répertoire courant) : permet de naviguer et de visualiser le contenu du répertoire courant de l'utilisateur.

Workspace (espace de travail) : permet de visualiser, notamment, les variables, leur type et leur état...

Command history : historique des commandes que l'utilisateur a exécutées.

1.2 Caractéristiques de MATLAB

→ MATLAB est *case sensitive* : différence entre les majuscules et les minuscules.
→ Les variables n'ont pas à être déclarées, et la mémoire nécessaire est automatiquement allouée à la définition d'une variable.
→ Par défaut, les valeurs numériques sont des flottants double précision. Elles peuvent cependant être converties en simple précision.
→ En MATLAB, le symbole d'affectation est le signe =

```
>> a = 8
a =
     8
>> sqrt(4)
ans =
     2
>> A = 9;
```

→ Si l'utilisateur n'affecte pas explicitement le résultat d'une opération (par exemple) à une variable, MATLAB l'affecte automatiquement à la variable `ans`.
→ Le symbole ; (point virgule) en fin de ligne annule l'affichage du résultat (mais pas des messages d'erreur).
→ Pour obtenir de l'aide sur une commande, il suffit de taper `help command` dans la console d'exécution, ou de lancer l'aide de MATLAB (*Help* > *MATLAB Help* ou *F1*).

```
>> help sqrt
 SQRT   Square root.
    SQRT(X) is the square root of the elements of X. Complex
    results are produced if X is not positive.
```

→ Pour rechercher une fonction relative à un mot clé, il suffit de taper `lookfor keyword`.

```
>> lookfor identity
EYE Identity matrix.
SPEYE  Sparse identity matrix.
SETDATID Sets tha estimation data identity for a model
```

→ En MATLAB, le symbole % est le symbole de commentaire.

2 Prise en main de MATLAB

2.1 Initialisation et opérations sur les matrices

L'initialisation d'une matrice peut se faire de plusieurs manières : affectation d'une liste d'éléments, résultats d'opérations ou de fonctions, à partir d'autres matrices, à partir de données externes, ...

```
>> A = [1 2 3 ; 2 3 4 ; 4 5 6]
A =
     1     2     3
     2     3     4
     4     5     6
>> B = eye(3) % matrice identité de taille 3*3
B =
     1     0     0
     0     1     0
     0     0     1
>> C = 1:0.5:3
C =
    1.0000    1.5000    2.0000    2.5000    3.0000
```

La matrice $\mathbf{A}$ est définie ligne par ligne (en commençant par la première). Chaque ligne est séparée par un point virgule, et chaque élément d'une ligne par un espace ou une virgule.

Les opérations arithmétiques $+$, $-$, $\times$, $/$, et `^` peuvent être effectuées sur les matrices (si leurs dimensions le permettent).

```
>> B * A
ans =
     1     2     3
     2     3     4
     4     5     6
>> A .* B
ans =
     1     0     0
     0     3     0
     0     0     6
```

Si un symbole d'opération arithmétique est précédé d'un point (.), l'opération est effectuée élément par élément sur les matrices.
L'opération $\mathbf{B}/\mathbf{A}$ est équivalente à $\mathbf{B} \times$ `inv(A)` (avec `inv(A)` : inverse de la matrice carrée $\mathbf{A}$).
Autrement, l'opération $\backslash$ peut également être utilisée : $\mathbf{A}\backslash\mathbf{B}$ est équivalent à `inv(A)` $\times \mathbf{B}$.

2.2 Accès aux éléments d'une matrice

L'accès aux éléments d'une matrice $\mathbf{A}$ se fait de la manière suivante :
→ `A(i)` : accès au premier élément de la i^e ligne de la matrice $\mathbf{A}$
→ `A(i,j)` : accès au j^e élément de la i^e ligne de la matrice $\mathbf{A}$ ($i \geq 1$ et $j \geq 1$)

```
>> A(3)
ans =
     4
>> A(2,2)
ans =
     3
>> A(4,4)=5
A =
     1     2     3     0
     2     3     4     0
     4     5     6     0
     0     0     0     5
```

Lors de l'affectation d'une nouvelle valeur à un élément d'une matrice, MATLAB pourra (si nécessaire) étendre la matrice pour que l'affectation ait du sens (les éléments ajoutés seront initialisés à 0).

L'accès à un ensemble d'éléments de la matrice **A** se fait de la manière suivante :
→ `A(i1 :i2)` : accès aux éléments i_1 à i_2 de la première colonne de **A** (résultat sous la forme d'un vecteur ligne)
→ `A(i1 :i2,j)` : accès aux éléments i_1 à i_2 de la j^e colonne de **A**
→ `A( :,j)` : accès à l'ensemble des éléments de la j^e colonne de **A**
→ `A(i,j1 :j2)` : accès aux éléments j_1 à j_2 de la i^e ligne de **A**
→ `A(i, :)` : accès à l'ensemble des éléments de la i^e ligne de **A**
→ `A(i1 :i2,j1 :j2)` : accès aux éléments i_1 à i_2 des colonnes j_1 à j_2 de **A**

```
>> A(1:4)
ans =
     1     2     4     0
>> A(2:4,1)
ans =
     2
     4
     0
>> A(:,1)
ans =
     1
     2
     4
     0
>> A(4,1:4)
ans =
     0     0     0     5
>> A(2:4,1:4)
ans =
     2     3     4     0
     4     5     6     0
     0     0     0     5
```

2.3 Variables définies dans l'espace de travail

La commande **who** permet d'afficher le nom de l'ensemble des variables définies dans l'espace de travail, et la commande **whos** permet, en plus, d'afficher la place mémoire qu'elles occupent.

```
>> whos
  Name        Size                    Bytes  Class

  A           3x3                        72  double array
  B           3x3                        72  double array

Grand total is 18 elements using 144 bytes
```

Il est à noter que, par défaut, les éléments créés par MATLAB sont de type double précision, `double` (pour plus de détails sur la norme IEEE-754, lire [2]). La matrice **A** peut cependant être convertie en simple précision par la commande **single**.

```
>> AA = single(A);
>> C = AA * B;
>> whos
  Name        Size                    Bytes  Class

  A           3x3                        72  double array
  AA          3x3                        36  single array
  B           3x3                        72  double array
  C           3x3                        36  single array

Grand total is 36 elements using 216 bytes
```

Le résultat d'un calcul faisant intervenir un élément simple précision est automatiquement converti en simple précision.

La commande `clear` efface l'ensemble des variables de l'espace de travail.

2.4 Vecteurs et polynômes

Un vecteur est défini de la même manière qu'une matrice. L'exemple suivant illustre la définition d'un vecteur ligne **v** et d'un vecteur colonne **u**.

```
>> v = [0,1,2,3]
v =
     0     1     2     3
```

```
>> u = [1;2;3;4]
u =
     1
     2
     3
     4
```

Les polynômes, quant à eux, sont définis comme des vecteurs ligne. Soit a un polynôme de degré $n : a(x) = \sum_{i=0}^{n} a_i x^i$. Ce polynôme est représenté par le vecteur ligne $[a_n \ a_{n-1} \ a_{n-2} \cdots \ a_1 \ a_0]$. L'exemple suivant illustre la définition, en MATLAB, du polynôme $a(x) = 4x^3 - 3x^2 + 2x - 1$.

```
>> a = [4 -3 2 -1]
a =
     4    -3     2    -1
```

3 Structures de contrôle

3.1 Construction `if-{elseif-else}-end`

```
if condition1
   command1
elseif condition2
   command2
else
   command3
end
```

3.2 Boucles `for` et `while`

```
for i=1:20
  command
end
```

```
for i=1:0.5:20
  command
end
```

```
for i=20:-0.5:1
  command
end
```

```
while condition
  command
end
```

Les boucles `for` et `while` peuvent être interrompues par l'instruction `break`.

3.3 Construction `swicth-case`

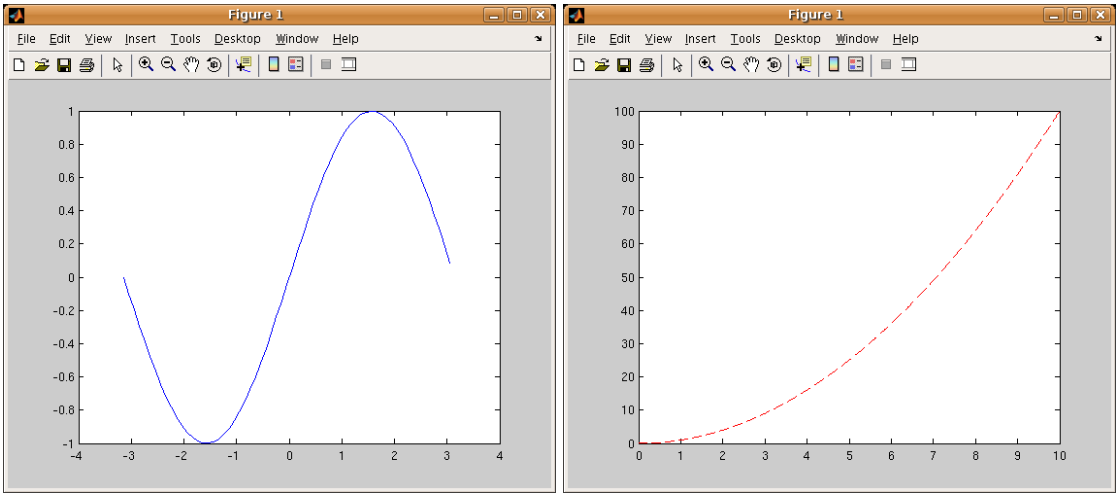
```
switch x
  case value1
    command1
  case {value2,value3}
    command23
  otherwise
    default_command
end
```

4 Affichage graphique 2D de base

L'affichage graphique 2D de base se fait par la commande `plot(x,y,`*options*`)`. Cette commande permet alors de tracer la courbe passant par les points (`x(1),y(1)`), (`x(2),y(2)`)... Le premier exemple montre l'affichage de la fonction $\sin(x)$ sur l'intervalle $[-\pi, \pi]$.

```
>> x = [-pi:0.1:pi];
>> y = sin(x);
>> plot(x,y);
```

Le résultat obtenu est illustré par la figure (a). La figure (b) est le résultat du deuxième exemple,



(a) (b)

qui calcule x^2 pour 1001 valeurs de $x \in [0, 10]$ par pas de 0.01.

```
>> idx = 1;
>> for i=0:0.01:10
     x(idx) = i;
     y(idx) = i*i;
     idx = idx + 1;
   end;
>> plot(x,y,'r--');
```

Pour plus de détails sur la commande `plot` : `help plot`.

5 Scripts et fonctions

Les scripts ou les fonctions sont stockés dans des fichiers à l'extension `.m` (*test.m*, par exemple), et contiennent un ensemble de commandes MATLAB.

Script Un fichier *script* n'a ni paramètre d'entrée ni paramètre de sortie, et travaille sur les variables de l'espace de travail.

Fonction Un fichier *fonction* contient la définition de la fonction, et peut accepter des paramètres d'entrée et de sortie. Les variables internes à la fonction sont locales (à moins qu'elles ne soient déclarées `global`).

L'éditeur de scripts/fonctions de MATLAB peut être lancé avec la commande `edit`. L'exemple suivant présente une fonction qui prend en paramètre d'entrée deux nombres a et b et retourne la somme $s = a + b$ et le produit $p = a \times b$.

```
function [s,p]=somme_produit(a,b)
  s = a + b;
  p = a * b;
end
```

Cette fonction doit être stockée dans le fichier `somme_produit.m`. L'exécution de cette fonction donne le résultat suivant :

```
>> [s,p]=somme_produit(4,5)
s =
     9
p =
    20
```

Pour exéctuer un script ou une fonction, le fichier doit se trouver dans le répertoire courant ou dans un répertoire de la variable `path` (voir `path`, `addpath` et `rmpath`). Pour exécuter un script, il suffit alors de taper dans la console d'exécution le nom du fichier script (sans l'extension `.m`), et pour une fonction, le nom de la fonction suivi des paramètres d'entrée entre parenthèses (si l'utilisateur veut récupérer la sortie, il doit indiquer les noms des variables de stockage, sinon, le dernier paramètre de sortie sera stocké dans la variable `ans`).

6 Exercices d'application

6.1 Polynôme caractéristique et matrice compagnon

Soit $\mathbf{A}$ une matrice carrée de taille $n \times n$. On note λ_i les valeurs propres de $\mathbf{A}$ et $a(x)$ son polynôme caractéristique : $a(x) = \prod_{i=1}^{n}(x - \lambda_i)$.

1. Créer manuellement ou générer aléatoirement une matrice de taille $m \times n$ (m, n donnés).
2. Écrire la fonction qui détermine le polynôme caractéristique d'une matrice $\mathbf{A}$ donnée en paramètre (afficher un message d'erreur dans le cas où la matrice $\mathbf{A}$ n'est pas carrée). La fonction devra également retourner le déterminant et la trace de la matrice $\mathbf{A}$.
3. Comparer les résultats de la question 2. avec ceux des fonctions `poly`, `det` et `trace`.

Le polynôme caractéristique de la matrice est de la forme : $a(x) = a_n x^n + \sum_{i=0}^{n-1} a_i x^i$. La matrice compagnon de a peut alors être définie de la manière suivante :

$$\begin{pmatrix} 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & \cdots & 0 & -a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -a_{n-1} \end{pmatrix}$$

4. Construire bloc par bloc la matrice compagnon $\mathbf{C}$ de a précédemment obtenu.
5. Calculer les valeurs propres de $\mathbf{C}$, et vérifier qu'elles correspondent aux racines de $a(x)$ (et donc aux valeurs propres de $\mathbf{A}$).
6. Vérifier le théorème d'Hamilton-Caley : $a(A) = A^n + a_{n-1}A^{n-1} + \cdots + a_1 A + a_0 I_n = 0_n$.

6.2 Matrices de translation et de rotation

Soit a un point du plan : $a = (x, y)$. On note (x_t, y_t) les coordonnées de a après translation d'un vecteur $\vec{u} = (u_1, u_2)$, et (x_r, y_r) les coordonnées de a après rotation de centre O (de coordonnées $(0, 0)$) et d'angle θ. Les coordonnées (x_t, y_t) et (x_r, y_r) peuvent être calculées de la manière suivante :

$$\begin{pmatrix} 1 & 0 & u_1 \\ 0 & 1 & u_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x_t \\ y_t \\ 1 \end{pmatrix} \qquad \text{et} \qquad \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x_r \\ y_r \\ 1 \end{pmatrix}$$

1. Écrire une fonction `Translate` qui prend en paramètre les coordonnées d'un point du plan et un vecteur de translation, et qui retourne les coordonnées du point résultat.
2. Écrire la même fonction pour la rotation (`Rotate`).
3. Construire et afficher graphiquement un carré (ou autre forme géométrique 2D).
4. Illustrer graphiquement l'utilisation des fonctions `Translate` et `Rotate`.

6.3 Méthode de Newton

La méthode de Newton est une méthode itérative, qui permet de résoudre numériquement l'équation $f(x) = 0$, par la récurrence suivante :

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \text{ avec } x_0 \text{ une valeur initiale.}$$

1. Écrire la fonction `NewtonSqrt(x, initial_range)`, qui permet de calculer, par la méthode de Newton, la racine carrée d'un nombre x donné avec une valeur initiale dans l'intervalle `initial_range`.
2. Illustrer graphiquement la convergence de cette méthode.
3. Comparer avec le résultat de la fonction `sqrt(x)`.

6.4 Évaluation d'un polynôme de degré n

Soit a un polynôme de degré n donné par ses coefficients a_i : $a(x) = \sum_{i=0}^{n} a_i x^i$. La méthode de Horner permet d'évaluer ce polynôme de la manière suivante :

$$a(x) = \big(\ldots(a_n x + a_{n-1})x + \cdots\big)x + a_0.$$

1. Créer manuellement ou générer un polynôme de degré n donné.
2. Écrire la fonction `Horner` qui évalue un polynôme de degré n donné en un point x donné.

La simple précision (associée à la double) peut être utilisée pour estimer la précision d'un algorithme numérique. Cette technique standard consiste alors à exécuter cet algorithme à la fois en simple et double précision. En considérant le résultat double comme exact, une estimation de l'erreur d'évaluation en simple précision peut être déterminée en calculant la différence entre le résultat en simple précision et celui en double précision.

3. Afficher l'estimation de l'erreur d'évaluation par la méthode de Horner du polynôme a (différence des valeurs calculées en simple et double précision), sur l'intervalle $[-0.5, 0.5]$.

Références

[1] DJ. Higham, NJ. Higham, MATLAB GUIDE - *Second edition*, Society for Industrial and Applied Mathematics (SIAM), 2005

[2] ML. Overton, Numerical computing with IEEE floating point arithmetic, Society for Industrial and Applied Mathematics (SIAM), 2001