Chapitre IV: Les fonctions

1. Introduction

Il est possible d'enregistrer une séquence d'instructions dans un fichier appelé un "**M-file**", et de le faire exécuter par Matlab. Les M-files se termine par une extension "**.m**" pour être considéré par Matlab comme un ficher d'instruction. On distingue deux types de M-files, les fichiers de scripts et les fichiers de fonctions.

2. Fichiers Script

Un ficher script est un ensemble d'instructions Matlab qui joue le rôle de programme principal. Le fichier scripte est nommé sous la forme suivante:

nom_de_fichier.m

On exécute ce fichier dans la fenêtre Matlab en tapant: **nom_de_fichier**.

Le fichier script permet de lancer les même opérations en faisant appel à ce fichier, au lieu de les retaper à chaque fois que nécessaire.

Exemple

Soit le fichier "test.m" comme suit:

```
% test.m

clear all
x = 4;
y = 2;
a = x + y
b = x * y
whos
```

produit la sortie suivante lorsque appelé:

```
>> test
a =
     6
b =
     8
Name          Size                Bytes  Class
a             1x1                     8  double array
b             1x1                     8  double array
x             1x1                     8  double array
y             1x1                     8  double array
Grand total is 4 elements using 32 bytes
>>
```

3. Fichiers fonction

Les fichiers de fonctions permettent de définir des fonctions qui ne figurent pas parmi les fonctions Matlab incorporées (built-in functions).

On définit la fonction (fonc) de la manière suivante:

Function [VARS1,…., VARSn] = fonc(VARE1,….., VAREn)
Séquence d'instructions

Où

- VARS1,…., VARSn: sont les variables de sortie de la fonction;
- VARE1,…., VAREn: sont les variables d'entrée de la fonction;
- Séquence d'instructions: est le corps de la fonction.

Exemple

Une fonction à sortie unique (le résultat d'addition):

```
function a = ma_fonction(x,y)
a = x + y;
b= x * y;
```

produit la sortie suivante:

```
>> a = ma_fonction(4,2)
a =
        6
>> whos
Name          Size                Bytes  Class
a             1x1                     8  double array
Grand total is 1 element using 8 bytes
>>
```

Le résultat de la multiplication n'est pas disponible. On peut cependant modifier les sorties de la manière suivante:

```
function [a,b] = ma_fonction(x,y)
a = x + y;
b= x * y;
```

produit la sortie suivante:

```
>> [a,b] = ma_fonction(4,2)
a =
        6
b =
        8
```

```
>> whos
Name        Size                    Bytes  Class
a           1x1                         8  double array
b           1x1                         8  double array
Grand total is 2 elements using 16 bytes
>>
```

Remarques

- Il n'y a pas de mot-clé (par exemple end) pour indiquer la fin de la fonction. La fonction est supposée se terminer à la fin du fichier.
- On ne peut écrire qu'une seule fonction par fichier (qui doit porter le nom de cette fonction). Toutefois depuis la version 5 de Matlab, il existe la notion de sous-fonction.
- Une sous-fonction est une fonction écrite dans le même fichier qu'une autre fonction (dite principale) et qui ne sera utilisable que par cette fonction principale (une sous-fonction ne peut être appelée par un autre sous-programme que la fonction principale).
- Si le fichier ne commence pas par le mot-clé **function** on a tout simplement écrit un script.

4. Les fonctions mathématiques usuelles

Toutes les fonctions mathématiques de base sont déjà programmées dans Matlab. Les fonctions les plus courantes sont présentées dans le tableau 1.

5. Les fonctions matricielles

Toutes les fonctions mathématiques de base sont déjà programmées dans Matlab. Les fonctions les plus courantes sont présentées dans le tableau 2.

Tableau 1- Fonctions courantes en Matlab.

Fonction	Description
sin(x)	sinus de x; x en radians
cos(x)	cosinus de x; x en radians
tan(x)	tangente de x; x en radians
exp(x)	exponentielle de x
log(x)	logarithme en base de e de x
sqrt(x)	racine carrée de x
power(x,a)	puissance a de x (x à la puissance a)
abs(x)	valeur absolue de x
asin(x)	$\sin^{-1}$ de x; résultat en radians
acos(x)	$\cos^{-1}$ de x; résultat en radians
atan(x)	$\tan^{-1}$ de x; résultat en radians
sinh(x)	sinus hyperbolique de x
cosh(x)	cosinus hyperbolique de x
tanh(x)	tangente hyperbolique de x
round(x)	arrondit un nombre à l'entier le plus près
floor(x)	arrondit vers l'entier immédiatement au dessous
ceil(x)	arrondit vers l'entier immédiatement au dessus

Tableau 2 – Fonctions matricielles en Matlab

Fonction	Description
det (A)	Déterminant de la matrice A
trace (A)	trace de la matrice A
rank (A)	rang de A (nombre de colonnes ou de lignes linéairement indépendantes)
norm (A)	norme de A (peut s'applique à un vecteur V)
inv (A)	inverse de la A
poly (A)	polynôme caractéristique de la matrice A
eig (A)	valeur propre de la matrice A
svd (A)	décomposition en valeur singulière de la matrice A
min (V)	indice du plus petit élément du vecteur V
max (V)	indice du plus grand élément du vecteur V
sum (V)	somme des éléments du vecteur V

6. Quelques fonctions spéciales

6.1. Les variables symboliques

Matlab permet la manipulation d'expressions symboliques (c'est-à-dire des variables sans affectation numérique). Il faut cependant déclarer les variables au préalable.

Exemple

Calcul de la matrice Jacobéenne associée à un système de deux équations différentielles:

```
%déclaration des variables symboliques
syms z1 z2 real

%déclaration de fonctions mathématiques
f1 = z1-z2^2+4;          % f1 := del z1/del t
f2 = 5(1-exp(-z1/66);  % f2 :=del z2/del t

% Création de la matrice jacobienne

% A = [(del f1/del z1) (del f1/del z2)
%         (del f2/del z1) (del f2/del z2)]


A = jacobian([f1;f2],[z1 z2]);
```

Matlab produit le résultat suivant (remarquez la classe des variables);

```
>> A = Jacobian([f1;f2],[z1 z2])
A =  [                    1,              -2*z2]
      [5/(66*exp(-1/66*z1),              0]

>>whos
Name          Size                Bytes  Class
A             2x2                 354    sym object
f1            1x1                 142    sym object
f2            1x1                 158    sym object
z1            1x1                 128    sym object
z2            1x1                 128    sym object
```

Il est possible ensuite d'évaluer l'expression à différents points (z1, z2) en utilisant la fonction *feval*.

```
>> z1 = 0;
>> z2 = 0;
>> A1 = feval(A)
A1 =
         1.0000          0
```

```
    0.0758          0
```

6.2. Racine d'une équation polynomiale

Supposons qu'on veuille résoudre l'équation polynomiale d'ordre 5 suivante:

$$x^5 + 3x^4 + 8x^3 + 12x^2 - x + 4 = 0$$

Il suffit de déclarer un vecteur contenant les coefficients du polynôme:

```
>>poly = [1 3 -8 12 -1 4];
```

et d'appeler la fonction *roots* de Matlab:

```
>> roots(poly)
ans =
        -5.0623
         1.1051 + 1.1056i
         1.1051 – 1.1056i
        -0.0739 + 0.5638i
        -0.0739 – 0.5638i
>>
```

6.3. Intégration numérique

La fonction d'intégration la plus populaire est *quad*. Il s'agit d'une méthode adaptive de l'algorithme de Simpson. Supposons qu'on veuille évaluer l'intégrale suivante:

$$\int \frac{\sin(x)dx}{x}$$

Pour l'intervalle]0, 1]. Il suffit d'utiliser les lignes de code suivantes:

```
>> S = quad('sin(x)./x',0,1)
S=
        0.9461
>>
```