Chapitre 04 : Notion de programmation

Jusqu'à présent, nous avons utiliser Matlab en mode interactif pour effectuer des calculs scientifiques ou pour évaluer des expressions. Les commandes sont saisies directement dans la fenêtre de commande, après le prompt >>, et dès qu'on écrit une commande, Matlab l'exécute et renvoie instantanément le résultat.

Cependant, pour résoudre des problèmes complexes, il est souvent nécessaire d'écrire plusieurs instructions organisé dans plusieurs lignes. Une collection d'instructions bien structurées visant à résoudre un problème donnée s'appelle un programme.

Ce chapitre présente les mécanismes d'écriture des programmes et de faire appel aux structures de programmation comme les boucles et les réalisations conditionnelles.

1. Les commentaires

Les commentaires sont des phrases explicatives ignorées par MATLAB et destinées pour l'utilisateur afin de l'aider à comprendre la partie du code commentée.

Le signe de pourcentage (%) permet d'écrire du texte que MATLAB ignorera à l'exécution.

```
% ceci est le premier commentaire
x = 30 ; % deuxième commentaire
```

2. Les entrées-sorties

2.1. Sorties, la commande disp

La commande disp permet d'afficher un tableau (scalaire, vecteur, matrice) de valeurs numériques ou de caractères. L'autre façon d'afficher un tableau est de taper son nom.

On utilise disp pour afficher un résultat numérique. $disp(a)$, où a est un tableau, affiche la valeur de a.

Exemple:

```
>> a = [1 2;3 4] ; disp(a)
1   2
3   4
```

disp peut utiliser avec une chaîne de caractères pour afficher un message.

disp('Message').

Exemple:

disp('Calcul du déterminant de la matrice A').

Cette commande sera souvent utilisée avec num2str pour afficher les valeurs des expressions numériques. La commande num2str (« number to string ») convertit une valeur numérique en une chaîne de caractères.

Exemple:

```
>> a = [1 2 ; 3 4] ;
 >> disp([ 'ordre de la matrice a : ' num2str(size(a,1)) ] );
ordre de la matrice a : 2
```

2.2. Entrées - input

La fonction input permet la saisie d'une valeur depuis le clavier:

- ✓ Pour les valeurs numériques, n = input('message') affiche message et affecte à la variable n la valeur numérique entrée au clavier.
- ✓ Pour les chaînes de caractères, str = input('message','s') affiche message et affecte à la variable str la valeur entrée au clavier considérée alors comme une chaîne de caractères.

```
iter = input('Entrez le nombre d'it{\'e}rations : ');
% input affiche le message et attend une valeur à partir du clavier. La touche Entrée associe la
%valeur entrée à la variable iter et continue l'exécution du programme.
```

3. Structures conditionnelles

Il existe deux structures possibles permettant de réaliser des tests sur les données.

- La structure if permet de tester la valeur d'une variable et d'effectuer différents traitement suivant les cas testés.
- La structure switch permet de choisir entre différents cas, et de faire correspondre un traitement adapté à chacun des cas reconnus.

3.1. L'instruction if

L'instruction if permette d'orienter l'exécution du programme en fonction de la valeur logique d'une condition. Différentes formes de structure conditionnelles if existent sous MATLAB.

```
if expr
        Instructions % instructions exécutées si expr est vraie
end
```

Dans cette forme, si l'expression logique (expr) a la valeur vrai, les instructions entre le if et le end seront exécutées, sinon elles ne seront pas.

```
a=input( ' donner la valeur de a \n a= ' )
if a>= 0
        b = sqrt(a)
end
```

Une variante plus élaborée est:

```
if expr
        Instructions % instructions exécutées si expr est vérifiée
else
        Instructions % instructions exécutées si expr n'est pas vérifiée
end
```

Si la condition est évaluée à vrai, les instructions entre le if et le end seront exécutées, sinon les instructions entre le else et le end seront exécutées.

```
a = 9;
if mod(a,2)= =0 % MOD retourne le reste de la division a sur 2
        disp('a est un nombre pair')
else
        disp('a est un nombre impair')
end
```

S'il est nécessaire de vérifier plusieurs conditions au lieu d'une seule, on peut utiliser des clauses elseif pour chaque nouvelle condition, et à la fin on peut mettre un else dans le cas ou aucune condition n'a été évaluée à vrai. Voici donc la syntaxe générale:

```
if expr01
        Instructions 01
elseif expr02
        Instructions 02
.....
.....
elseif exprn
        Instructions n
else
        Instructions si toutes les expressions étaient fausses
end
```

Exercice 01

Faire une séquence d'instructions Matlab qui résout le problème suivant:

$$\begin{cases} y = x^2 & si\ x < 0 \\ y = \sqrt{x} & si\ x > 0 \\ y = \dfrac{x}{10} & si\ 10 \leq x < 50 \\ y = -x, & si\ \ x \geq 50 \end{cases}$$

3.2. L'instruction switch

L'instruction switch exécute un block d'instructions parmi plusieurs choix si sa condition est satisfaite.

```
switch  var
        case  val1
                Instructions
        case val2
                Instructions
        ...
        ...
        otherwise
                Instructions
end
```

Si la variable var est égale à l'une des valeurs val1, val2, ..., alors la séquence d'instructions correspondante est exécutée. Le programme reprend ensuite à la première instruction suivant le mot-

clé end. Si la variable var n'est égale à aucune des constantes la séquence d'instructions par défaut est exécutée.

Exemple 01

```
Mention = input('Donner la Mention:');
switch(grade)
        case 'A'
                disp('Excellent' );
        case 'B'
                disp('Trés bien\n' );
        case 'C'
                disp('Bien\n' );
        case 'D'
                disp('Essaye encore\n' );
        otherwise
                disp('Invalid grade\n' );
end
```

Exemple 02

```
rep = input('Votre reponse (oui, non) :');
switch rep
        case {'oui','o'}
                disp('bravo ...')
         case {'non','n'}
                disp('perdu ...');
        otherwise
                disp('Essayer autre chose');
end
```

4. Les Boucles

Une boucle permet de répéter les même instructions un grand nombre de fois en faisant varier un paramètre.

4.1. La boucle While

La boucle while permet d'exécuter des instructions de manière répétée (un nombre indéterminé de fois) tant qu'une condition donnée est satisfaite. Elle a la forme générale suivante:

```
while expression_logique
        Instructions
end
```

Tant que l'expression de while (expression_logique) est évaluée à vrai, l'ensemble d'instructions s'exécutera en boucle.

La boucle while s'écrit de la manière suivante :

```
n=0; v=[];
while (n < 10)
        v=[v, n.^2]
        n=n+1;
end
```

Cette boucle construit le vecteur v dont chaque cellule contient la valeur de n^2, pour tous les entiers n de 0 à 9.

Exercice 02

1. Faire un programme qui calcule la somme suivante:

$$S = 1 + \frac{2}{2!} + \frac{3}{3!} + \cdots + \frac{n}{n!}$$

2. Modifier ce programme, où le programme s'arrête le calcul quand S>2.5

4.2. La boucle for

L'instruction for répète l'exécution d'un groupe d'instructions pour un nombre d'itérations définis. Elle a la forme générale suivante:

```
for var=expr_vecteur
        Instructions
end
```

expr_vecteur corresponde à la définition d'un vecteur : début : pas : fin ou début : fin

La variable var doit parcourir tous les éléments du vecteur défini par l'expression, et pour chacun il va exécuter le groupe d'instructions.

Exemple

Dans le tableau suivant, nous avons trois formes de l'instruction for

nfac = 1; n=6
for k = 1:n
  nfac = nfac*k;
end
	v=[];
	for k = 2:2:10
	  v=[v ; k^2];
	end
		v=[3 -5 7 4 2.5];
		for k = v
		  w=[w , k^2];
		end

Exercice 03

Essayer d'exécuter les trois exemples de la boucle for illustrée dans le tableau précédant et expliquer le rôle de chaque boucle.

5. Les M fichiers

MATLAB permet d'exécuter des programmes contenant une suite d'instructions stockées dans un fichier. Ces fichiers ont une extension .m. Il y a deux types de fichier .m:

- Les scripts
- Les fonctions

5.1. Les scripts

Un script (script file) est un fichier (**pgm01.m** par exemple) contenant une suite des instructions MATLAB. Ces fichiers ont l'extension .m.

Pour créer un script : **File > New > Blank M-File.**

Nous pouvons écrire dans un script toutes les commandes que nous désirons exécuter consécutivement.

Pour exécuter un script existant, il suffit de taper le nom du script (sans extension) dans la fenêtre de commande suivi de < entrer >. Ou appuyer sur le bouton Run de l'Editeur.

Le script ne prend pas d'argument en entrée et ne retourne rien. Ces deux points le différentient des fonctions.

Exemple

```
x= []; y= round(rand(1,15)*12);

v= find(y>=5);

x=y(v)
```

5.2. Les fonctions sous MATLAB

Les fichiers de fonctions permettent de créer des fonctions utilisateur qui ne figurent pas parmi les fonctions incorporées de MATLAB et de les utiliser de la même manière que ces dernières.

Une fonction est caractérisée par son nom et elle peut prendre des arguments et renvoyer des valeurs. (Par exemple, si l'on fourni à une fonction une date de naissance, cette fonction peut renvoyer l'âge de la personne). La définition d'une fonction se fait de la manière suivante:

```
function [vars1, vars2, …,varsN]=nomfonct(vare1, vare2, …,vareM)
        …
        Séquence d'instructions
        …
end
```

Avec :

- ✓ vars1, ..., varsN sont les variables de sortie (arguments de sortie) de la fonction;
- ✓ vare1, ..., vareM sont les variables d'entrée (arguments d'entrée) de la fonction;
- ✓ Séquence d'instructions est le corps de la fonction.

Le fichier doit impérativement commencer par le mot-clé function. Suit entre crochets les variables de sortie de la fonction, le symbole =, le nom de la fonction et enfin les variables d'entrée entre parenthèses.

Pour écrire une fonction dans Matlab, une règle doit être à respecter est que de donner au fichier .m le même nom que la fonction que l'on est en train d'écrire. Par exemple, une fonction qui s'appellerait mafact devra être écrite dans le fichier mafact.m.

L'appel de la fonction s'effectue à travers la saisie de son nom et des arguments correspondants.

[var_s1, var_s2, …,var_sn]=nomfonct(var_e1, var_e2, …,var_en) ;

Remarquer que le mot « function » n'y figure pas.

Exemple

```
function [mean, stdev] = stat(x)
% STAT Calcul de la moyenne et de l'{\'e}cart type.
% Pour un vecteur x, stat(x) rend la moyenne de x;
% [mean, stdev] = stat(x) rend la moyenne et l'{\'e}cart type.
% Pour une matrice x, stat(x) agit par colonnes.
m= length(x);
mean = sum(x)/m;
```

On écrit dans la fenêtre de commande, la commande [xm, xd] = stat(x), alors la moyenne et l'écart type de x sont calculés et affectés aux les variables xm et xd.
Notez que c'est possible de prendre une seule variable de retour, même si la fonction en rend 2. Par exemple la commande xm = stat(x) (les crochets [et] ne sont pas nécessaires dans ce cas), calcule seulement la moyenne de x.