TP 5 : Programmation MATLAB

Remarque IMPORTANTE : dans ce TP, lorsqu'un exemple ou un exercice est donné, vous êtes invité fortement à le réaliser et à en noter le résultat.

Introduction

Semblables à celles qui sont utilisées dans de nombreux langages à savoir les structures if ... end, while... end ainsi que la boucle for ... end , elles sont assez faciles à mettre en œuvre.

Utilisation de l'éditeur MATLAB

Création de m-files

Les m-files permettent d'enregistrer les scripts sous forme de fichiers-texte et servent en particulier à définir de nouvelles fonctions (une grande partie des fonctions prédéfinies de MATLAB sont stockées sous forme de m-files dans la toolbox matlab).

Les m-files peuvent être crées par n'importe quel éditeur. Dans les versions récentes de MATLAB il existe un petit éditeur intégré que l'on peut appeler à partir du menu file ou à partir de la barre de menu de la fenêtre de commande.

Exemple :

Dans la fenêtre de l'éditeur tapez les lignes suivantes :

```
% script - essai . m

a = .5;

b = pi;

c = a * b
```

Sauvez le fichier dans le répertoire de travail sous le nom de essai.m.

Rappel : la présence d'un point-virgule ; à la fin des deux premières lignes du script

a neutralisé l'affichage des valeurs de a et b

Exécution d'un m-file

Pour exécuter le script contenu dans un m-file et Il suffit de taper le nom de ce m file dans la fenêtre de commande suivi de < entrer >

>> essai

Eléments d'écriture de m-files

a. Commentaires : les lignes de commentaires sont précédées du caractère %.

b. Entrées - input et menu

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

- 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

Exemples :

>> n = input('Entrez la valeur de n ')

>> nom = input('Entrez votre nom ','s')

La fonction menu génère un menu dans lequel l'utilisateur doit choisir une option :

result = menu('titre', 'opt1', 'opt2', . . . , 'optn')

La valeur retournée dans la variable result est égale au numéro de l'option choisie. menu est souvent utilisé en relation avec la structure algorithmique switch-case.

Exemple : result = menu('Traitement', 'Gauss', 'Gauss-Jordan', 'Quitter')

c. Affichages - disp - num2str

La fonction num2str(x) où x est un nombre, retourne la valeur littérale de ce nombre.

Exemple : >> s = [’la valeur de pi est : ’ num2str(pi)]

La commande disp(t) où t est une chaîne de caractères ou un tableau, affiche la valeur de cette chaîne de caractère ou de ce tableau sans faire référence au nom de la variable. Cette commande sera souvent utilisée avec num2str pour afficher les valeurs des expressions numériques.

Exemple :

>> a = [1 2;3 4] ;

>> disp(a)

>> disp([’ordre de la matrice a : ’ num2str(size(a,1))])

Interaction avec le système d’exploitation

MATLAB possède des fonctions et des commandes qui permettent d’obtenir la liste des répertoires accessibles ou matlabpath, la liste des fichiers d’un répertoire donné, les éditer et éventuellement les effacer :

– addpath path : ajoute path à la liste matlabpath des répertoires accessibles par MATLAB ;

– p = pwd: retourne dans p le chemin d’accès au répertoire de travail actuel ;

– cd path : change le répertoire de travail pour celui spécifié par path ;

– d = dir ou d = ls: retourne dans d la liste des fichiers du répertoire de travail ;

– what : retourne la liste des m-files et des mat-files du répertoire de travail ;

– edit test : édite le m-file test.m, identique à Open dans le menu File ;

– delete test.m : efface le m-file test.m ;

– type test : affiche le m-file test.m dans la fenêtre de commande.

Opérateurs de comparaison et opérateurs logiques

a) **Les opérateurs de comparaison** sont:

= = : égale à (X= = Y)

> : Strictement plus grand que (X > Y)

< : Strictement plus petit que (X< Y)

>= : plus grand ou égale à (X>= Y)

<= : plus petit ou égale à (X<= Y)

~= : différent de (X~=Y)

b) **Les opérateurs logiques** sont:

& : et (X &Y)

| : ou (or) (X | Y)

-: Non X (-X)

Il n'existe pas dans MATLAB de vrai type booléen, que false est représenté par la valeur 0 et que true est représentée par la valeur 1 et par extension par toute valeur non nulle.

Exemples :

```
>> 2&3
>> 2&0
>> 2|3
>> ~3
>> 2==3
```

Instructions de contrôle

Boucle for (parcours d'un intervalle)

Sa syntaxe est:

```
for indice = borne_inf : pas : borne_sup
        Séquence d'instructions
```

end

Exemple faire un programme Matlab qui calcule la somme suivante

$$\sum_{i=3}^{n} i$$

Solution:

```
n=input('donner la valeur de n');
s=0;
for i=3:n
        s=s+i;
end
disp('la somme s est: '),s
%disp(['la somme s est: ',num2str(s)]
```

L'exécution:

```
>> ex1_matlab
donner la valeur de n6
la somme s est:
s=
        18
```

Boucle While (tant que)

Sa syntaxe est:

```
while expression logique
        Séquence d'instructions
end
```

Exemple: faire un programme sous matlab qui calcule la somme suivante:

S=1+2/2! +3/3!+... on arrête le calcule quand S>2.5

Solution:

```
clear all
s=1;i=1,f=1;
while s<=2.5
        i=i+1
        f=f*i;
        s=s+i/f
end
```

L' execution:

```
>> ex3_matlab
i=
        1
i=
        2
s=
        2
i=
        3
s=
        2.5000
i=
        4
s=
```

2.6667

L'instruction if (si)

1er cas : Sa syntaxe est:

```
if expression logique
    Séquence d'instructions
end
```

Exemple:

Faire un programme sous Matlab qui résout le problème suivant:

1. y = x si x<0

2. y=x2 si x>0

3. y=10 si x=0

Solution

```
clear all
x=input('introduire la valeur de x ');
if x<0
    y=x;
end
if x>0
    y=x^2;
end
if x==0
    y=10;
end
disp('la valeur de y est: '),y
```

L'exécution:

```
>> ex4_matlab
introduire la valeur de x 5
la valeur de y est:
y=
        25
```

2er cas :Sa syntaxe est:

```
if expression logique
        Séquence d'instructions
else
        Séquence d'instructions
end
```

Exemple:

Faire un programme sous Matlab qui résout le problème suivant:

1. $y = x$ si $x<0$
2. $y=x2$ si $x\geq 0$

Solution

```
clear all
x=input('introduire la valeur de x ');
if x<0
        y=x;
else
        y=x^2;
end
```

```
disp('la valeur de y est: '),y
```

L'exécution:

```
>> ex4_matlab
introduire la valeur de x 5
la valeur de y est:
y=
        25
```

3er cas :Sa syntaxe est:

```
if expression logique
        Séquence d'instructions
elseif expression logique
        Séquence d'instructions
elseif expression logique
        Séquence d'instructions
.
.
else
        Séquence d'instructions
end
```

Exemple:

Faire un programme sous Matlab qui résout le problème suivant:

Y=x si x<0

Y=10 si x=0

Y=sqrt(x) si 0<x<20

$Y=x^2$ si x =20

$Y=x^3$ si x>20

Solution

```
clear all
x=input('introduire la valeur de x  ');
if x<0
        y=x;
elseif x==0
        y=10;
elseif x==20
        y=x^2;
elseif x>0 & x<20
        y=sqrt(x);
else
        y=x^3;
end
disp('la valeur de y est: '),y
```

L'exécution:

```
>> ex5_matlab
introduire la valeur de x 60
la valeur de y est:
y=
        216000
```

L'instruction switch

Sa syntaxe est:

```
Switch var
case cst-1
        Séquence d'instructions-1
case cst-2
        Séquence d'instructions-2
.
.
.
case cst-N
        Séquence d'instructions-N
otherwise
        Séquence d'instructions par défaut
end
```

var: est une variable numérique ou chaîne de caractère. cst-1, cst-2....cst-N: sont des constantes numérique ou chaîne de caractères. Si l'instruction à exécuter est la même pour un ensemble de cas alors la syntaxe est : Case {cst-1, cst-2,...}

Exemples:

```
jj=input('donner le jour 1 :7 ');
switch jj
case 1
        disp('samedi')
case 2
        disp('dimanche')
```

```
case 3
        disp('lundi')
case 4
        disp('mardi')
case 5
        disp('mercredi')
case 6
        disp('jeudi')
case 7
        disp('vendredi')
end
%
mm=input('donner le mois: ');
switch mm
case 'Ja'
        disp('Janvier')
case 'F'
        disp('Février')
case 'M'
        disp('Mars')
case 'Av'
        disp('Avril')
otherwise
        disp('autre')
```

```
end
```

A l'exécution:

```
>> exp_switch
donner le jour: 4
mardi
donner le mois: 'Av'
Avril
```

Instructions d'interruption d'une boucle

Il est possible de provoquer une sortie prématurée d'une boucle de contrôle.

- L'instruction ***break*** permet de sortir d'une boucle for ou d'une boucle while. En cas de boucles imbriquées, on interrompt seulement l'exécution de la boucle intérieure contenant l'instruction break.
- L'instruction ***return*** provoque un retour au programme appelant (ou au clavier). Les instructions suivant le return ne sont donc pas exécutées. L'instruction return est souvent utilisée conjointement avec une instruction conditionnée par exemple pour tester dans le corps d'une fonction si les paramètres d'entrée ont les valeurs attendues.
- L'instruction ***error*** permet d'arrêter un programme et d'afficher un message d'erreur. La syntaxe est : *error(' message d''erreur ')*.
- L'instruction ***warning*** permet d'afficher un message de mise en garde sans suspendre l'exécution du programme. La syntaxe est *warning(' message de mise en garde ')*.

Il est possible d'indiquer à MATLAB de ne pas afficher les messages de mise en garde d'un programme en tapant ***warning off*** dans la fenêtre de commandes. On rétablit l'affichage en tapant ***warning on***.

- ***pause*** : interrompt l'exécution jusqu'à ce que l'utilisateur tape un return

- ***pause(n)*** : interrompt l'exécution pendant n secondes.

- ***pause off*** : indique que les pause rencontrées ultérieurement doivent être ignorées, ce qui permet de faire tourner tous seuls des scripts requérant normalement l'intervention de l'utilisateur.

Exemples:

a- Commande break

```
S=0 ;
for i=1:10
        i
        S=S+i^2
        if S > 15
                break
        end
end
```

A l'exécution:

```
i=
        1
S=
        1
i=
        2
S=
        5
```

```
i=
        3
S=
        14
i=
        4
S=
        30
```

b- Commande return

```
function somme(n)
%
if n < 0
        return
end
S=0;
for i=1:n
        S=S+i^2
end
```

A l'exécution:

```
>> somme(2)
S=
1
S=
5
```

```
>> somme(-2)
>>                → sortie de la fonction
```

c- Commande error

```
clear all
s=2;
for i=1:5
        s=s+i^2
        if s>5
                error('s est superieur à 5')
        end
end
```

A l'exécution:

```
i=
        1
s=
        3
i=
        2
s=
        7
??? Error using ==> myfile
s est superieur à 5
```

d- Commande warning

```
clear all
```

```
% warning off/on    → afficher ou masquer warning
s=2;
for i=1:5
        i
        s=s+i^2
        if s>5
                warning('s est superieur à 5')
        end
end
```

A l'exécution:

```
i=
        1
s=
        3
i=
        2
s=
        7
Warning: s est superieur à 5
> In C:\MATLAB6p5\work\exp.m at line 8
i=
        3
s=
        16
```

```
Warning: s est superieur à 5
> In C:\MATLAB6p5\work\exp.m at line 8
i=
     4
s=
     32
Warning: s est superieur à 5
> In C:\MATLAB6p5\work\exp.m at line 8
i=
     5
s=
     57
Warning: s est superieur à 5
> In C:\MATLAB6p5\work\exp.m at line 8
```

e- Commande pause

```
clear all
s=2;
for i=1:5
     i
     s=s+i^2
     if s>5
          pause
     end
end
```

A l'exécution:

```
i=
     1
s=
     3
i=
     2
s=
     7
|          → Le curseur clignote ⇒ tapez Entrée
```

Fonctions et m-files

Les m-files sont aussi le moyen pour l'utilisateur de MATLAB de définir ses propres fonctions.

Exemple :

La fonction moyenne calcule la moyenne des éléments d'une liste ou d'un vecteur

```
function m = moyenne(x)
% MOYENNE(X) : moyenne des éléments d'une liste ou d'un vecteur
% un argument autre qu'une liste ou un vecteur conduit a une erreur
[K,l] = size(x) ;
if ( (k~=1) & (l~=1) )
     error('l"argument doit être une liste ou un vecteur')
end
m = sum(x)/length(x) ;
```

La fonction est enregistrée sous le nom moyenne.m. Elle est ensuite appelée depuis le fenêtre de commande :

```
>> x = 1 : 9
>> y = moyenne(x)
>> A = [ 1 2 ; 3 4] ;
>> moyenne(A)
```

MATLAB utilise la section de commentaires en conjonction avec la commande help pour fournir la définition de la fonction :

```
>> help moyenne
```

Syntaxe

Une fonction est constituée par :

- un en-tête : function résultat = nom de la fonction (liste de paramètres)
- une section de commentaires :dont chaque ligne commence par le symbole % ;
- le corps de la fonction défini par un script.

Règles et propriétés

• Le nom de la fonction et celui du fichier m-file qui en contient la définition doivent être identiques. Ce fichier est le fichier m-file associé à la fonction.

• La commande *help* affiche les neuf premières lignes de la section de commentaires ; la première ligne est utilisée par la commande *lookfor* pour effectuer des recherches parmi les fonctions, MATLAB accessibles.

• Chaque fonction possède son propre espace de travail et toute variable apparaissant dans le corps d'une fonction est locale à celle-ci, à moins qu'elle ait été déclarée comme globale au moyen du qualificateur *global* précédant le nom de la variable dans tous les espaces de travail où cette variable

est utilisée.

• Un fichier m-file associé à une fonction (i.e. qui porte le nom d'une fonction et contient sa définition) peut contenir d'autres définitions de fonctions. La fonction qui partage son nom avec le fichier ou fonction principale doit apparaître en premier. Les autres fonctions ou **fonctions internes** peuvent être appelées par la fonction principale, mais pas par d'autres fonctions ou depuis la fenêtre de commande.

Exemple : l'ensemble des trois fonctions est enregistré dans un seul fichier m-file portant le nom de la fonction principale myStat.m :

```
function [m, v] = myStat(x)
% MYSTAT(X) : moyenne et variance des éléments d'une liste ou d'un vecteur
% un argument autre qu'une liste ou un vecteur conduit a une erreur
[k,l] = size(x) ;
if ( (k~=1) & (l~=1) )
        error('l"argument doit être une liste ou un vecteur')
end
m = moyenne(x);
v = variance(x);

function a = moyenne(u)
% Calcul de la moyenne
a = sum(u)/length(u);

function b = variance(u)
% Calcul de la variance
```

```
c = sum(u)/length(u);

u2 = (u - c).^2;

b = sum(u2)/length(u);
```

• L'exécution d'une fonction s'achève :

- lorsque la fin du script définissant la fonction a été atteint ;
- lorsque une commande return ou un appel de la fonction error a été rencontré
- return termine immédiatement l'exécution de la fonction sans que la fin du script définissant celle-ci ait été atteinte,
- error('message') procède de même, mais en plus, affiche le contenu de 'message'.

Le contrôle est alors renvoyé au point d'appel de la fonction, fenêtre de commande ou autre fonction.

Traitement des erreurs - try . . . catch

La commande try . . . catch a pour but de permettre un traitement qui permette à l'utilisateur d'intervenir en présence d'erreurs ou de situations inhabituelles. Sa syntaxe est la suivante :

try script1 catch script2 end

Le fonctionnement en est assez simple pour les habitués des langages modernes,java par exemple :

- l'exécution de script1 est lancée ;
- si une erreur survient, alors l'exécution de script1 est arrêtée et script2 est exécuté
- sinon, script1 est exécuté jusqu'à son terme, script2 n'est pas exécuté, les exécutions suivantes se poursuivent après le end final.

On peut utiliser *lasterr* pour accéder à l'erreur qui provoque l'arrêt de l'exécution de script1.

Optimisation des calculs

Les calculs sont accélérés de façon spectaculaire en utilisant des opérations vectorielles en lieu et

place de boucles. Comparons les deux fonctions suivantes (la commande tic déclenche un chronomètre ; toc arrête le chronomètre et retourne le temps écoulé depuis tic) :

```
function [t,b] = test1(n)
% détermine le temps mis pour créer la liste
% des racines carrées des entiers compris entre 1 et n
m = 0 ;
tic ;
for k = 1 : 1 : n
        b(k) = m+sqrt(k) ;
end
t = toc ;

function [t,b] = test2(n)
% détermine le temps mis pour créer la liste
% des racines carrées des entiers compris entre 1 et n
tic ;
a = 1 : 1 : n ;
b = sqrt(a) ;
t = toc ;
```

Les résultats obtenus montrent que test2 est plus efficace que test1.

Remarque : MATLAB contient un utilitaire appelé **profiler** qui est précieux pour étudier les performances d'une ou plusieurs fonctions. Les modalités d'utilisation du profiler ont évolué en

fonction des versions de MATLAB. On les trouvera dans l'aide en ligne help profile.

Exercice 1 :

Faire un programme Matlab qui calcule la somme suivante 1+1.2+1.4+1.6+1.8+2

Exercice 2 :

Ecrire un programme qui permet de chercher le plus petit et le plus grand éléments d'un vecteur colonne noté V choisit au préalable et de faire la somme de ses éléments. L'enregistrer sous tp5_2 puis l'exécuter.

Exercice 3 :

Ecrire des scripts qui permettent de calculer N ! (factoriel de N) en utilisant la boucle while puis la boucle for

Exercice 4 :

Calculer le PGCD selon cet algorithme :

a,b entiers positifs

tant-que $a \neq b$

si $a > b$ alors $a \leftarrow a - b$

sinon $b \leftarrow b - a$

fin tant-que

$PGCD \leftarrow a$