4. La Programmation

Jusqu'à présent dans ce cours, toutes les commandes ont été tapées dans la fenêtre de commande de MATLAB. Toute la liste de commandes est perdue de cette fenêtre lorsqu'on quitte MATLAB. (On peut toujours voir ces commandes dans la fenêtre **Command History**.)

On peut d'ailleurs créer des fichiers de commandes MATLAB - les M-files - qui sont composés d'une suite de commandes comme dans la fenêtre de commande.

Ces fichiers .m sont créés par l'éditeur de MATLAB ou par un autre éditeur ASCII. Pour entrer dans l'éditeur MATLAB on utilise la barre de tâche dans le fenêtre de commande.
File/New/M-file (ou bien File/Open/nom.m)

Il existe deux types de fichiers .m

- Les scripts - les suites de commandes. Un script voit toutes les variables dans l'espace de travail MATLAB
- Les fonctions - elles ont des paramètres en entrée et/ou en sortie. Toutes les variables définies à l'intérieur de la fonction sont "locales".

Les scripts et les fonctions peuvent comprendre les instructions de contrôle.

4.1 Scripts

Voici un script sauvegardé dans le fichier toto.m

toto.m

```
%calcul de la moyenne, des écart-types et les max d'un tableau

A = rand(50,3);

%calcul de la moyenne
ave = mean(A)

%calcul des ecart types
ec_type = std(A)

% calcul des max des colonnes
mx = max(A)
```

Dans la fenêtre de commande on peut voir ce qu'il y a dans toto.m en tapant la commande >> type toto. Aussi >>help toto affiche la première commentaire dans toto. Cà veut dire que c'est une bonne idée d'inclure un première commentaire pour expliquer le fonctionnement du script.

```
» toto                          exécution du script toto
ave =
   0.4914    0.5710    0.5202
ec_type =
   0.2857    0.2532    0.2718
mx =
```

0.9830 0.9994 0.9623

La commande >>**echo on** avant un appel à un script permet l'affichage des commandes du script avant d'être exécutée. On peut ainsi voir le déroulement du script sans faire appel au déboggeur.

4.2 Les Instructions de contrôle

Il y a très peu de mots réservés dans MATLAB mais les suivants le sont

if else elseif end	test - commande réalisée si vrai
for end	boucle
while end	boucle avec arrêt sur condition
switch case end	exécution selon valeur
break	sortie précoce d'une boucle

Exemples:

Testif.m

```
% exemple d'une instruction if
[n,m] = size(A);

if rem(n,2)==0
   res = ('matrice A un nombre pair des lignes')
else
   res= ('matrice A un nombre impair des lignes')
end
```

Exercices:

1. Ecrire un script permettant de calculer $\sum n^2$ de n = 1 tandis que la valeur donnée par n^2 n'est pas plus grande que 1000 utilisant une boucle **while**.

2. Un sinus cardinal = (sin(x))/x. Ecrire un script permettant de calculer le sinus cardinal d'un scalaire

4.3 Les Fonctions

Syntaxe de la déclaration dans l'éditeur:

```
function [ sortie1,sortie2,….] = nom_du_fonction(entrée1, entrée2,…..)
% aide en ligne
```

Syntaxe d'appel de la fonction (de la fenêtre de commande ou d'une autre fonction)

[sortie1,sortie2,….] = nom_du_fonction(entrée1, entrée2,…..)

Il existe des variables de MATLAB qui nous permettent des tester dès qu'on entre dans une fonction si on a appelé la fonction avec le bon nombre de paramètres:

nargin
nargout

matmax_min.m

```
function[ymax,ymin] = matmax_min(X)
% Calcul du max et du min d'une matrice

if nargin ~= 1
    error('il faut 1 paramètre d'entrée')
else
    ymax = max(max(X));
    ymin = min(min(X));
end
```

ATTENTION ** Le nom du fichier.m et de la fonction elle-même devraient être les mêmes.

4.3.1 Les variables globales

Toutes les variables utilisées dans une fonction sont des variables "locales". Pour créer des variables globales, on fait appel à la commande global <variables> dans les fonctions où les variables doivent être partagées.

matmax_min.m

```
function[ymax,ymin] = matmax_min(X)
% Calcul du max et du min d'une matrice

if nargin ~= 1
    error('il faut 1 paramètre d'entrée')
else
    global coeff
    ymax = max(max(X))*coeff;
    ymin = min(min(X))*coeff;
end
```

```
>> global coeff
>> coeff = 100;
>> ymax = matmax_min(A)
```

4.3.2. Les sous-fonctions

La fonction et les sous-fonctions ont la même syntaxe et sont placées dans le même fichier. Les sous-fonctions sont prioritaires par rapport aux fonctions du répertoire courant et du path.

Exemple: la fonction statperso calcule la moyenne et l'écart type d'un tableau de valeurs en faisant appel aux sous fonctions (mean et f_ecarttype)

statperso.m

```
function[moyenne,ecart] = statperso(A)
% statistiques sur les données

moyenne = mean(A);
ecart = f_ecarttype(A);

function m=mean(A)
%calcule de la moyenne

m = sum(sum(A))/prod(size(A));

function e = f_ecarttype(A)
%calcule de l'ecart type

e = std(A(:));
```

```
>> Donnee = rand(10);
>> [m,e] = statperso(Donnee)
m =
    0.5150
e =
    0.2968
```

Exercices

1. a)Créer une fonction qui normalise les colonnes d'une matrice par rapport au max des colonnes. Ecrire cette fonction pour traiter des matrices de taille (nb de colonnes) variable.
   b) Donnees.txt est un fichier texte. Utilisez-le comme vos données à normaliser

2. Ecrivez une fonction qui trouve les max et les min d'un signal oscillant.