Institut Supérieur d'informatique Année Universitaire 2016-2017

MODULE : Analyse d'information multimédia

TP 2 : Initiation à Matlab (2)

-Les entrées-sorties, les structures de contrôle et les fonctions-

# 1 Scripts et m-files

## 1. 1 Scripts

Un script est une séquence d'expressions ou de commandes. Un script peut se développer sur une ou plusieurs lignes. Les différentes expressions ou commandes doivent être séparées par une virgule, un point-virgule ou par le symbole de saut de ligne constitué de trois points . . . suivis de <entrée> (le rôle des trois points et d'inhiber le mécanisme d'évaluation lors d'un passage à la ligne). Comme pour une expression unique, la frappe de <entrée> déclenche le processus d'évaluation. Les expressions sont évaluées dans leur ordre d'écriture. Seule la valeur des expressions suivie d'une virgule ou d'un saut de ligne est affichée, celle des expressions suivies d'un point-virgule, ne l'est pas.

## 2.1 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.

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.

## 3.1 Exécution d'un m-file

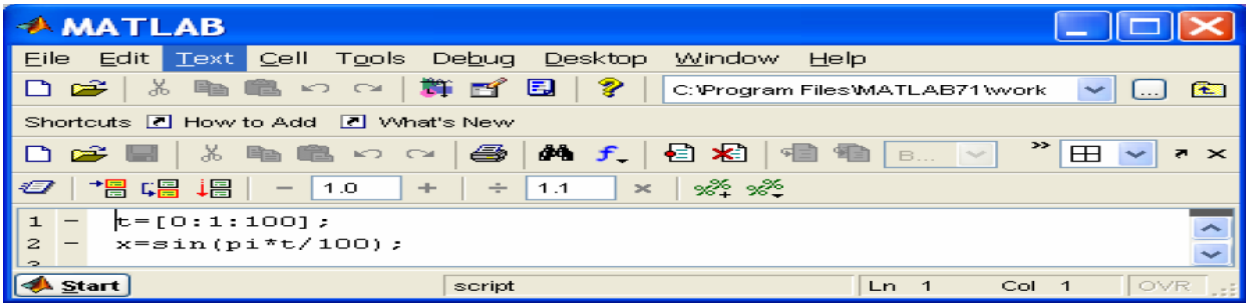
Pour exécuter le script contenu dans un m-file et Il suffit de taper le nom de ce m-file dans la fenêtre de commande suivi de < entrée >

```
>> essai
```

### Autres exemples

Exemple 1

Créer un fichier appelé «premierpas.m», dont le contenu est le suivant :

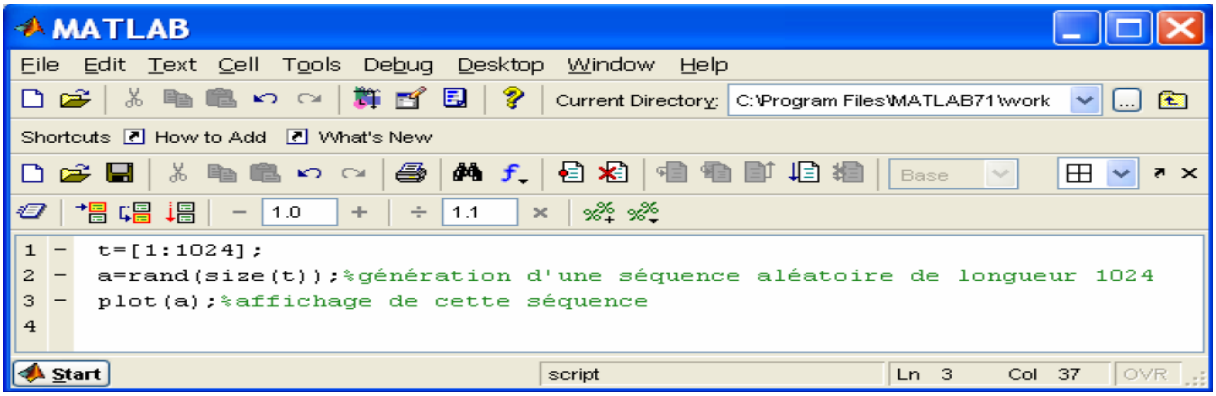



Exécuter le programme contenu dans le fichier.

La deuxième commande « plot(t,x) » trace la séquence x en fonction de la séquence t

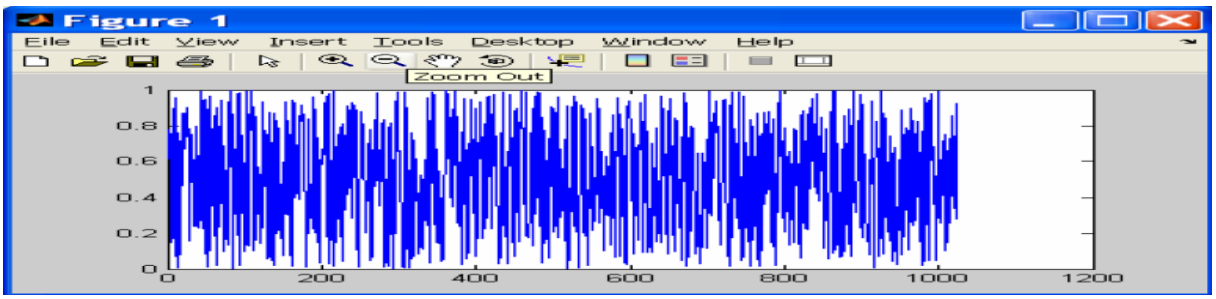

Exemple 2

Voici un programme essai1.m




Qui en engendre (fonction rand) une séquence pseudo aléatoire « a » dont la longueur est celle de la séquence « t » (soit ici 1024) et qui la trace Son exécution dans la fenêtre de commande


produit l'affichage




### 4.1 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.

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

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

### c. Affichage - disp

La valeur d'une variable est très simplement affichée en faisant évaluer une expression réduite à la variable elle-même.

```
>> a = [1 2] ;
>> a
```

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.

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

# 2 Instructions de contrôle

Ce sont les boucles et les branchements. Les boucles permettent de répéter commodément une suite d'instructions ; le nombre de fois est soit connu d'avance (boucles inconditionnelles), soit déterminé au cours de l'exécution (boucles conditionnelles). Les branchements conditionnels (ou tests) permettent de choisir un traitement parmi plusieurs possibles en fonction de critères évalués lors de l'exécution.

## 2.1- Boucles inconditionnelles

Leur forme générale est

*for k = val*

*liste d'instructions*

*end*

où k est une variable et val est un vecteur-ligne.

La liste d'instructions est exécutée m fois en donnant successivement à la variable k les valeurs val(1), . . . , val(m) où m(=length(val)) désigne la taille du vecteur val.

*Exemples* :

- Si $(x_n)$ est la suite récurrente définie par $x_0 = 1$ et $x_{n+1} = \sin(x_n)$ pour $n >= 0$, on calcule $x_{10}$ en écrivant

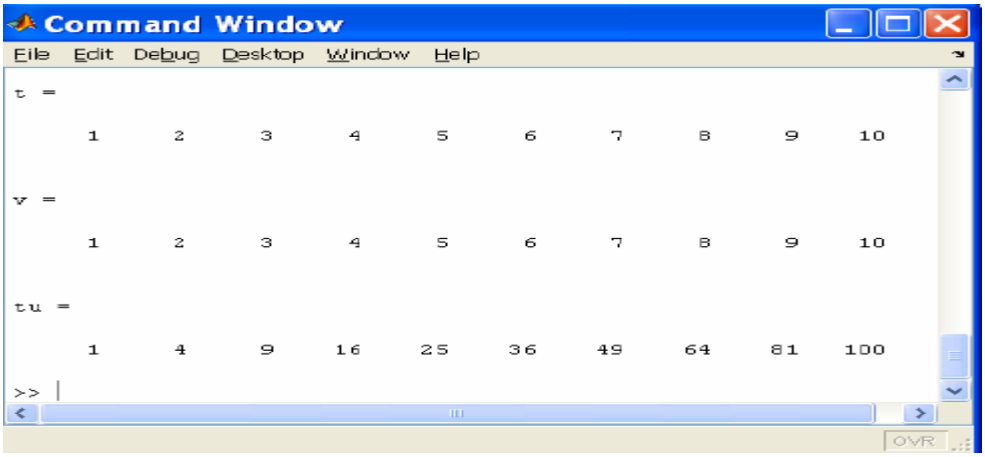
  *x=1;*

  *for n=1:10*

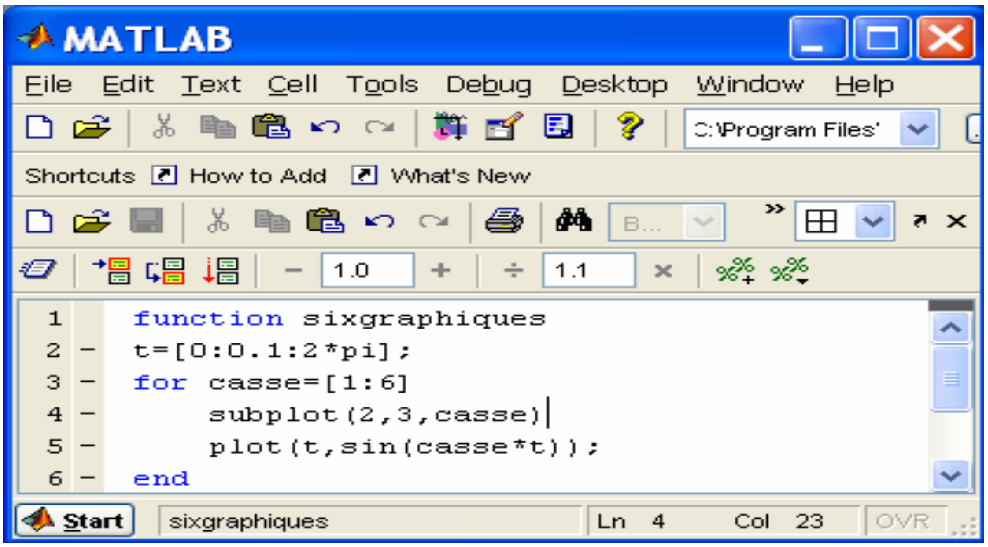
  *x=sin(x);*

  *end*

- Ecrire un programme qui permet de calculer le factoriel de n en utilisant la boucle For
- Ecrire un programme qui affiche le résultat suivant à l'écran (utiliser deux méthodes de calcul différentes)




- Recopier ce code à l'écran et voir son exécution




*Boucles imbriquées* : ces boucles peuvent être imbriquées comme dans l'exemple ci-dessous (calcul des 6 premières lignes du triangle de Pascal)

```
A=zeros(6);
for i=1:6
A(i,1)=1;
for j=2:i
A(i,j)=A(i-1,j-1)+A(i-1,j);
end
end
```

### 2.2- Boucles conditionnelles

Elles s'écrivent

```
while condition
        liste d'instructions
end
```

où la condition s'exprime à l'aide des opérateurs arithmétiques et logiques :

| Operator | Description |
|---|---|
| < | Less than |
| <= | Less than or equal to |
| > | Greater than |
| >= | Greater than or equal to |
| = = | Equal to |
| ~= | Not equal to |

| Operator | Description |
|---|---|
| & | AND |
| \| | OR |
| ~ | NOT |

A l'exécution, la liste d'instructions est répétée aussi longtemps que la condition est satisfaite. Par exemple,

```
n=0;
x=0;
while ( x <= 0.99 ) & ( x >= -0.99 )
n=n+1;
x=sin(n);
end;
```

calcule le plus petit entier positif n tel que | sin(n)| > 0.99 et la valeur de sin(n). On peut afficher ces deux nombres avec l'instruction [n,x] ou disp([n,x])

### 2.3 Branchements conditionnels

Leur forme générale est :

```
if condition_1
        liste_1 d'instructions
elseif condition_2
        liste_2 d'instructions
        ...
else
        liste d'instructions
end
```

Le nombre de blocs elseif condition, liste d'instructions est quelconque (éventuellement nul). De même, la présence d'un bloc else liste d'instructions à la fin est optionnelle.
Seule la liste d'instructions qui suit la première condition évaluée vraie est exécutée (il se peut très bien qu'il n'y en ait aucune). Ensuite, l'exécution se poursuit par l'instruction qui vient immédiatement après le end.

Une autre forme des branchements conditionnels est le switch ; leur forme générale est :

```
switch expression (expression est un scalaire ou une chaîne de caractères)
        case value1
                instructions (instructions effectuées si expression=value1)
        case value2
                instructions
                 ...
         otherwise
                instructions
end
```

# 3 Fonctions

### 3.1. 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

### 3.1. 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.

• 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.

• 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.

Exemple :

La fonction moyenne définie ci-contre calcule la moyenne des éléments d'une liste ou d'un vecteur. Le texte de cette fonction est saisi dans un éditeur.

```
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) ;
end
```

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

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

Exercices

1. Ecrire la fonction parite qui reçoit en entrée un vecteur de nombres entiers et qui sort un vecteur avec des 1 à la place de tous les nombres pairs et des -1 à la place de tous les nombres impairs.

2. Ecrire la fonction qui reçoit en entrée un vecteur x et qui donne en sortie deux vecteurs, l'un étant la partie entière de chaque composante de x et l'autre étant le nombre pair le plus proche de chaque composante de x.