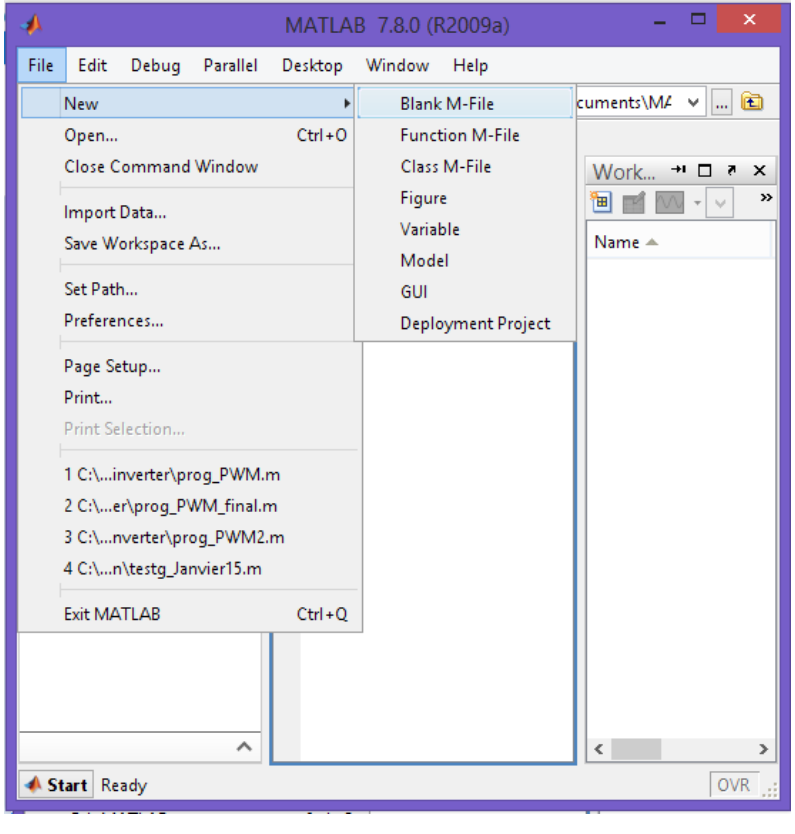
Le logiciel de simulation MATLAB
Partie 3
Programmation en script

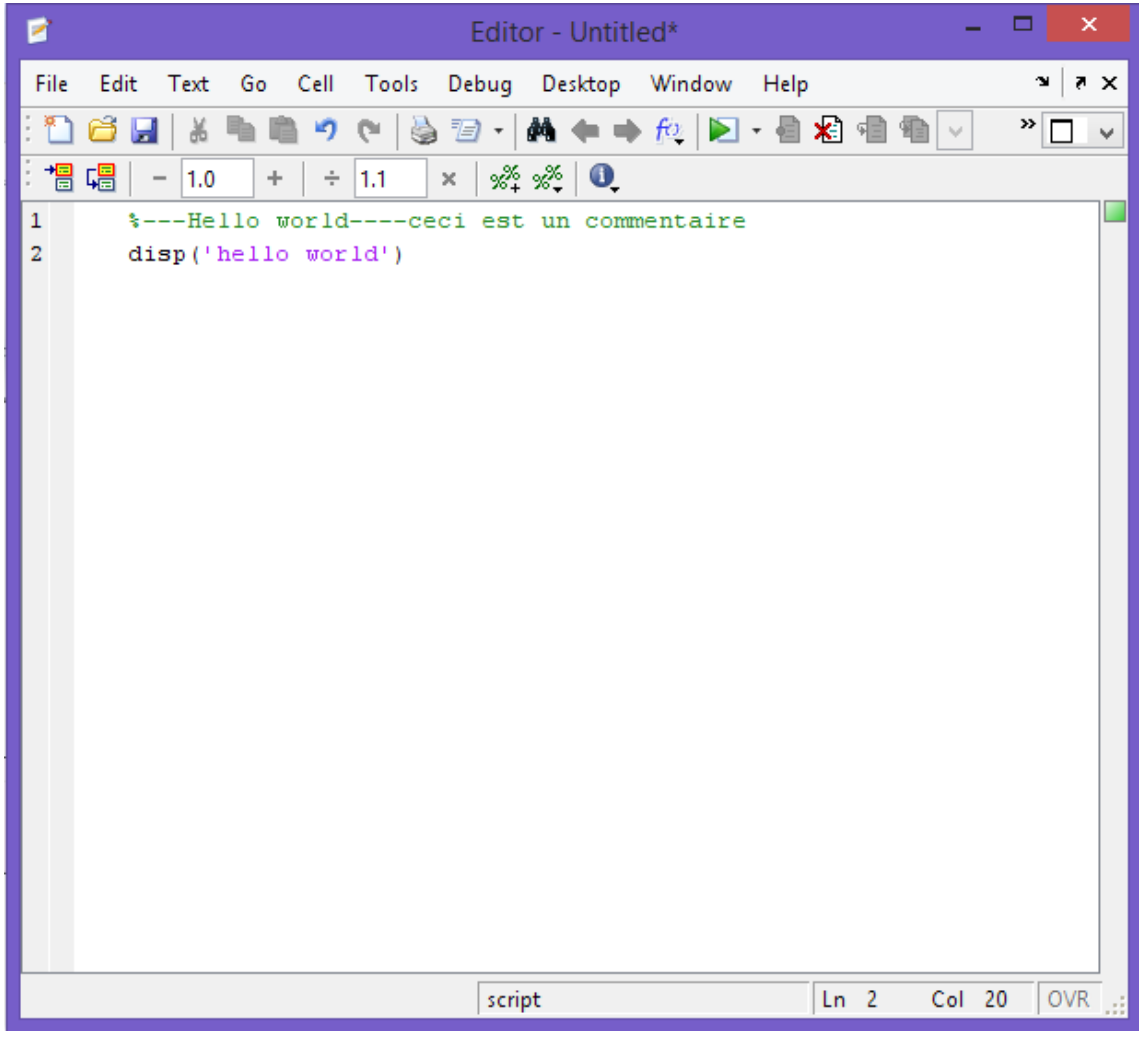
Professeur Ali Tahri
Université des sciences et de la technologie d'Oran Mohamed Boudiaf

1. Les fichiers m (m-files)

Un m-file est un fichier contenant une suite d'instructions que Matlab peut exécuter.
Un m-file peut aussi être utiliser comme fichier de librairie contenant des fonctions définies par l'utilisateur.
Pour créer un m-file, utilise le menu File ⟶ new → M-file



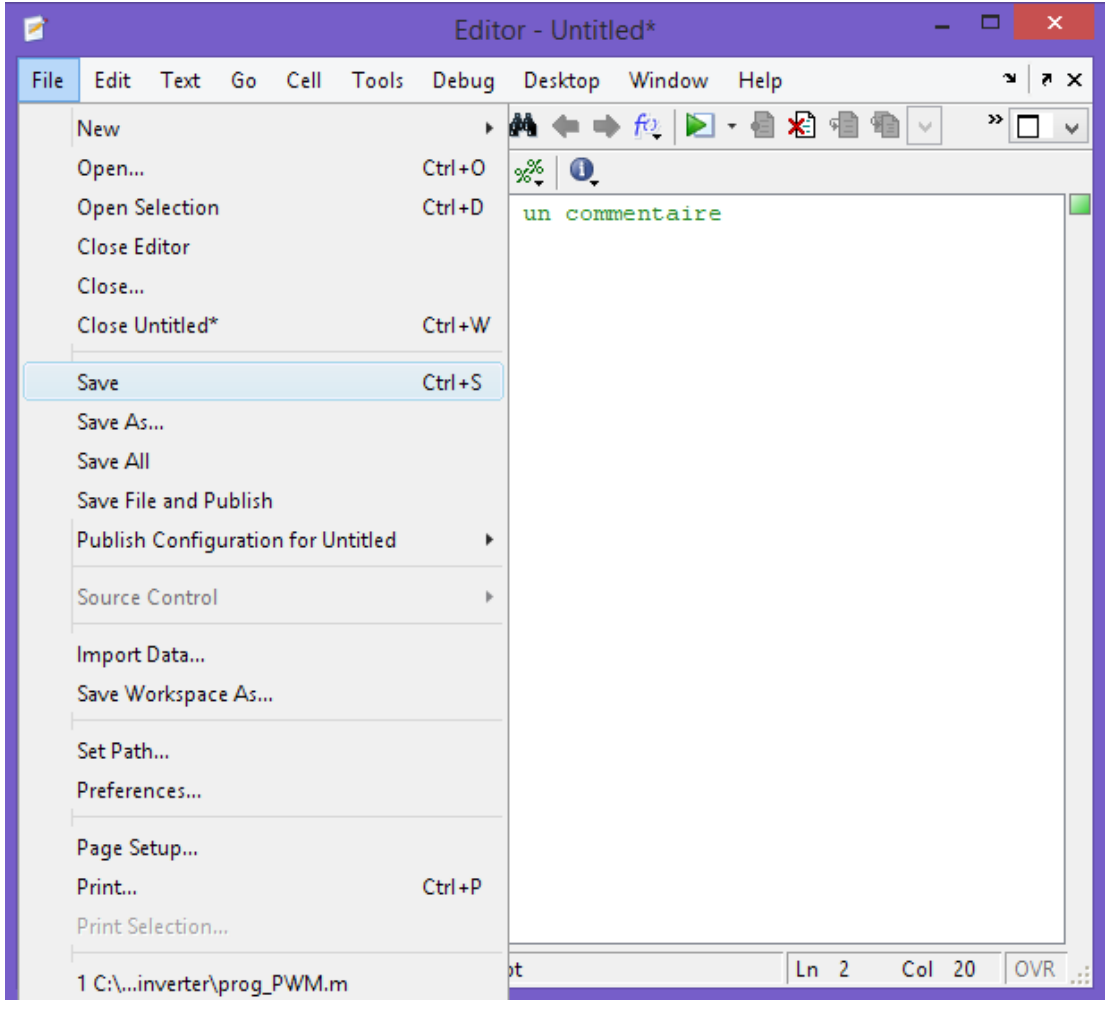
Une fois le M-file est crée, écrivez ceci



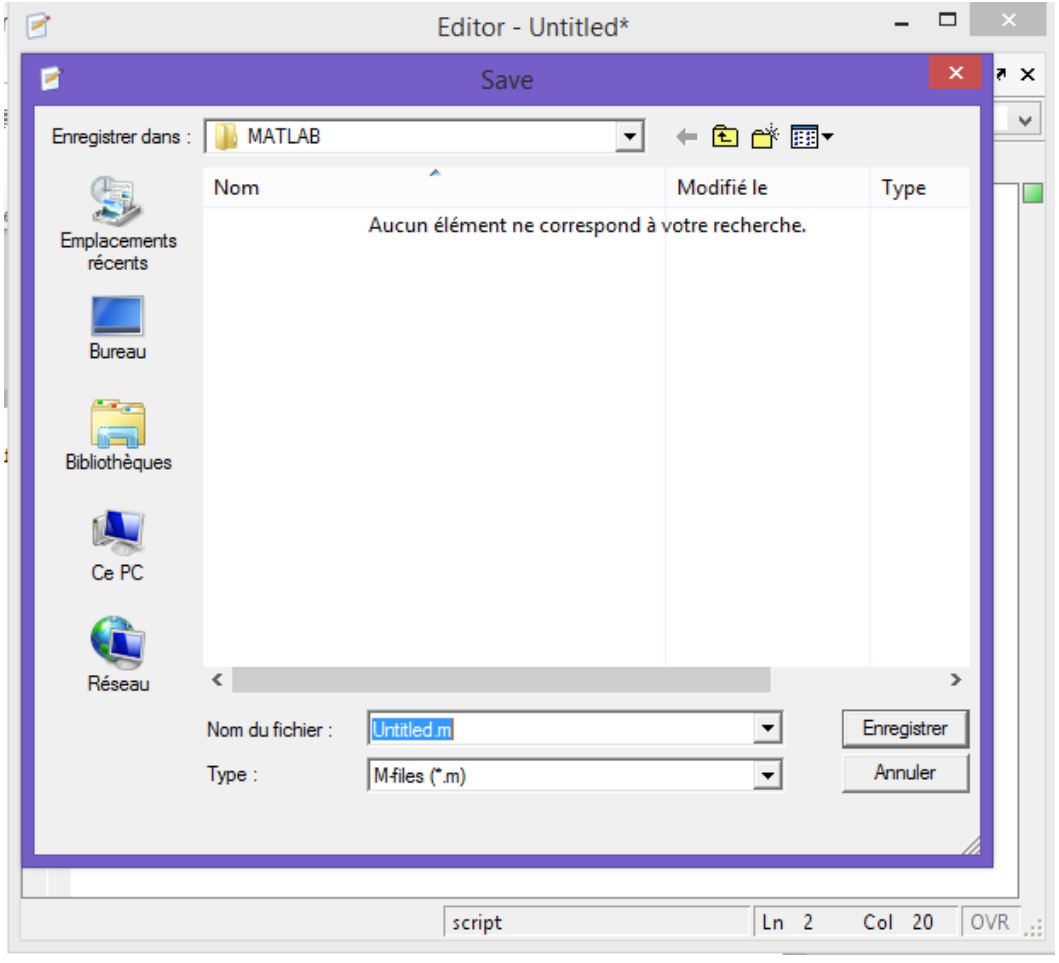
disp est une fonction qui affiche des messages ou des variables

Ce type de programme est appelé script

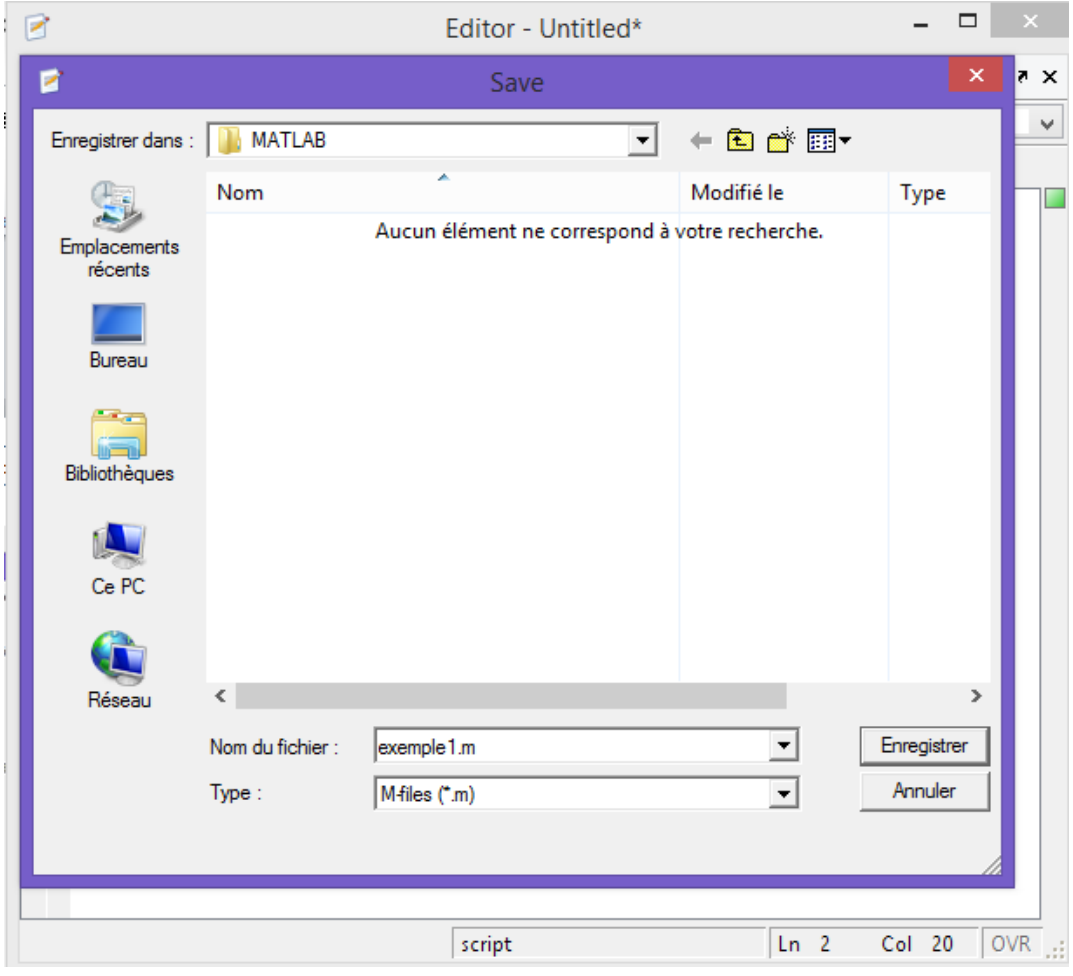
Maintenant le M-file va être sauvegarder en utilisant menu File →save



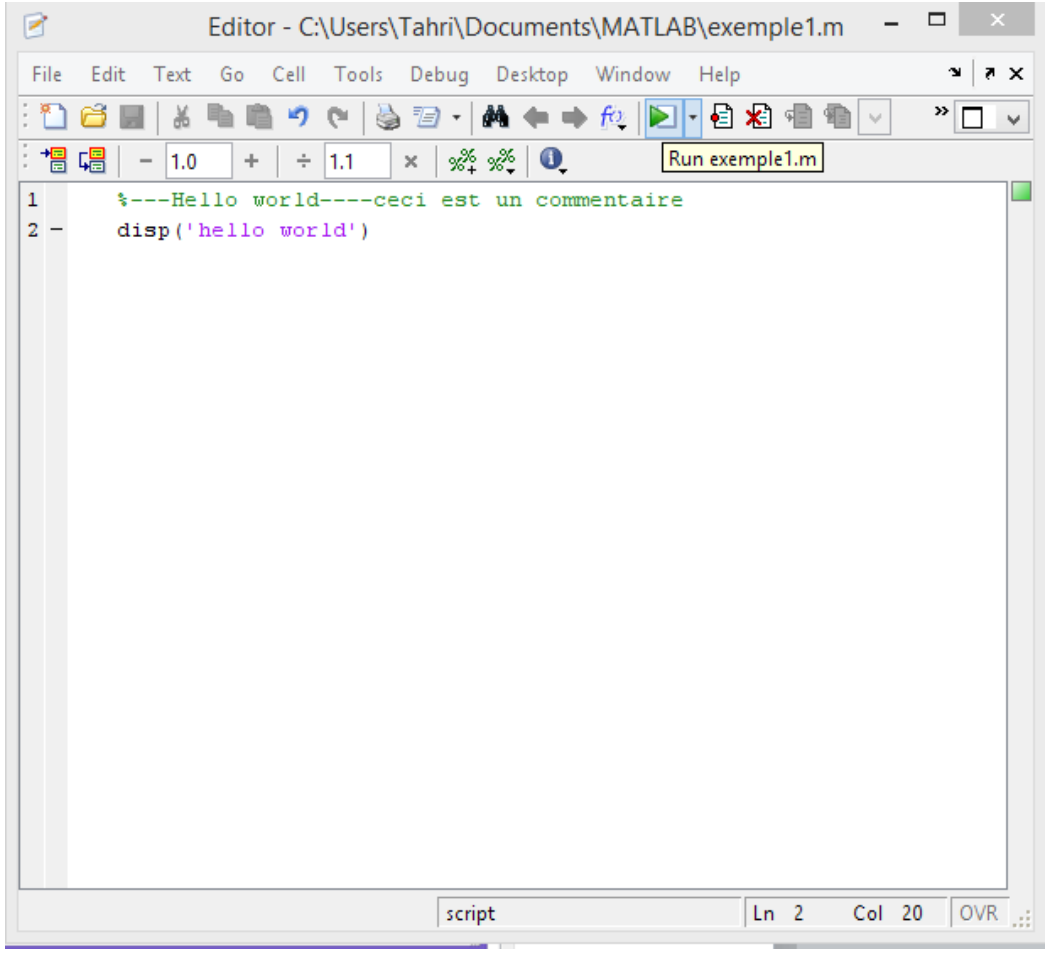
Il faut donner un nom à votre programme



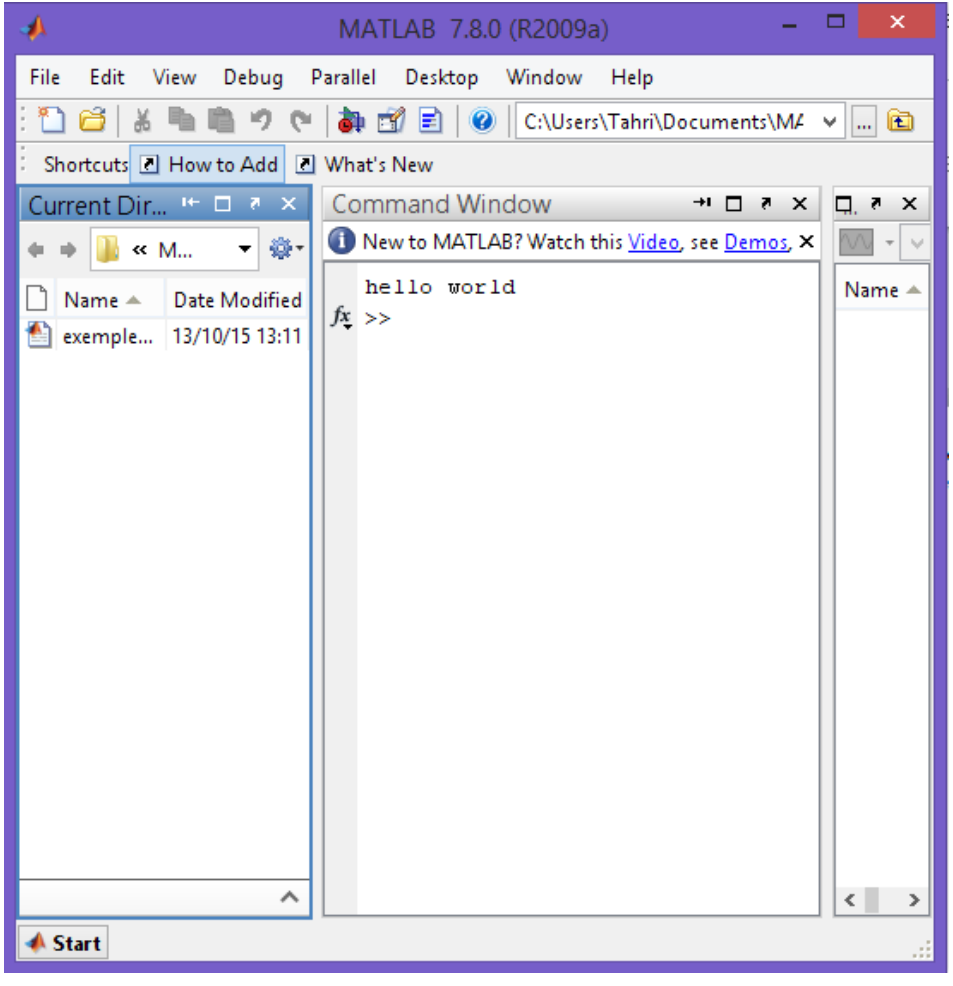
On va donner exemple1.m comme nom au programme.
Et vous cliquer sur enregistrer



Pour exécuter le programme, il suffit de cliquer sur le bouton vert (Run exemple1.m) comme sur la figure suivante.



Ainsi sur l'écran du workspace sera afficher le message



2. Les opérateurs logiques

```
% — Opérateurs logiques ———————————————————————————— %

<          ... est plus petit que ...
>          ... est plus grand que ...
<=         ... est plus petit ou égal à ...
>=         ... est plus grand ou égal à ...
==         ... est égal à ...
~=         ... n'est pas égal à ...
&          ... est vrai et ... aussi (pour tableaux)
|          ... est vrai ou ... est vrai, ou les deux (pour tableaux)
&&         ... est vrai et ... aussi
||         ... est vrai ou ... est vrai, ou les deux
~          ... n'est pas vrai
xor(x,y)   ... est vrai ou ... est vrai
any(x)     ... vrai si un des éléments de x est non nul
all(x)     ... vrai si tous les éléments de x sont nuls
```

3. Les mots gardés

Les mots ou caractères suivants ont une signification particulière dans le langage Matlab.

% — Mots/symboles gardés ———————————————————— %

:	Create vectors, subscript arrays, specify for-loop iterations
( )	Pass function arguments, prioritize operators
[ ]	Construct array, concatenate elements, specify multiple outputs from function
{ }	Construct cell array, index into cell array
.	Insert decimal point, define structure field, reference methods of object
.( )	Reference dynamic field of structure
..	Reference parent directory
...	Continue statement to next line
,	Separate rows of array, separate function input/output arguments, separate commands
;	Separate columns of array, suppress output from current command
%	Insert comment line into code
% %	Insert block of comments into code
!	Issue command to operating system
' '	Construct character array
@	Construct function handle, reference class directory

4. Entrées / Sorties

4.1 Utilisateurs

Les commandes suivantes permettent l'interaction avec l'utilisateur lors de l'exécution d'un script. L'utilisateur peut dans l'exemple ci-dessous saisir un nombre que le script pourra alors utiliser. Dans le sens inverse la commande disp permet d'afficher des variables à l'utilisateur.

```
% — Exemple 1 ————————————————————————————————————- %
n = input('Saisissez un nombre :');     Saisie de l'utilisateur
disp(n);                                Sortie vers l'affichage Matlab
```

4.2 Disques

Il est aussi possible d'exporter ou d'importer sur le disque dur via les deux commandes suivantes.

```
>> save('nomdufichier','variables')

>> save('nomdufichier','variable1','variable2')
```

Exporter vers le disque dur

Exporter du workspace vers le disque dur.

```
>> load('nomdufichier')

>> load('nomdufichier','variable1','variable2')
```

Importer du disque dur

Importer du disque dur vers le workspace.

5. Le contrôle de l'exécution

5.1 Boucle FOR

La boucle FOR permet d'effectuer des opérations pour un nombre d'itérations définis.

L'avantage de la boucle FOR sur la boucle WHILE est sa simplicité d'écriture dans le cas d'un nombre d'itérations définis et bien connu à l'avance (par exemple, le parcours d'un tableau).

Autre remarque, lors de l'exécution de la boucle FOR, la variable qui sert à boucler est accessible en lecture et en écriture . Il est donc possible de réduire ou d'augmenter le nombre d'itérations au cours de l'exécution de la boucle.

```
for i=1:n
  séquence d'instructions
  end
```

Avec pas positif égale à 1

```
for i=n:-0.2:1
  séquence d'instructions
  end
```

Avec pas négatif égale à -0.2

```
 for i=1:m
   for j=1:n
  séquence d'instructions
   end
   end
```

Deux boucle imbriquées

```
% — Exemple 1 ——————————————————————————————— %

for n = 1 :5                % Boucle pour n allant de 1 à 5 inclus par pas de 1
  disp(n)                   % Affichage
end                         % Fin de boucle

% — Exemple 2 ——————————————————————————————— %

for n = 8 :-2 :0            % Boucle pour n allant de 8 à 0 par pas de 2
  disp(n)                   % Affichage
end                         % Fin de boucle

% — Exemple 3 ——————————————————————————————— %

for n = [ 1 9 3 5 6 7 ]     % Boucle pour n égal chaque valeur du vecteur
  disp(n)                   % Affichage
end                         % Fin de boucle
```

5.2 Boucle WHILE

La boucle WHILE permet d'effectuer des opérations de manière répétée jusqu'à ce qu'une condition soit falsifiée (par exemple : tant que la solution n'est pas précise à 4 décimales, continuer à chercher une solutions plus précise.).

```
while expression logique

séquence d'instructions

end
```

```
% — Exemple 1 ———————————————— %

n = 5;
m = 8;
while (n < 10 && m > 0)     % Boucle tant que n est inférieur à 10
                            % et m est supérieur à 0
  n = n + 1;                % Incrémente n
  m = m - 1;                % Décrémente m
  disp(n);                  % Affichage
  disp(m);                  % Affichage
end                         % Fin de boucle
```

5.3 Instruction de choix IF

L'instruction IF est une instruction de choix. Autrement dit, en fonction que son gardien (expression logique) sera évalué vrai ou faux, la commande exécutera un groupe d'instruction ou l'autre.

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

```
% — Exemple 1 ———————————————— %

n = 5;
m = 8;
if (n > 0)                      % Gardien 1

  if ( n > 5 && m < 0)          % Gardien 1 et gardien 2
    disp ('ici 1')              % Affichage
  elseif (n == 5)               % Si gardien1 et non gardien2 et gardien 3
    disp ('ici 2')              % Affichage
  else                          % Si gardien1 et non gardien2 et non gardien 3
    disp('ici 3')               % Affichage
  end                           %

else                            % Si gardien 1 est faux
  disp('ici 4')                 % Affichage
end                             %
```

5.4 Instruction de choix SWITCH

L'instruction SWITCH est une instruction de choix comme le IF mais avec la particularité de pouvoir effectuer plus de branchements que le IF.

La commande SWITCH doit être utilisée dans le cas ou, par exemple, en fonction de la valeur d'une variable, on effectue différentes opérations. Attention toute fois, que le nombre de valeurs possibles de cette variable doit être restreint pour conserver une certaine lisibilité du code.

Le mot clé break signifie que l'on arrête la commande SWITCH et que l'on transfert l'exécution au mot clé end.

```
switch var

case cst1,

séquence d'instructions 1

case cst2,

séquence d'instructions 2

...

case cstN,

séquence d'instructions N

otherwise

séquence d'instructions par défaut

end
```

où

- var est une variable numérique ou une variable chaîne de caractères;
- cst_1, ..., cst_N, sont des constantes numérique ou des constantes chaîne de caractères;
- *séquence d'instructions i* est la séquence d'instructions à exécuter si le contenu de la variable var est égal à la constante cst_i ($var==cst_i$).

```
% — Exemple 1 ———————————————— %

n = input('Pour sauver jack tapez 1,      % Demande de saisie de l'utilisateur
Jessy tapez 2, Brian tapez 3 :');

switch n

  case 1                                  % Si n vaut 1
    disp('Jack est sauvé')
  break

  case 2
    disp('Jessy est sauvé')               % Si n vaut 2
  break

  case 3
    disp('Brian est sauvé')               % Si n vaut 3
  break

  otherwise
    disp('Vous n"avez sauvé personne')    % Si n est différent de 1,2,3

end
```

Merci

Pour

Votre Attention