

# **Cours Outils de programmation pour les mathématiques**

**Cours : Outils de programmation pour les mathématiques**

**Crédits : 2**

**Coefficient : 1**

**Objectifs de l'enseignement : Maîtrise de logiciels scientifiques.**

**Connaissances préalables recommandées : Notions de programmation**

**Table de matière :**

- I. Chapitre1 : Introduction à MATLAB**
  - 1. Présentation de l'environnement MATLAB**
  - 2. Manipulation des variables**
  - 3. Les vecteurs et les matrices**
- II. Chapitre2 : Programmation avec MATLAB**
  - 1. Généralités**
  - 2. Les structures de contrôles**
  - 3. Les fonctions**
- III. Chapitre3 : Représentation graphique**
  - 1. Graphiques 2D**
  - 2. Graphiques 3D**
- IV. Chapitre4 : GUI Interfaces Graphiques.**

## I. Introduction à MATLAB

MATlab« MATrix LABoratory »est un langage de haut niveau pour la programmation scientifique basé sur le calcul matriciel (les variables manipulées sont des matrices), il est développé depuis 1984 par The MathWorks Company (<http://www.mathworks.com/>). Son noyau est composé de bibliothèques écrites au début en Fortran puis en C++.

L'objectif de MATLAB est de fournir aux chercheurs et ingénieurs un environnement de calcul numérique à la fois simple à utiliser et efficace.

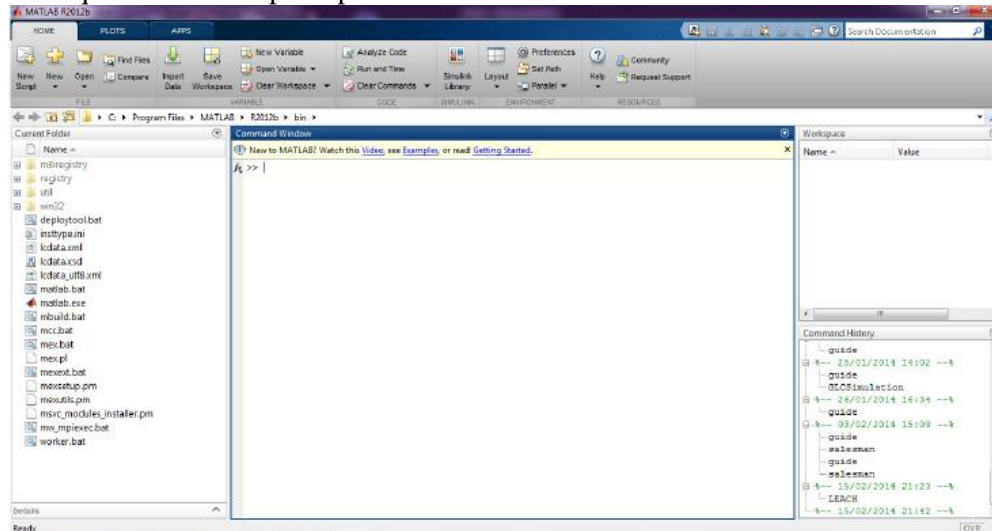
MATLAB permet le travail interactif soit en mode commande (interactif), soit en mode programmation (exécutif).

MATLAB possède les particularités par rapport à d'autres langages, il permet :

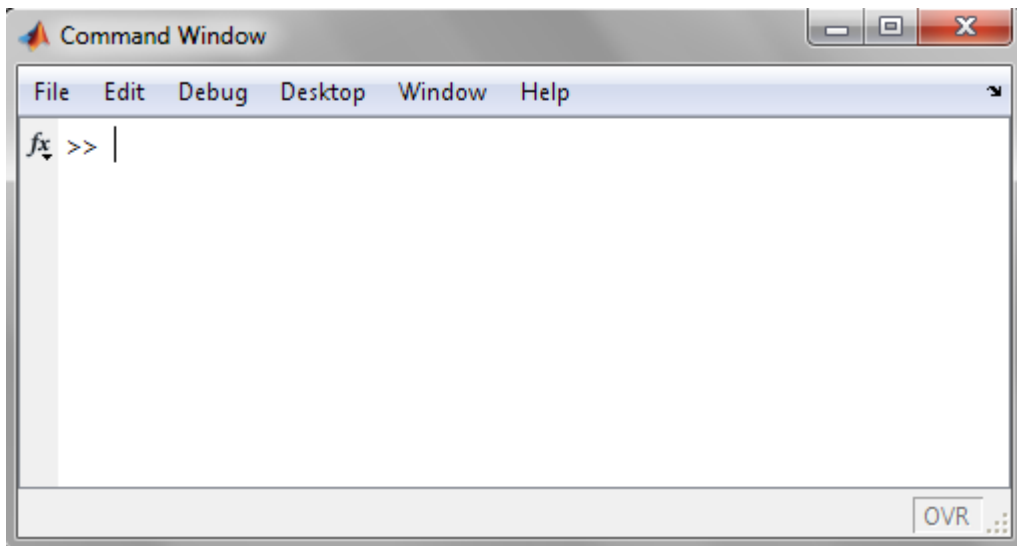
- le calcul numérique et le traitement du signal.
- le tracé de graphiques (visualisations graphiques).
- la programmation facile.
- la possibilité de liaison avec les autres langages classiques de programmations (l'interfaçage avec C ou Fortran).
- le développement avec l'outil graphique qui inclut les fonctions d'interface graphique et les utilitaires (Graphical User Interface « GUI »).
- l'utilisation des boîtes à outils (Toolboxes) spécialisées très complètes pour résoudre des catégories spécifiques de problèmes.
- L'intérêt de MatLab tient, d'une part, à sa simplicité d'utilisation :  
Pas de compilation (logiciel interprété nécessite une grande mémoire et un temps de calcul très long), Déclaration implicite des variables.

### 1. Présentation de l'environnement MATLAB

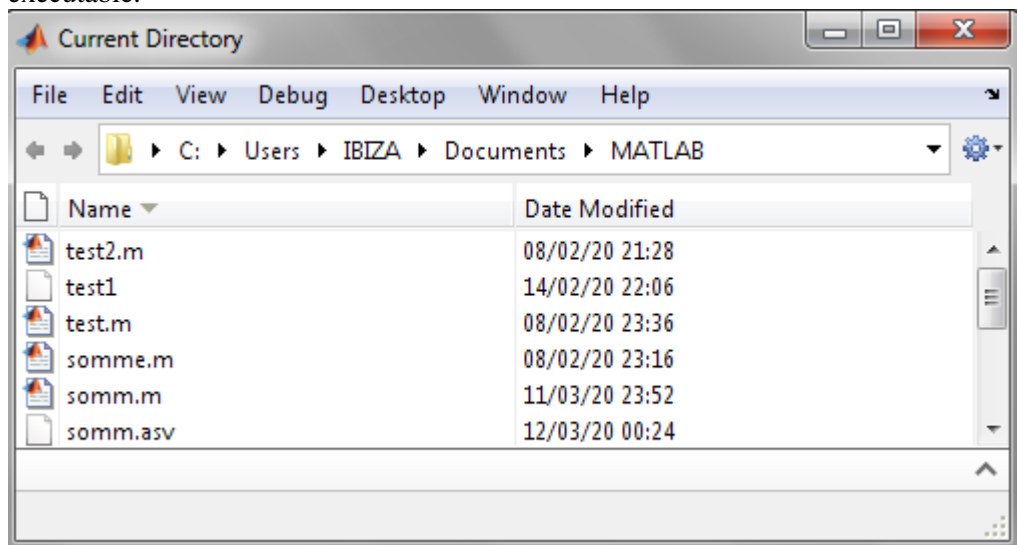
Selon la version utilisée, l'interface peut changer légèrement mais les points centraux resteront identiques. La fenêtre principale de MatLab est divisée en 4 sous-fenêtres :



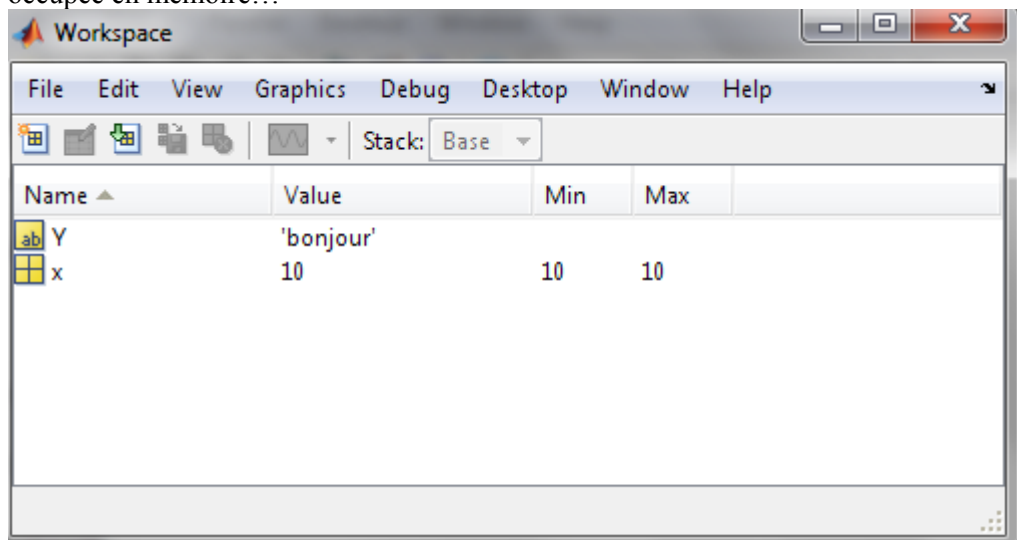
- **Command window** (console d'exécution):
  - il s'agit de la fenêtre la plus importante, elle permet à l'utilisateur de saisir directement les commandes (instructions) à exécutées.
  - **Prompt** « >> » : le curseur indique que MATLAB est en attente d'une commande.
  - Toutes les commandes sont en minuscule et en anglais
  - Lorsque l'on entre une commande, MATLAB affiche systématiquement le résultat de cette commande



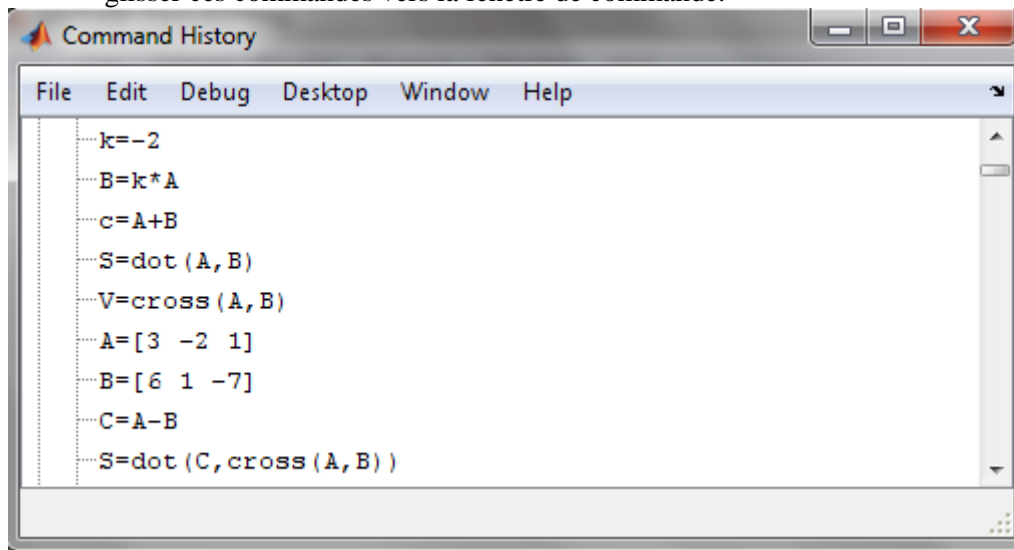
- **Current directory** (répertoire courant) : Permet de naviguer et de visualiser le contenu du répertoire courant de l'utilisateur. Les programmes de l'utilisateur doivent être situés dans ce répertoire pour être visible et donc exécutable.



- **Workspace** (espace de travail) : permet de visualiser les variables définies, leur type, la taille occupée en mémoire...



- **Command history** : historique des commandes que l'utilisateur a exécutées. Il est possible de faire glisser ces commandes vers la fenêtre de commande.



**Remarque** : il est possible d'affichage ou de masquer l'une des fenêtres vues précédemment en utilisant le menu **DESKTOP** puis en cochant / décochant les fenêtres à afficher/masquer.

## 2. Manipulation des variables

- MATLAB gère les nombres entiers, réels, complexes, les chaînes de caractères .
- Il est inutile de déclarer préalablement le type de la variable que l'on manipule, il suffit simplement d'assigner une valeur au nom de la variable avec l'instruction `<< = >>`.  

```
>> a=10
a=
10
```
- Toutes variable en MATLAB quelque soit son type est considérée comme matrice :
- Les scalaires sont des matrices 1 x 1.
- les vecteurs lignes sont des matrices 1 x n.
- les vecteurs colonnes sont des matrices n x 1.
- Lorsqu'on n'utilise pas des variables, le résultat de la commande est automatiquement affecté à la variable `ans` qui peut être par la suite utilisée comme une variable normale :  

```
>> 2+7
ans=9
```

**Exemple** : Définir le type de chaque variable :

```
>> a = 1.3; b = 3+i; c = 'bonjour'; x=4;
>> d1 = (a==x); d2 = logical(1);
>> e = int8(2);
```

`a` représente un réel, `b` un complexe, `c` une chaîne de caractères, `d1` et `d2` sont deux manières de définir une variable logique et `e` est un entier codé sur 8 bits. On peut alors vérifier le type de ces différentes variables en utilisant la fonction **whos** :

**L'identifiant d'une variable** : une variable est désignée par un identificateur qui doit respecter les conditions suivantes :

- un identificateur débute nécessairement par une lettre, éventuellement suivie de lettres, de chiffres ou du caractère souligné (`_`) ;
- sa longueur est inférieure ou égale à 31 caractères ;
- dans les identificateurs, les majuscules sont distinguées des minuscules (on dit qu'ils sont case-sensitive).

- Un identificateur ne doit pas être un mot clés MATLAB

**Variables prédéfinies :** Voici quelques identificateurs prédéfinis :

- **ans** désigne le résultat de la dernière évaluation ;
- **pi** qui est la constante mathématique 3,1415926535897 ;
- **eps** qui est la précision numérique relative des calculs en virgule flottante ( $2^{-52} \approx 2.2204e-16$ ) ;
- **inf** qui représente l'infini où le résultat d'une expression excède **realmax**. (au sens d'une évaluation du type  $(1/0)$ ) ;
- **NaN** qui représente Not-a-Number où le résultat d'une opération non-définie (calcul qui n'est pas numérique) comme  $0/0$  .
- **i,j** qui sont la racine carrée de -1 (le nombre imaginaire)
- attention à ne pas utiliser i et j comme indices pour accéder aux éléments d'un tableau ;
- **realmin** désigne le petit nombre réel positif ;
- **realmax** désigne le plus grand nombre réel positif.

```
>> pi
ans =
    3.1416

>> eps
ans =
    2.2204e-016

>> i
ans =
    0 + 1.0000i

>> j
ans =
    0 + 1.0000i

>> realmin
ans =
```

### Variables d'environnement

• MATLAB garde en mémoire les variables qui ont été créées. On les voit en haut, à gauche, lorsque MATLAB dispose d'une interface graphique. On outre, on peut les afficher et les effacer par la ligne de commande :

- **who** donne la liste des variables présentes dans l'espace de travail.
- **whos** donne la liste des variables présentes dans l'espace de travail ainsi que leurs propriétés.
- **what** donne la liste des fichiers (.m) et (.mat) présents dans le répertoire courant.
- **clear var1... varn** efface les variables **var1...varn** de l'espace de travail.
- **clear** efface toutes les variables créées dans l'espace de travail.
- **exist var** vérifie si une fonction ou une variable existe dans le workspace.
- Les commandes **save** et **load** permettent d'écrire (charger) toutes les variables du workspace dans le fichier matlab.mat. Si un nom de fichier est spécifié (save myfile ou load myfile) l'espace de travail est sauvegardé (chargé) conformément.

**Les opérations de base sur les variables :**

- **a + b** addition
- **a – b** soustraction
- **a / b** division
- **a \* b** multiplication
- **a ^b** ou **power(a,b)** mettre **a** à la puissance de **b**
- **mod(a, b)** le reste de la division entière de a sur b
- **abs(a)** la valeur absolue
- **ceil(a)** la valeur entière supérieure
- **floor(a)** la valeur entière inférieure
- **round(a)** la valeur entière la plus proche de a
- **fix(a)** la valeur entière
- **a & b** ou **and(a,b)** et logique
- **a | b** ou **or(a,b)** ou logique
- **~a** ou **not(a)** négation logique
- **xor(a,b)** ou exclusif logique
- **false** valeur logique de faux
- **true** valeur logique de vrai
- **a == b** égalité
- **a ~= b** inégalité
- **a > b** supérieur
- **a >= b** supérieur ou égal
- **a < b** inférieur
- **a <= b** inférieur ou égal
- **log(a)** logarithme naturel
- **log2(a)** logarithme de la base 2
- **log10(a)** logarithme de la base 10
- **conj(c)** le conjugué d'un nombre complexe
- **real(c)** la partie réelle d'un nombre complexe
- **imag(c)** la partie imaginaire d'un nombre complexe
- **angle(c)** l'argument d'un nombre complexe
- **abs(c)** le module d'un nombre complexe

**Exercice récapitulatif :** Créer une variable x et donnez-la la valeur 2, puis écrivez les expressions suivantes :

•  $3x^3 - 2x^2 + 4x$

•  $\frac{e^{1+x}}{1-\sqrt{2x}}$

•  $|\sin^{-1}(2x)|$

**La solution**

```
>> x=2 ;
```

```
>> 3*x^3-2*x^2+4*x ;
```

```
>> exp(1+x)/(1-sqrt(2*x)) ;
```

```
>> abs(asin(2*x)) ;
```

En MATLAB une variable logique peut prendre les valeurs 1(vrai) ou 0 (faux) avec une petite règle qui admette que :

- 1) Toute valeur égale à 0 sera considérée comme fausse ( $= 0 \Rightarrow$  faux)
- 2) Toute valeur différente de 1 sera considérée comme vrai ( $\neq 0 \Rightarrow$  vrai).

**La commande format :** par défaut, MATLAB affiche les valeurs numériques réelles sous format de point fixe à 5 chiffres.

On peut changer la façon dont les valeurs numériques sont affichées a comme suit:

- **format short** : point fixe, 5 chiffres.
- **format long** : point fixe, 15 chiffres.
- **format short e** : point flottant, 5 chiffres.
- **format long e** : point flottant, 15 chiffres.
- **format rational** : format rationnel.

### 3. Les vecteurs et les matrices

## II. Programmation avec MatLab

### 1. Généralités

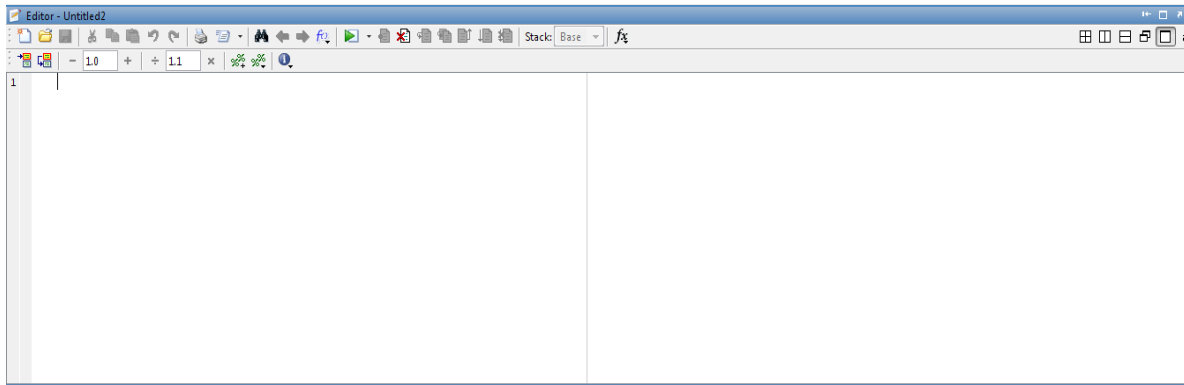
**Exemple d'un programme :**

```
a=3;  
b=6;  
c=input('donner la valeur de c : ');  
d=a+b/c;  
disp(d),
```

Donc : ni entête ni déclarations de types, de constantes ou de variables. Toutes les ressources de Matlab sont directement disponible sur l'espace de travail et n'ont pas besoin d'être chargée via des instructions telles que include de C ou uses de Pascal. (le type et la dimension d'une variable est donc géré automatiquement d'après affectation).

**L'écriture d'un programme en MATLAB:**

- Si nous avons un nombre très réduit d'instructions à exécuter, il est en effet possible de les écrire directement dans la fenêtre de commande.
- Si nous avons des programmes longs et surtout de sauvegarder ces programmes. L'éditeur de programmes ou de fichiers d'extension .m est fait pour cela.
- Un script MATLAB est composé d'une suite d'instructions ou de commandes, toutes séparées par une virgule (,), un point-virgule (;) ou un retour à la ligne.
- La commande **edit** aussi fait appel à l'éditeur de MATLAB en créant un fichier du nom spécifié (**edit** monfichier.m).
- L'éditeur de MATLAB crée par défaut un nouveau script sous le nom 'untitled.m'. Le programmeur peut spécifier ultérieurement un nom de son choix en utilisant le bouton (**Save**).
- Généralement, le script est sauvegardé dans un fichier avec l'extension '.m' sous le dossier courant. On peut aussi charger un fichier déjà édité en utilisant le bouton (Open).



### L'exécution d'un programme en MATLAB:

- L'exécution du script se fait à travers l'invite de commande en spécifiant juste le nom du script.
- L'exécution peut être effectuée directement depuis l'éditeur en cliquant sur le bouton (**Run**).
- Néanmoins, les sorties des scripts (affichage des résultats) se fait toujours sur la fenêtre des commandes.

## 2. Les structures de contrôles

### a. 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. En MATLAB un commentaire commence par le symbole **%** et occupe le reste de la ligne. Par exemple :

```
>> A=B+C ; % Donner à A la valeur de B+C
```

### b. Instructions de lecture \ Ecriture

#### ▪ Lecture des données dans un programme (Les entrées)

Pour lire une valeur donnée par l'utilisateur, il est possible d'utiliser la commande **input**, qui a la syntaxe suivante :

**Variable = input ('une phrase indicative')**

La valeur déposée par l'utilisateur sera mise dans cette variable, une phrase indicative = Une phrase aidant l'utilisateur à savoir quoi entrer

**Par exemple :**

```
>> A = input ('Entrez un nombre entier : ')
```

```
Entrez un nombre entier : 5
```

```
A =
```

```
5
```

#### ▪ Ecriture des données dans un programme (Les sorties)

- On a déjà vu que MATLAB peut afficher la valeur d'une variable en tapant seulement le nom de cette dernière.

**Exemple :**

```
>>A=12
```

```
>>A
```

```
A=
```

```
5
```

- On peut utiliser la fonction **disp**, et qui a la syntaxe suivante : **disp (objet)**

La valeur de l'objet peut être un nombre, un vecteur, une matrice, une chaîne de caractères ou une expression.

**Exemple :**

```
>> A=5
```

```
>> disp(A)
```

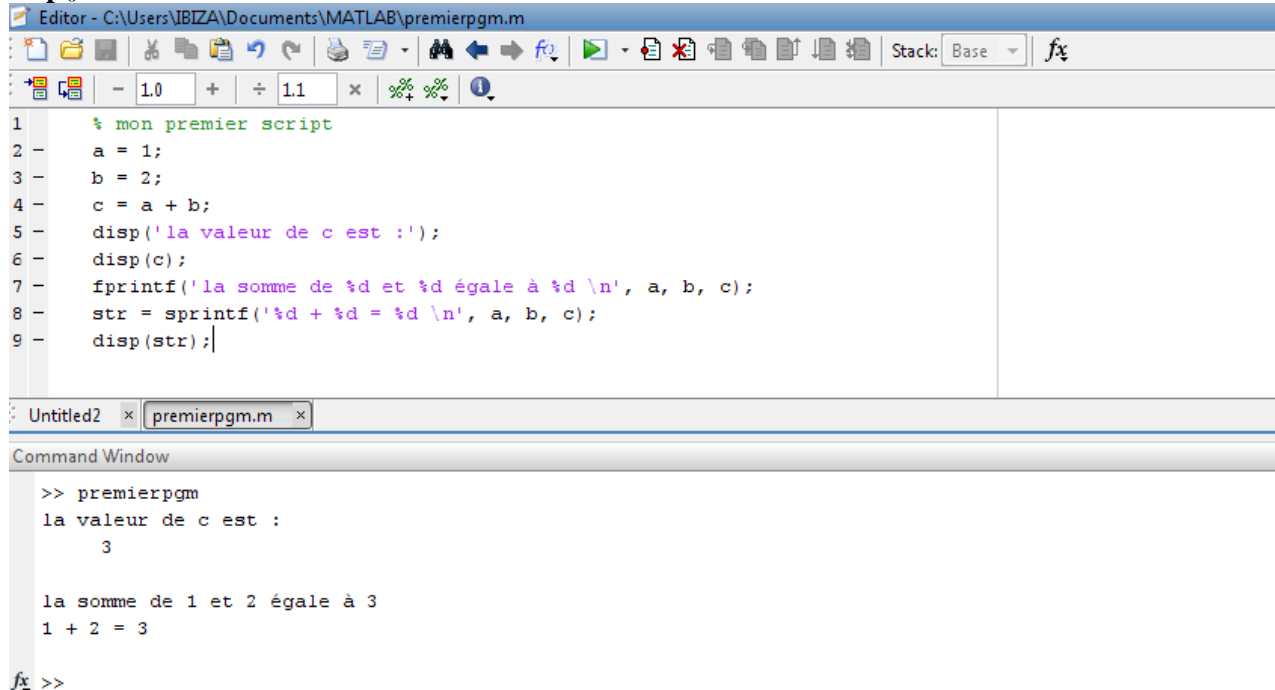


5

- La commande **fprintf()** accepte tous genre de variable en utilisant leurs formats de types : **%d** pour les entiers, **%f** pour les reels, **%s** pour les chaines de caractères, **%c** pour le caracteres, ... . On peut ajouter un retour à la ligne via **'\n'** et une tabulation via **'\t'** .

• La commande **sprintf()** sauvegarde tous genre de variables de la même manière que **fprintf**, mais sans affichage sur la fenêtre des commandes. **sprintf()** retourne une chaine de caractères qui peut être affichée ultérieurement en utilisant la commande

**disp()**.



The screenshot shows the MATLAB Editor window with a script named 'premierpgm.m'. The script contains the following code:

```

1 % mon premier script
2 a = 1;
3 b = 2;
4 c = a + b;
5 disp('la valeur de c est :');
6 disp(c);
7 fprintf('la somme de %d et %d égale à %d \n', a, b, c);
8 str = sprintf('%d + %d = %d \n', a, b, c);
9 disp(str);

```

The Command Window shows the output of the script:

```

>> premierpgm
la valeur de c est :
    3

la somme de 1 et 2 égale à 3
1 + 2 = 3

```

### C. Instruction conditionnelle **If**

#### Syntaxe

<b>if</b> (condition)		<b>if</b> (condition)
Instruction1	ou bien	ensemble d'instructions1
Instruction2		<b>else</b>
....		ensemble d'instructions2
InstructionN		<b>end</b>
<b>end</b>		

Si la condition est évaluée à vrai, les instructions entre le **if** et le **end** seront exécutées, sinon elles ne seront pas (ou si un **else** existe les instructions entre le **else** et le **end** seront exécutées). 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 (expression1)
    Ensemble d'instructions1
elseif (expression2)
    Ensemble d'instructions 2
....
elseif (expressionN)
    Ensemble d'instructions N
else
    Ensemble d'instructions si toutes les expressions étaient fausses
end

```

#### Exemple :

```
>> age = input('Entrez votre âge : ');
```

```

if (age < 2)
disp('Vous êtes un bébé')
elseif (age < 13)
disp('Vous êtes un enfant')
elseif (age < 18)
disp('Vous êtes un adolescent')
elseif (age < 60)
disp('Vous êtes un adulte')
else
disp('Vous êtes un vieillard')
end

```

#### d. L'instruction switch

L'instruction switch exécute des groupes d'instructions selon la valeur d'une variable ou d'une expression

**Syntaxe :**

```

switch (expression)
  case valeur_1
    ensemble d'instructions 1
  case valeur_2
    ensemble d'instructions 2
  ...
  case valeur_n
    ensemble d'instructions n
  otherwise
    ensemble d'instructions si tous les case ont échoué
end

```

**Exemple :**

```

x = input ('Entrez un nombre : ');
switch(x)
  case 0
    disp('x = 0 ')
  case 10
    disp('x = 10 ')
  case 100
    disp('x = 100 ')
  otherwise
    disp('x n'est pas 0 ou 10 ou 100 ')
end

```

#### e. L'instruction for

L'instruction **For** répète l'exécution d'un groupe d'instructions un nombre déterminé de fois.

Elle a la forme générale suivante :

```

          for variable = expression
          ensemble d'instructions
          end
for compteur = valeur_initiale : increment : valeur_finale
instructions
.....
          end

```

L'expression correspond à la définition d'un vecteur : **début : pas : fin** ou **début : fin**

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

**Exemple :**

```

for i = 1 : 4   résultat  2
  j=i*2 ;      4
end

```

```

disp(j)           6
end               8
for i = 1 : 2: 4  résultat
j=i*2 ;          2
disp(j)          6
end
for i = [1 , 4 , 7 ] résultat
j=i*2 ;          2
disp(j)          8
end              14

```

#### f. L'instruction while

L'instruction **while** répète l'exécution d'un groupe d'instructions un nombre indéterminé de fois selon la valeur d'une condition logique. Elle a la forme générale suivante :

```

while (condition)
Ensemble d'instructions
end

```

#### Exemple :

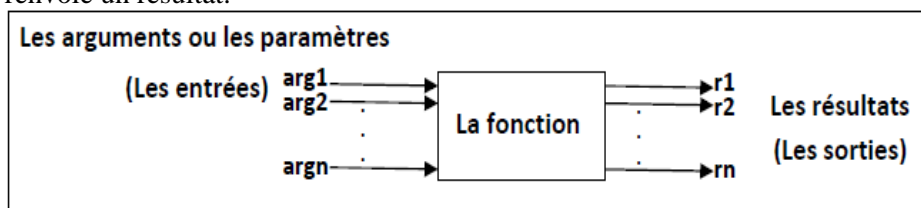
```

a=1 ;
while (a~=0)
a = input ('Entrez un nombre (0 pour terminer) : ');
end

```

### 3. Les fonctions

une fonction est une routine (un sous programme) qui accepte des arguments (des paramètres) et qui renvoie un résultat.



- Les fonctions peuvent être déclarées directement dans un fichier Script qui doit obligatoirement porter le même nom de la fonction à déclarer
- Une fonction est déclarée comme suit :
  - **function** nomdefonction(arg1, arg2,...) % une fonction sans paramètre de sortie.
  - **function** r = nomdefonction(arg1, arg 2,...) % une fonction avec un seul paramètre de sortie.
  - **function** [r1, r2, ...] = nomdefonction(arg1, arg 2,...)% une fonction avec plusieurs paramètres de sortie.
- Il faut noter que : La fonction ne se termine pas par **end** comme dans les autres langages évolués.
- Le point-virgule n'est pas nécessaire a la fin des lignes, sauf si un affichage du résultat de la ligne en question est requis.
- Les variables d'entrée de la fonction sont passés par valeur : On ne peut pas modifier leurs valeurs.
- Les variables de sortie de la fonction sont passés par résultat : On doit leur affecter une valeur.
- Les variables locales sont des variables temporaires a utilisation locale a l'intérieur de la fonction : On ne peut pas les utiliser à l'extérieur.

**III. Chapitre3 : Représentation graphique**

**1. Graphiques 2D**

**2. Graphiques 3D**

**IV. Chapitre4 : GUI Interfaces Graphiques.**