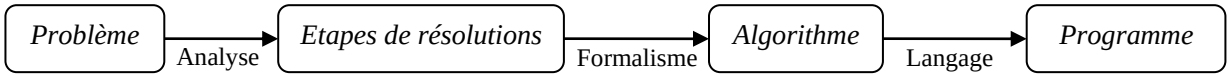
# Introduction au langage Matlab

## ***Rappel des notions algorithmiques***

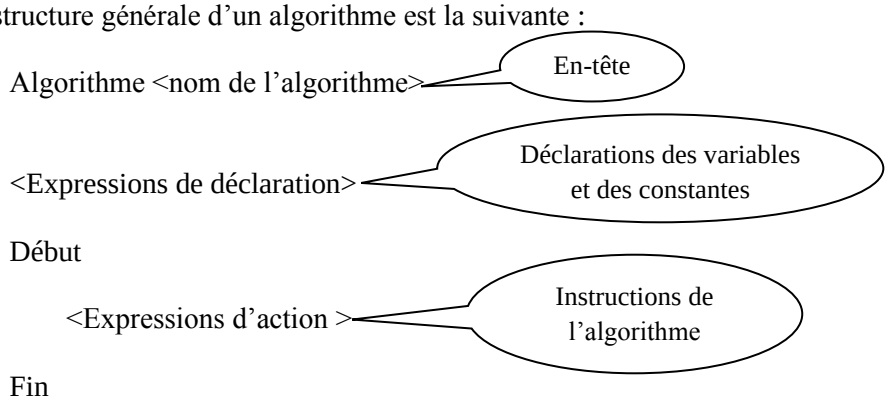
La figure suivante, présente les étapes à suivre pour passer d'un problème à un programme. Ce dernier sera exécuté par un ordinateur, afin de traiter des données et fournir un ou plusieurs résultats.




*Etapes de résolutions d'un problème par un ordinateur*

### Structure d'un algorithme

La structure générale d'un algorithme est la suivante :




Le tableau suivant résume les instructions utilisées dans le langage algorithmique.

| *Instruction* | *Signification* | *Syntaxe* |
|---|---|---|
| Affectation | L'affectation consiste à placer une valeur dans une variable (ce qui revient à changer le contenu de cette variable). | *Identificateur_ de_ variable ← Expression*<br>***Exemples***<br>☺ *B1 ← 5*<br>☺ *a ← (b * c)/2* |
| Traitements conditionnelles | Les traitements conditionnels permettent d'exécuter des instructions différentes en fonction de certaines conditions. Une condition (expression conditionnelle ou logique) est évaluée. Si elle est vraie, un traitement est réalisé. | ***si*** *(condition)* ***alors*** *instruction 1*<br>*[**sinon** instruction 2]*<br>***finsi***<br>***Exemples***<br>☺ ***si*** *a < 10* ***alors***<br>*a ← (a*10)/d*<br>***finsi***<br>☺ ***si*** *a < 10* ***alors***<br>*a ← (a*10)/d*<br>***sinon***<br>*a ← (a/10)*d*<br>***finsi*** |
| | | ***selon*** *(selecteur)*<br>*valeur 1 : instruction 1*<br>*……*<br>*valeur N : instruction N*<br>***autrement*** *: instruction*<br>***finselon***<br>***Exemple***<br>***selon*** *c*<br>*'a'..'z', 'A'..'Z', '_' : nat := Lettre*<br>*'0'..'9' : nat := chiffre*<br>***autrement*** *: nat := Autre_type*<br>***finselon*** |

| | | |
|---|---|---|
| Traitements répétitives | Permet la répétition de certaines Instructions. | ***pour*** *compteur* ***de*** *valeurdebut* ***à*** *valeurfin* ***faire***<br>*instructions*<br>***finpour***<br>***Exemple***<br>*cumul←0*<br>***pour****a****de****1****à****100****faire***<br>*cumul← cumul * i*<br>***fFinpour*** |
| | | ***tant que*** *(condition)* ***faire***<br>*instructions*<br>***fintantque***<br>***Exemple***<br>*cumul←0*<br>*i←1*<br>***Tant que*** *(i<=100)* ***faire***<br>*cumul← cumul * i*<br>*i←i+1*<br>***fintantque*** |
| | | ***répéter***<br>*instructions*<br>***jusqu'à*** *(condition)*<br>***Exemple***<br>*cumul←0*<br>*i←1*<br>***répéter***<br>*cumul← cumul * i*<br>*i←i+1*<br>***jusqu'à*** *(i>100)* |
| Lecture des données | Permet de communiquer des données au programme. | ***Lire***(variable1 [, variable2,…])<br>***Exemple***<br>☺ ***Lire***(A)<br>☺ ***Lire***(A, b, t[1]) |
| Affichage des résultats | Permet de fournir des résultats sous forme compréhensible pour l'utilisateur. | ***Ecrire*** (expression1 [, expression2, ….])<br>***Exemple***<br>☺ ***Ecrire***('La moyenne de l''étudiant est :', moy)<br>☺ ***Ecrire***(A, b, t[1]) |

## Analyse d'un problème

Avant de commencer l'écriture d'un algorithme ou d'un programme, il est préférable de faire une analyse du problème (en langage naturel). Celle-ci permet la détermination de l'objectif(s) cherché(s), des données à fournir et des résultats attendus. Cette réflexion met en lumière les traitements à effectuer pour l'obtention des résultats.

*Analyse*

- ***Objectifs***: déterminer clairement l'objectif (les objectifs) de l'algorithme ou du programme.
- ***Entrées*** : fournir des données, si nécessaire, au programme ou à l'algorithme.
- ***Sorties***: présenterles résultats attendus de l'algorithme ou du programme.
- ***Traitement*** : décrirel'ensemble des étapes à suivre pour l'obtention de(s) l'objectif(s).

## Exemple

Ecrire un algorithme qui permet d'additionner deux nombres entiers.

## Solution

*Analyse*

*Objectif* : calculer la somme de deux valeurs entières.
*Entrées* : l'algorithme reçoit les deux entiers.
*Sortie* : l'algorithme donne la somme des deux nombres.

*Traitement* :

Pour calculer la somme, l'algorithme pourra suivre les étapes suivantes :

1. Lire la première valeur entière.
2. Lire la deuxième valeur entière.
3. Additionner les deux valeurs.
4. Afficher la somme calculée.

*Algorithme* : l'algorithme décrivant cette solution s'écrit de la manière suivante :

```
Algorithme SommeEntier
Variable
  nombre1, nombre2, somme : entier
Debut
  Lire (nombre1)
  Lire (nombre2)
  somme ← nombre1 + nombre2
  Ecrire ('La somme des deux nombres est :, somme)
Fin
```

## Exercices

1. Ecrire un algorithme qui permet la multiplication de trois nombres réels introduits par l'utilisateur.
2. Ecrire un algorithme qui détermine la moyenne des notes d'un examen pour une classe composé de dix étudiants.
3. Donner l'algorithme qui affiche la valeur maximale d'un ensemble de cinq valeurs réelles.

# *Introduction à l'environnement Matlab*

Matlab (MATrix LABoratory) est un langage de script émulé par un environnement de développement du même nom ; il est utilisé à des fins de calcul numérique. Développé par la société ***The Math Works***. Matlab permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java, et Fortran.

Matlab est utilisé dans de nombreux domaines, tels que le traitement des images et des signaux, les communications, les systèmes de contrôle du secteur industriel, la conception de réseaux intelligents, la robotique, la finance computationnelle…etc.

Le langage Matlab a été conçu par **Cleve Moler** à la fin des années 1970 à partir de deux bibliothèques écrites en **Fortran** : *LINPACK* et *EISPACK*. Matlab a passé par plusieurs versions, à titre d'exemple nous citons quelques-unes : version1.0, version 2, version 3, version 3.5, version 4, version 5.1, version 5.2, version 5.3, version 6.0, version 6.5, version 7.0, version 7.0, version 7.14, version 8.0, version 9.2, …etc.

## L'environnement de travail Matlab

L'environnement de travail Matlab est un environnement de programmation interactif pour le calcul scientifique, la programmation et la visualisation des données. Il est très utilisé dans les domaines d'ingénierie et de recherche scientifique, ainsi qu'aux établissements d'enseignement supérieur.

Après le lancement de l'environnement Matlab, on va voir apparaître une ou plusieurs nouvelles fenêtres sur l'écran. Ceci est dû à la version installée et/ou au choix de l'utilisateur. Parmi les fenêtres les plus utilisées en trouve :

- ☺ L'invite de commande (**Command Window**) au centre de l'environnement.
- ☺ Le contenu de l'espace courant de travail (**workspace**) en haut à droite.
- ☺ La liste des fichiers du répertoire courant (**Current Folder**) à gauche.
- ☺ L'historique des commandes tapées (**Command History**) en bas à droite.

### L'invite de commande (Command Window)

Le moyen le plus simple d'utiliser Matlab, est d'écrire directement des commandes dans l'invite des commandes, juste après le curseur (prompt) >>. Toutes les commandes sont en ***minuscules*** et en ***anglais***.

Lorsque l'on entre une commande, Matlab affiche systématiquement le résultat de cette commande dans cette même fenêtre.

Par exemple, pour **calculer une expression mathématique** il suffit de l'écrire comme ceci :

```
>>100+1445 puis on clique sur la touche Entrer pour voir le résultat
ans =
        1545
```

Si nous voulons qu'une expression soit calculée mais sans afficher le résultat, on ajoute un point-virgule'**;**' à la fin de l'expression comme suit :

```
>> 7+8 ;
>>
```

## *L'affectation*

Matlab gère les nombres entiers, réels, complexes, les chaînes de caractères ainsi que les tableaux de façon transparente. Autrement dit, il est inutile de déclarer préalablement le type de la variable que l'on manipule, même pour les tableaux et les matrices, il suffit simplement d'assigner (affecter) une valeur au nom de la variable avec *l'instruction d'affectation* « **=** ». Par exemple :

```
>> a = 10 ;
>> u = cos(a) ;
>> v = sin(a) ;
>> u^2+v^2
ans =
        1
>> ans + 10
ans =
        11
>>
```

*Remarques*

- ☺ Toutes les variables sont des matrices (a=5 ***variable scalaire (1 × 1)***, b= [4 6] ***vecteur ligne (1 × 2)***, c= [-5; 2] **vecteur colonne (2 × 1)**, d=[2 3; -1 7] ***matrice carrée (2 × 2)***.
- ☺ Le nom d'une variable ne doit contenir que des caractères alphanumériques ou le symbole '_' (underscore), et doit commencer par un alphabet. Nous devons aussi faire attention aux majuscules car le Matlab est sensible à la casse (**A** *et* **a** *sont deux identifiants différents*).
- ☺ Lorsque le calcul d'une opération n'est pas affecté à une variable, Matlab crée de manière automatique une variable ans (answer) qui contient le résultat de l'opération.

Il est possible d'écrire plusieurs expressions dans la même ligne en les faisant séparées par des virgules ou des points virgules. Le tableau ci-contre présente deux exemples.

| ***Exemple 1*** | ***Exemple 2*** |
|---|---|
| >> x=5+6, y=2*5-1, 12-4<br>x =<br>11<br>y =<br>9<br>ans =<br>8 | >> 5+6; 2*5-1, 12-4;<br>ans =<br>9 |

## *Les commentaires*

Les commentaires sont, en *programmation informatique*, des portions du code source ignorées par le compilateur ou l'interpréteur, car destinées en général à un lecteur humain et non censées influencer l'exécution du programme.

Les commentaires sont le plus souvent utilisés pour expliquer le code, et comme pour tout langage de programmation, il est très important de commenter le code au fur et à mesure de sa rédaction - jamais après, car après signifie généralement jamais.

En Matlab, le symbole '%' dans une ligne a pour effet que le reste de la ligne ne sera pas exécuté ;ceci signifie l'insertion d'un commentaire.

*Exemple*

```
>> la = 10 ; lo = 20 ;   % la : largeur, lo : longueur
>> su = la * lo  % surface = largeur * longueur
su =
  200
```

## *Les commandes d'entrée et de sortie*

Matlab fournit plusieurs commandes d'entrée et de sortie. Parmi les quelles on trouve :

***La commande input***

Cette commande permet la saisie d'une valeur depuis le clavier.
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, **var** = input ('**message**','s') affiche **message** et affecte à la variable **var** la valeur entrée au clavier considérée alors comme une chaîne de caractères.

*Exemples*

| ***Exemple 1*** | ***Exemple 2*** |
|---|---|
| >> A=input ('Entrée la valeur de A : ')<br>Entrée la valeur de A : 3<br>A =<br>3 | >> S=input ('Entrée une phrase : ','s')<br>Entrée une phrase : bonjour tout le monde<br>S =<br>bonjour tout le monde |

***La commande disp***

La commande **disp(X)** affiche une variable X (chaine de caractères, entier, …) sur la fenêtre des commandes. Elle n'accepte qu'un seul type de donnée à la fois. Dans le cas échéant, une conversion de type est toujours nécessaire ; par exemple, toutes les variables sont converties à des chaines de caractères.

La commande disp permet d'afficher un tableau sans écrire le nom de la variable, ceci peut améliorer certaines présentations.

*Exemples*

```
>> matrice = [2 3; -1 7] ;
>> disp (matrice)
     2     3
    -1     7
```

On utilise fréquemment la commande **disp** avec un tableau qui est une chaîne de caractères pour afficher un message. Par exemple **disp**('Calcul du déterminant de la matrice '). On utilise également la commande **disp** pour afficher un résultat. Par exemple **disp(['Le déterminant de la matricevaut ', num2str(det(matrice))])**. On remarque que l'usage de la commande **disp** est alors un peu particulier. En effet un tableau doit être d'un type donné, les éléments d'un même tableau ne peuvent donc être des chaînes de caractères et des valeurs numériques. On a donc recours à la commande **num2str** (Number to String) pour convertir une valeur numérique en une chaîne de caractères.

*Remarque*

Si la chaîne de caractères contient une apostrophe il est impératif de doubler l'apostrophe.

*Remarque*

Probablement, on ne connaîtra jamais toutes les commandes de Matlab, mais ce n'est pas un problème, car nous pourrons retrouver toutes les informations nécessaires facilement en se servant de l'aide.

La commande **help** permet de donner un aperçu des commandes disponibles, ou si vous vous rappelez d'une commande mais pas de comment on l'utilise, alors la commande « *help commande* » vous sera utile.

## L'espace courant de travail (Workspace)

Cette fenêtre affiche les variables (avec leurs valeurs, leurs dimensions, …etc.) existant dans l'espace de travail. Ces variables peuvent être créés, importer dans Matlab à partir des fichiers de données ou d'autres programmes.

Par exemple, après l'exécution des commandes présentées précédemment sous le sous-titre *Affectation*, les variables s'affichent comme présenté dans la figure ci-contre.

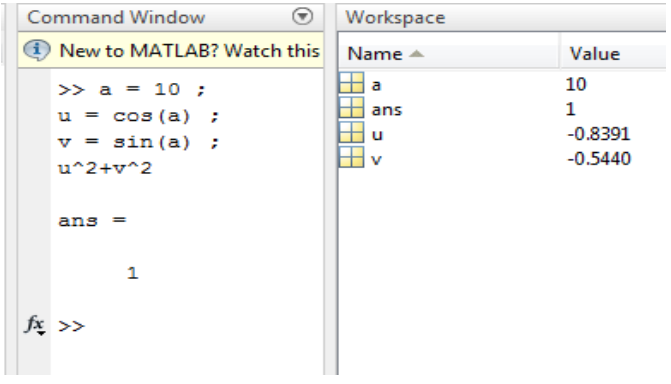



**Figure.1.** Affichage des variables de l'espace de travail

Les variables de l'espace de travail ne sont pas sauvegardées lorsque nous quittons Matlab. Pour une utilisation ultérieure des données, il est possible de les enregistrer à l'aide de la commande **save** :

*save myfile.mat*

L'enregistrement préserve l'espace de travail dans le dossier de travail actuel, dans un fichier compressé avec une extension ***.mat***, appelé **fichier MAT**.

La restauration des données depuis un **fichier MAT** dans l'espace de travail se fait à l'aide de la commande **load** : *load myfile.mat*

Pour supprimer toutes les variables de l'espace de travail, on utilise la commande **clear**.

### La liste des fichiers du répertoire courant (Current Folder)

Un clic sur la fenêtre ***Current folder*** (figure.2.) permet d'afficher un gestionnaire de fichiers qui liste les différents fichiers et sous-répertoires présents dans le répertoire courant. Il est aussi possible d'afficher la taille et le type de ces fichiers.

Afin de garantir une meilleure gestion des fichiers, il est fortement conseillé de créer un répertoire autre que celui fournit par Matlab pour chaque projet.

### L'historique des commandes (Command History)

Matlab conserve l'historique des commandes tapées (figure.3.).Il est donc possible à l'aide des flèches du clavier de remonter dans la liste des instructions déjà entrées pour retrouver une instruction particulière pour la réutiliser et éventuellement la modifier avant de l'utiliser à nouveau.

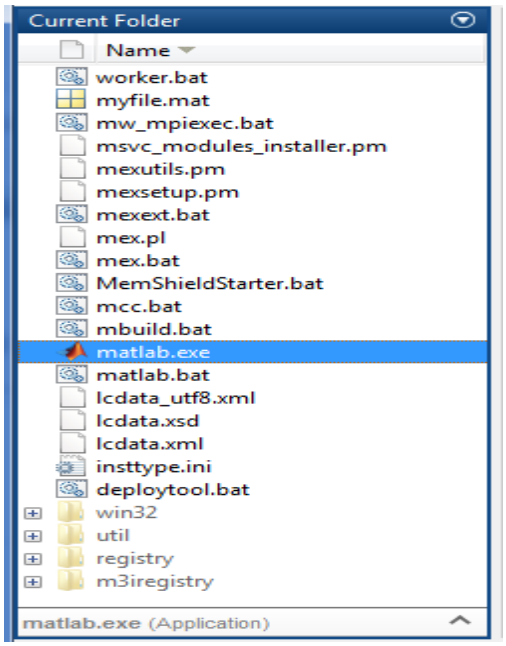



**Figure.2.** Liste des fichiers du répertoire courant

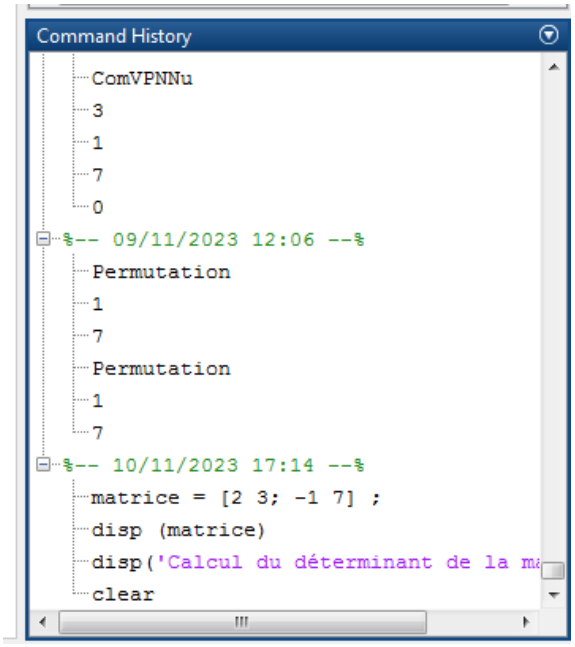



**Figure.3.** Historique des commandes tapées

# ***Programmer sous MATLAB***

Nous avons vu jusqu'à présent comment utiliser Matlab pour effectuer des commandes ou pour évaluer des expressions en les écrivant dans la ligne de commande (Après le prompt >>).

Les commandes utilisées s'écrivent généralement sous forme d'une seule instruction (éventuellement sur une seule ligne). Cependant, il existe des problèmes dont **la description de leurs solutions nécessite plusieurs commandes**, ce qui réclame l'utilisation de plusieurs lignes.

Afin de faciliter la réalisation des programmes Matlab, on enregistre la séquence des commandes dans un fichier appelé un «**M-file** » et de les faire exécuter par Matlab. Un tel fichier doit obligatoirement avoir une extension de la forme **.m** (d'où le nom M-file) pour être considéré par Matlab comme un fichier de commande. On distingue deux types de M-file, les **fichiers de scripts** et **les fichiers de fonctions**.

***Un script*** est un ensemble de commandes Matlab qui joue le rôle d'un programme principal. Si le script est écrit dans le fichier de nom ***nom.m*** on l'exécute dans la fenêtre '**Command Window'** en tapant ***nom***. Même si l'on ne souhaite pas à proprement parler écrire de programme, utiliser un script est très utile. Il est en effet beaucoup plus simple de modifier des commandes dans un fichier à l'aide d'un éditeur de texte que de les retaper dans la fenêtre des commandes.

Les instructions de contrôle sous Matlab sont très proches de celles existant dans d'autres langages de programmation. Le tableau suivant les présente :

| *Instruction* | *Syntaxe* |
|---|---|
| Traitements conditionnelles | **if** *expression logique*<br>séquence de commandes<br>**end**<br>**Ou**<br>**if** *expression logique*<br>séquence de commandes 1<br>**else**<br>séquence de commande 2<br>**end** |
| | **switch** var<br>case cst_1 , séquence de commandes 1<br>case cst_2 , séquence de commandes 2<br>...<br>case cst_N , séquence de commandes N<br>**otherwise** séquence de commandes par défaut<br>**end** |
| Traitements répétitives | **for** indice = borne_inf : borne_sup<br>séquence de commandes<br>**end** |
| | **while***expression logique*<br>séquence de commandes<br>**end** |
| | **repeat**<br>séquence de commandes<br>**until** *expression logique* |

## Opérateurs de comparaison et opérateurs logiques

*Les opérateurs de comparaison et les opérateurs logiques* sont utilisés essentiellement dans les instructions de contrôle. Les opérateurs de comparaison sont :

== : Égal à (x == y)
> : Strictement plus grand que (x >y)
< : Strictement plus petit que (x <y)
> = : Plus grand ou égal à (x ≥ y)
< = : Plus petit ou égal à (x ≤ y)
~= : Différent de (x ~= y)

Pour les opérateurs logiques, on a les symboles **&, |, ~** pour les fonctions logiques *ET, OU et NON*.

*Débogage (Debug)*

En informatique le débogage est une action permettant l'élimination des bogues (les défauts de réalisation ou de conception se manifestant par des anomalies de fonctionnement).

## Les nombres en MATLAB

MATLAB utilise la notation décimale conventionnelle, avec un point décimal facultatif '**.**' et le signe '**+**' ou '**-**' pour les nombres signés. La notation scientifique utilise la lettre '**e**' pour spécifier le facteur d'échelle en puissance de 10.

Les nombres complexes utilisent les caractères 'i' et 'j' (indifféremment) pour designer la partie imaginaire. Ils peuvent être écrits sous forme cartésienne ou polaire. Le tableau suivant présente quelques exemples :

| ***Type*** | ***Exemples*** | | |
|---|---|---|---|
| Entier | 1 | -36 | + 1445 |
| Réel en notation décimale | +0.107 | 3.1419 | -2.1237 |
| Réel en notation scientifique | -1.67e-10 | 1.4457e+13 | +1.5007e-25 |
| Complexe | 1+7i | -1.2 + j*0.789 | 1.25*exp(j*0.246) |

### Les opérations arithmétiques de base

MATLAB fournie une série des opérations arithmétiques de base. Parmi les quelles on cite :

- a + b addition
- a – b soustraction
- a ^ b mettre a à la puissance de b
- mod(a, b) le reste de la division entière de a sur b
- a / b division
- a * b multiplication
- abs(a) la valeur absolue
- sqrt(a) la racine carrée de a

## Affichage graphique

Les possibilités graphiques de Matlab sont innombrables. Par exemple, on peut utiliser la fonction **plot** pour tracer la fonction *sinus* sur un **vecteur** dont les valeurs vont de 0 à 2π :

```
>> x = [0 :2*pi/100 :2*pi], y = sin(x), plot(x,y)   %  2*pi/100 représente le pas
```

On peut superposer un ***quadrillage*** (ou le faire disparaître) par la commande **grid** (**grid off**).

On peut aussi mettre un ***titre*** et des ***étiquettes aux axes*** en utilisant les commandes :

**title**('graphe de sinus'),
**xlabel**('x'),
**ylabel**('sinus(x)').

Il est possible d'afficher **deux** ou **plusieurs graphes** en même temps. Pour faire on introduit une série des coordonnées (x,y) à la fonction plot: **plot(x1, y1, x2, y2, …)**

Afin de différencier les différents graphes on utilise la commande **legend (legend1, legend2, …)**.

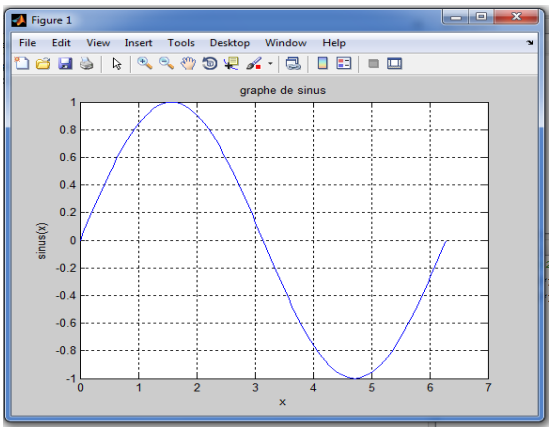



**Figure.3.** Exemple d'affichage graphique