

1.1 Introduction à MATLAB.

1.1.1 Généralités.

MATLAB est un logiciel de calcul matriciel développé il y a plus de vingt ans, qui permet avec les versions actuelles de piloter des instruments de mesure, de faire de l'analyse d'images, du traitement du signal, etc... **MATLAB** est compatible avec les différents systèmes d'exploitation usuels (Windows , Unix, Mac, Linux).

Le cœur du logiciel étant le calcul matriciel, on retrouvera donc toutes les fonctions nécessaires à ces calculs, ce type de représentation (matrice) étant lui aussi à la base de l'automatique, tout naturellement c'est devenu l'un des premiers logiciels de simulation de systèmes linéaires. Afin d'en faciliter l'utilisation de nombreuses fonctions de tracé, de sauvegarde, etc... lui ont été adjointes et il est ainsi devenu un standard de logiciel de calcul scientifique.

Par ailleurs, avec ses fonctions de boucle (for, while, ...), de condition (if, else, ...) et de test (égalité, >, <, ==, ...), il peut être considéré comme un langage de programmation (certaines universités nord-américaines l'utilise comme logiciel d'apprentissage de l'informatique scientifique).

Le cœur du logiciel **MATLAB** comprend essentiellement 5 éléments :

Un environnement et des outils de développement

L'environnement visible est constitué de fenêtres graphiques telles que la Fenêtre de Commandes, l'historique des Commandes, un Editeur de texte et différents navigateurs qui permettent de visualiser l'espace de travail, les fichiers, et les chemins d'accès.

Une bibliothèque de fonctions mathématiques

Cette bibliothèque est constituée d'un ensemble très complet de fonctions mathématiques préprogrammées telles que la somme, le sinus...ou des fonctions plus complexes telles que l'inverse d'une matrice, ses valeurs propres, la transformée de Fourier etc...

Le langage MATLAB

MATLAB comporte un certain nombre d'instructions qui permettent d'utiliser des instructions de contrôle, différentes structures de données, d'effectuer des entrées/sortie. MATLAB permet donc d'écrire des programmes permettant de résoudre des problèmes complexes.

Les graphiques

MATLAB possède une interface et une bibliothèque permettant de produire des graphiques de qualité en deux ou trois dimensions, de créer des animations, etc...Au niveau de l'utilisation au Département GEN, la possibilité de résoudre un problème mathématique et de produire des graphiques dans un unique environnement de travail constitue un des atouts majeurs de MATLAB.

Une interface externe

C'est une bibliothèque qui permet d'écrire des programmes en C ou en Fortran qui interagissent avec MATLAB ou d'inclure dans MATLAB des fonctions écrites en C ou Fortran. Cette fonctionnalité ne sera pas évoquée dans le cadre de ce fascicule.

1.1.2 Spécificités de MATLAB

Par rapport à d'autres langages tels que *Fortran*, *C* ou *C++*, MATLAB possède quelques particularités:

- il n'est pas nécessaire de compiler un programme pour l'exécuter.
- il n'est pas nécessaire de déclarer les variables utilisées dans un programme ; il est possible d'effectuer une déclaration dynamique de variables, c'est-à-dire de créer et supprimer des variables au cours de l'exécution d'un programme.
- il est possible d'exécuter un grand nombre d'opérations sur les matrices sans utiliser de boucles sur les indices des éléments des matrices, ce qui permet d'écrire des programmes "compacts".
- MATLAB possède une interface graphique qui permet de visualiser très rapidement et facilement les résultats dans des graphiques de qualité. Cette interface graphique est l'un des atouts majeurs de MATLAB.
- MATLAB possède une bibliothèque mathématique très riche ; ainsi, la plupart des algorithmes usuels utilisés en analyse numérique (intégration de fonctions, résolution d'équations différentielles,...) existent sous forme de fonctions préprogrammées.
- MATLAB possède un certain nombre de Boîtes à Outils (ou Toolbox) spécialisées qui permettent d'effectuer de l'Acquisition et du Traitement d'Images ou du Signal, d'interfacer un appareil de mesure (oscilloscope, camera,...) ou une carte de numérisation analogique/digital, de résoudre des équations aux dérivées partielles,...
- MATLAB possède une Toolbox particulière, nommée SIMULINK, qui permet de simuler des systèmes; l'utilisation de cette fonctionnalité de MATLAB fait l'objet d'un enseignement spécifique au Département GEN.
- Il est possible de suivre l'exécution d'un programme instruction par instruction, et de le "debugger" grâce à un éditeur spécial.
- Enfin, MATLAB a une diffusion mondiale, tant au plan universitaire qu'industriel.

1.1.3 Fonctions spéciales de MATLAB (Toolbox)

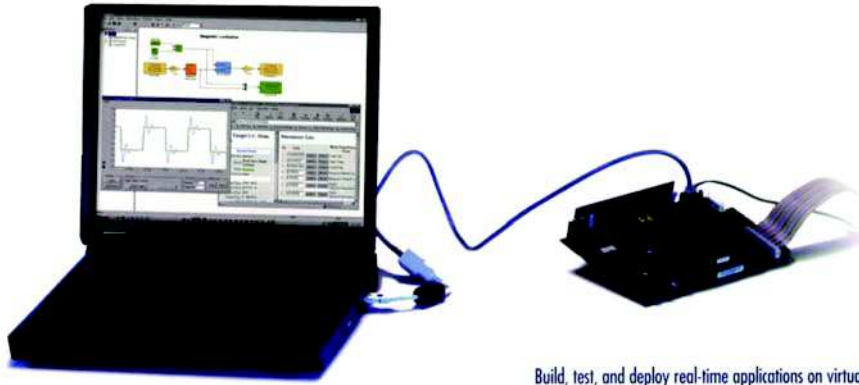
A partir du noyau dur, des boîtes à outils ont été élaborées, ainsi nous pouvons trouver des modules pour le traitement du signal (Signal Processing Toolbox) l'automatique classique (Control system toolbox) ainsi que pour l'Automatique avancée (Robust control, Nonlinear Control Design).

A l'heure actuelle Matlab dispose d'environ 45 boîtes à outils dans des domaines aussi divers que l'Automatique, le Traitement du Signal des Images et des Télécommunications, l'Analyse de données, les Mathématiques appliquées, la Finance, le Temps Réel. Ces boîtes à outils contiennent des instructions spécifiques au domaine de la boîte à outil que vous pourrez appeler dans vos lignes de commande. Au département vous utiliserez surtout les boîtes à outils suivantes : Control toolbox pour l'automatique continue et Signal Processing toolbox pour le traitement du signal.

Il existe aussi des outils complémentaires (14) pour générer des programmes C ou C++, des liens avec Excel, des Bases de Données (Oracle, Microsoft SQL Server, etc..), des instruments de mesures (GPIB, VISA), des serveurs Web et d'autres choses. Des Bibliothèques SIMULINK dans le domaine du Génie Electrique, de la Mécanique et de l'espace sont aussi disponibles.

Depuis cette année il existe aussi une boîte à outil qui permet de faire directement des acquisitions à partir de cartes placées dans le PC ou d'instruments de mesures (Oscilloscope, générateur de fonctions, analyseur de spectre, etc ...) au format GPIB ou VISA.

Il est aussi possible de piloter des systèmes en temps réel grâce à xPC Target ou Dspace. A partir de Simulink et des modèles mis au point en simulation, on génère "du code Temps réel" qui sera téléchargé sur la machine cible, qui pilotera le système. Il est aussi possible de générer du code qui sera embarqué sur un système autonome.



Build, test, and deploy real-time applications on virtually any PC hardware. This example uses a laptop PC as the host computer and a single-board computer as a real-time target.

Matlab et Simulink sont à l'heure actuelle les logiciels les plus utilisés tant dans le monde de la Recherche que dans l'Industrie. Ce sont des outils généraux qui sont capables de résoudre une grande quantité de problèmes, mais on peut toujours trouver des logiciels beaucoup plus performants pour des domaines très spécifiques.

L'intérêt d'outils de simulation avancés ne doit pas faire perdre de vue à l'utilisateur que la représentation interne des nombres est entachée d'erreurs qui peuvent conduire à des résultats erronés. Il faut donc faire très attention dans l'interprétation que vous ferez lors de l'utilisation de ces logiciels de calcul numérique. En effet vous allez travailler avec des données codées (approximées) sur un nombre d'octets plus ou moins important... Vous allez ensuite faire des calculs avec ces approximations et donc les résultats seront entachés d'erreurs plus ou moins importantes... La pertinence des résultats issus de la résolution par une méthode numérique est fortement sujette à la forme, au conditionnement numérique initial, au niveau de précision des variables, au test de convergence, aux méthodes de résolution choisies.... *L'ingénieur et l'étudiant doivent à tout moment rester critique et avoir conscience du résultat qu'il doit obtenir à priori.*

Matlab est un langage interprété (comme le BASIC), c'est à dire que les lignes sont interprétées (transformées en langage machine) et exécutées ligne à ligne au fur et à mesure de leur écriture. Il est donc nécessaire d'avoir dans sa machine le logiciel pour exécuter un programme fait par quelqu'un d'autre ou par vous-même.

Il existe trois possibilités pour exécuter ces programmes :

Mode interactif. Ce qui est décrit précédemment, lignes exécutées au fur et à mesure de leur écriture.

Mode exécutif. Matlab exécute un fichier spécifique (M-file, extension .m) qui sera exécuté ligne à ligne.

Création d'un exécutable. Il est nécessaire d'avoir à sa disposition MATLAB Compiler qui permet de transformer son fichier.m en C ou C++ qui sera compilé et pourra être exécuté sans disposer du logiciel MATLAB.

2 L'ENVIRONNEMENT MATLAB - GENERALITES

2.1 Présentation de l'interface

Pour lancer MATLAB, il faut double-cliquer sur l'icône de la *Figure 1* présent sur le bureau du PC. La version disponible au Département GEN est une version 7.0 destinée à l'enseignement. Une fenêtre telle que celle de la *Figure 2* va s'ouvrir.



Figure 1 Icône du logiciel Matlab

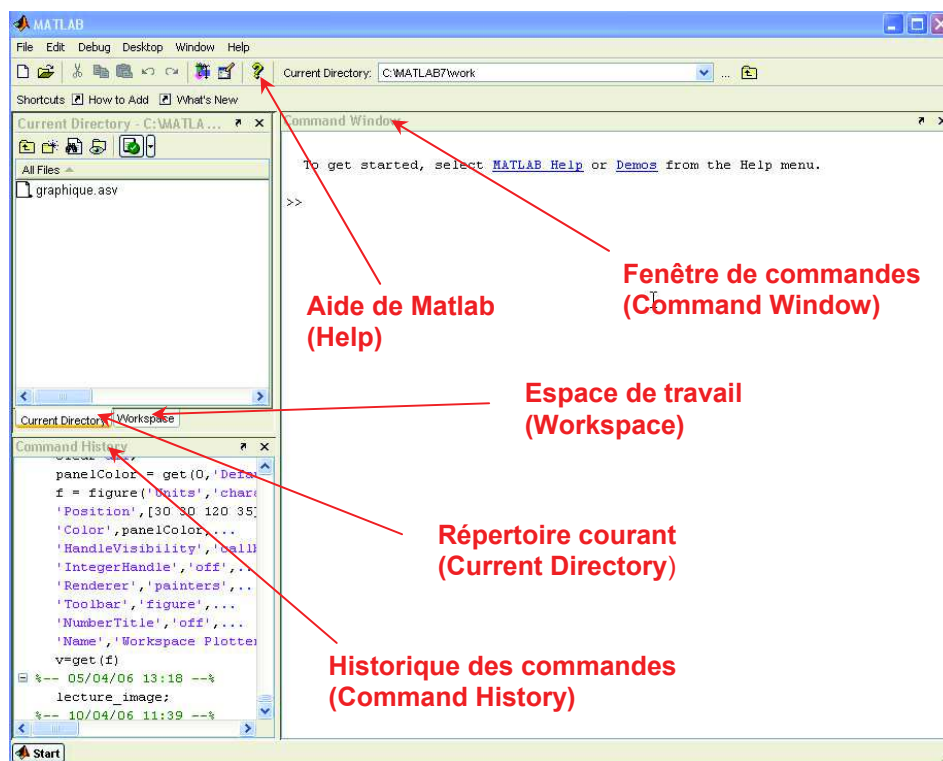


Figure 2 Interface de Matlab

La figure 2 montre que, par défaut, l'interface de MATLAB comporte plusieurs éléments et fenêtres :

- une fenêtre principale, intitulée **Command Window** (ou **Fenêtre de Commandes**) située dans la partie droite de la fenêtre principale ; c'est dans cette fenêtre que sont tapées les instructions dont l'exécution sera obtenue en tapant sur la touche *Entrée* (ou *Return*) du clavier; c'est aussi dans cette fenêtre que l'on tape le nom d'un programme (ou script) dont on souhaite lancer l'exécution ;
- une fenêtre située en haut à gauche comportant deux onglets, intitulés **Workspace** (**Espace de Travail**) et **Current Directory** (**Répertoire courant**). Un clic sur l'onglet **Workspace** permet de visualiser les variables qui sont présentes en mémoire de l'ordinateur, leur type, leur taille,...C'est un outil très utile pour identifier des sources

- d'erreurs dans un programme. Un clic sur l'onglet **Current Directory** permet d'afficher un gestionnaire de fichiers qui liste les différents sous-répertoires et fichiers présents dans le répertoire courant, ainsi que la taille et le type de ces fichiers ;
- différents onglets intitulés **File**, **Edit**, **Debug**, **Desktop**, **Window**, **Help**. L'onglet **Help**, également accessible par l'icône comportant un point d'interrogation jaune, permet d'accéder à l'aide de Matlab.

Remarque

Lorsque l'on ouvre le logiciel MATLAB le répertoire de travail par défaut est généralement C:\Matlab7\work. Toute opération de stockage de fichiers sur le disque dur sera donc effectuée dans ce répertoire, ce qui est fortement déconseillé. Il est donc nécessaire de spécifier le répertoire de travail grâce à une instruction décrite plus loin, ou en utilisant l'onglet de navigation situé en haut de la fenêtre principale.

2.2 Travail en mode interactif ou exécutif

Il existe principalement deux façons de travailler avec MATLAB :

- un **mode interactif** (fonctionnement par ligne de commande), pour lequel on tape dans la fenêtre de commande les instructions dont l'exécution s'effectue à l'aide de la touche *Entrée* (ou *Return*) du clavier. Ce mode de fonctionnement est très utile si on veut exécuter quelques tâches ou calculs très simples en temps réel; il devient vite inefficace lorsqu'il s'agit d'enchaîner de nombreuses tâches (résolution d'une équation différentielle, tracé des résultats, stockage des résultats sur le disque dur,...) ;
- un **mode exécutif** qui consiste à lancer l'exécution d'une suite d'instructions qui auront été préalablement tapées dans une fenêtre d'édition. Le contenu de cette fenêtre est stocké sur le disque dur dans un fichier dont l'extension est **.m** (par exemple, **monprogramme.m**), et son exécution sera obtenue en tapant **monprogramme** dans la Fenêtre de Commande, suivi de la touche *Entrée* (Matlab comprend alors qu'il doit exécuter le contenu des lignes du fichier intitulé **monprogramme.m**). En fait, il existe deux types de fichiers .m décrits plus loin.

Ces deux modes de fonctionnement, qui sont utilisés dans le cadre de cet enseignement, nécessitent que le logiciel MATLAB soit installé sur la machine. Il existe néanmoins une troisième possibilité qui consiste à compiler un fichier **.m** afin de créer un exécutable qui pourra être lancé sur n'importe quelle machine non pourvue de Matlab.

Les deux modes de fonctionnement interactif et exécutif sont illustrés dans l'exemple suivant qui trace la fonction $x = \sin(t)$ pour t compris entre 0 et 2π .

Exemple de fonctionnement en mode interactif

Pour fonctionner en mode interactif (par ligne de commandes), il faut taper chaque instruction à exécuter dans la *Fenêtre de Commandes* à côté du symbole **>>**, suivi de la touche *Entrée*.

Pour éviter de stocker des fichiers n'importe où sur le disque dur, à l'aide de la barre de navigation, il faut d'abord se placer dans un répertoire de travail, par exemple le répertoire *Documents Partagés* du PC. Le résultat de l'exécution de chacune des 6 lignes de commande ci-dessous :

```
>> t = [0 :2*pi/100 :2*pi] ;  
>> x = sin(t) ;  
>> plot(t,x) ;  
>> save monfichier t x ;  
>> clear all ;  
>> load monfichier ;
```

L'affichage dans les fenêtres **Workspace** et **Current Directory** permet de surveiller le résultat de l'exécution de chacune des lignes.

La première ligne de commande a pour effet de créer une variable nommée t , constituée de réels double précision, t variant de 0 à 2π , avec un pas égal à $2\pi/100$. Cette variable t est effectivement présente en mémoire de l'ordinateur (fenêtre **Workspace**), sous la forme d'un tableau à 1 ligne et 101 colonnes.

La deuxième ligne de commande a pour objet de créer une variable x , de la même dimension que t , et dont chaque terme $x(i)$ est égal au sinus de $t(i)$, i variant de 1 à 101. La création effective de cette variable x est visualisée dans la fenêtre **Workspace**.

La troisième ligne de commande a pour objet de tracer dans une fenêtre graphique la courbe $x(t)$. La quatrième ligne de commande a pour effet de créer sur le disque dur dans le répertoire courant un fichier intitulé **monfichier.mat** (l'extension d'un fichier de données Matlab est par défaut **.mat**) et contenant les deux variables t et x .

La cinquième ligne de commande a pour effet d'effacer de la mémoire de l'ordinateur toutes les variables.

Enfin, la sixième ligne de commande a pour effet de charger en mémoire le contenu du fichier **monfichier.mat**.

Pour sauvegarder sur le disque dur le graphique, sélectionner dans l'onglet *File* situé en haut à gauche de cette fenêtre l'option *Save As...* et choisir par exemple **mafigure** pour le nom du fichier. L'examen de la Fenêtre **Current Directory** montre qu'il existe maintenant deux fichiers sur le disque dur :

- un fichier de données intitulé *monfichier.mat* ;
- un fichier graphique intitulé *mafigure.fig*.

Remarques

- i) Ces deux fichiers possèdent des extensions propres à Matlab (.mat et .fig); leur contenu ne peut donc être lu qu'avec Matlab. Vous aurez néanmoins l'occasion de voir qu'il existe d'autres formats de stockage qui permettent à ces fichiers et figures d'être utilisés par d'autres logiciels d'utilisation courante (Word, Excel, Powerpoint,...).*
- ii) Toutes les lignes de commande sont terminées par un point-virgule ; en l'absence du point virgule, MATLAB va non seulement exécuter l'instruction correspondant à cette ligne, mais également afficher à l'écran le résultat de cette exécution (par exemple, pour la première ligne, affichage à l'écran de toutes les valeurs de la variable t).*
- iii) Comme le montre cet exemple, MATLAB ne nécessite pas de déclaration de type ou de taille de variable. Lorsque MATLAB rencontre un nouveau nom de variable, il crée automatiquement cette variable, et lui alloue la quantité de mémoire nécessaire.*
- iv) La première ligne de commande pour la création de la variable t illustre la façon dont on peut travailler avec une seule instruction sur tous les éléments d'un tableau sans utiliser de boucles. Le contenu de la variable t est identique à celui qu'on obtiendrait en remplaçant la première ligne par les trois lignes suivantes :*

```
for k=1 :101
    t(k)=(k-1)*2*pi/100 ;
end
```

Exemple de fonctionnement en mode exécutif

On souhaite maintenant réaliser les mêmes instructions, mais en mode exécutif, c'est-à-dire en lançant l'exécution d'un programme stocké sur le disque dur dans un fichier dont l'extension MATLAB est **.m**. Un clic sur l'icône situé en haut à gauche de la barre de commande (qui représente une feuille de papier) a pour effet ouvrir une fenêtre d'édition, dans laquelle il faut taper les 6 lignes de commande précédentes. Pour sauvegarder dans un fichier du disque dur le contenu de ces instructions, il faut sélectionner dans l'onglet **File** situé en haut à gauche de cette fenêtre l'option *Save As...* et donner au fichier un nom, par exemple *monprogramme*. L'examen du répertoire courant permet de vérifier la création sur le disque dur du fichier intitulé **monprogramme.m**. Pour lancer l'exécution du programme, il faut alors taper **monprogramme** dans la fenêtre de commande à côté du symbole **>>**. De même que pour le mode interactif, l'exécution des instructions conduit à la création des variables **t** et **x**, au tracé du graphique **x(t)** et à la création sur le disque dur d'un fichier **monfichier.mat** contenant les variables **t** et **x**.

Remarque

A travers la présentation des deux modes de fonctionnement interactif et exécutif, les trois types de fichiers propres à MATLAB qui seront utilisés dans le cadre de cet enseignement ont été présentés:

- *programme : fichier avec extension .m*
- *données : fichier avec extension .mat*
- *figure : fichier avec extension .fig*

2.3 Fonction "help"

Pour obtenir de l'aide sur un sujet, une instruction ou une fonction, on tape **help** suivi par le sujet, le nom de l'instruction ou de la fonction désirée.

Exemple :

» help atan

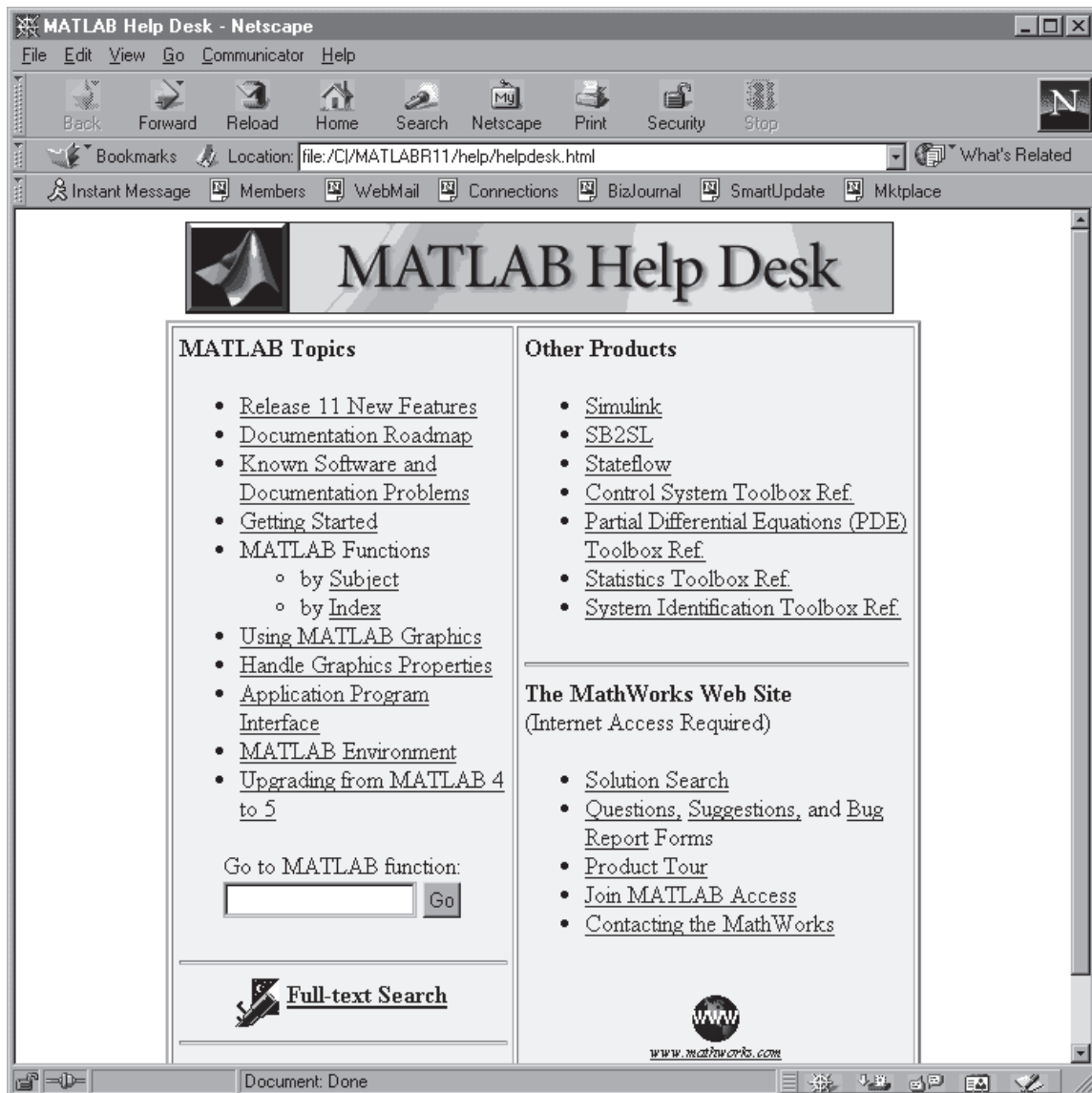
atan Inverse tangent.

atan(X) is the arctangent of the elements of X.

See also atan2.

La réponse proposée suggère souvent une aide sur une instruction similaire ('See also ...')

Une autre méthode pour obtenir de l'aide consiste à taper **helpdesk** dans une console et on obtient alors la fenêtre suivante qui fait appel à l'environnement Netscape :

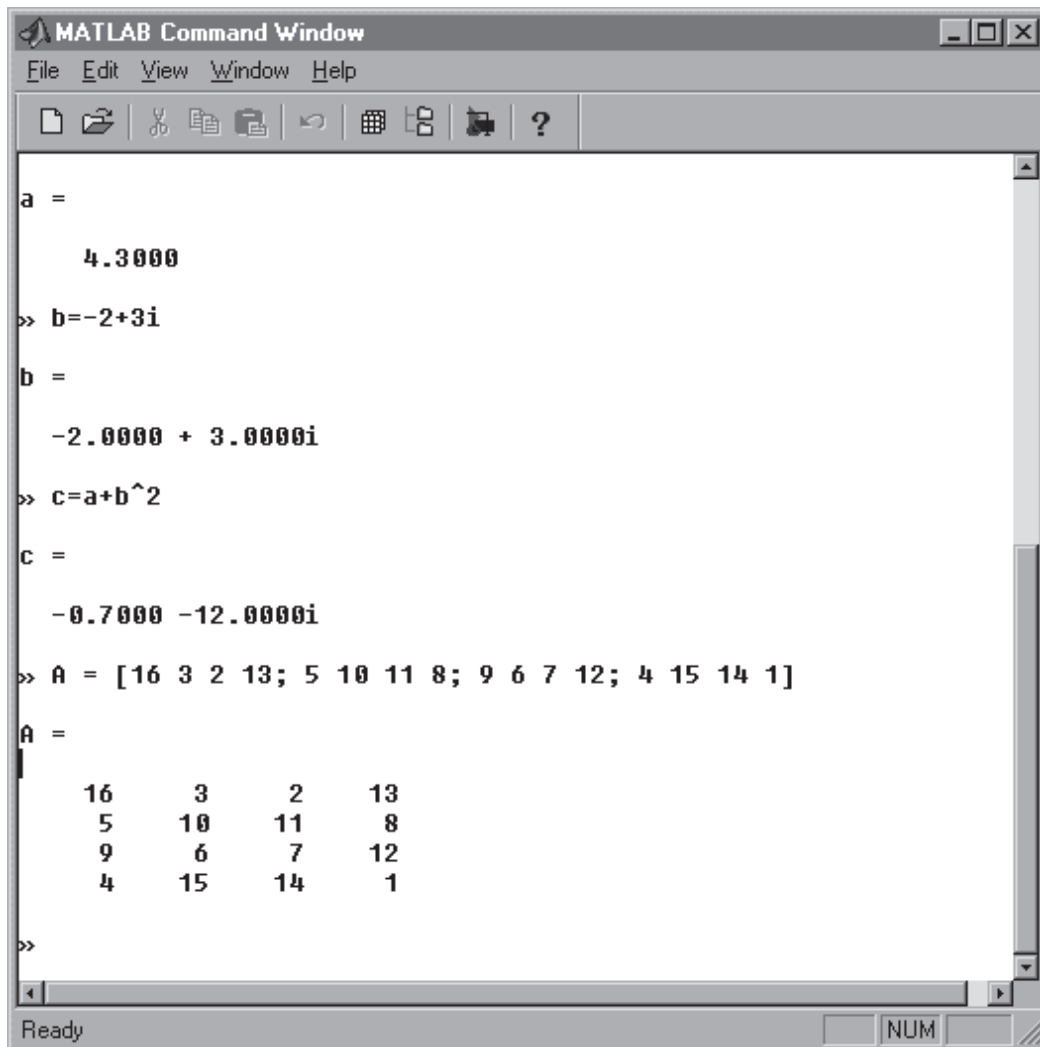


Il est possible d'accéder à une aide en ligne très complète comprenant notamment toute la documentation en cliquant sur le « ? » dans la barre de menu de la console Matlab.

2.4 Espace de travail (Workspace).

Les variables sont définies au fur et à mesure que l'on donne leurs noms et leurs valeurs numériques ou leurs expressions mathématiques.

Les variables ainsi définies sont stockées dans l'espace de travail et peuvent être utilisées dans les calculs ultérieurs. Attention, le langage Matlab fait la différence entre les majuscules et les minuscules.



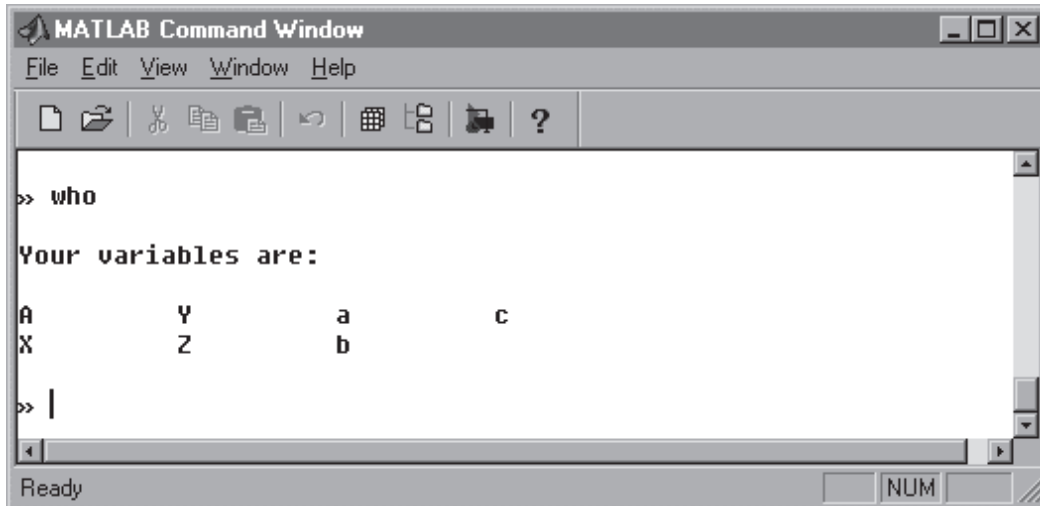
```
MATLAB Command Window
File Edit View Window Help
[Icons]
a =
    4.3000
>> b=-2+3i
b =
   -2.0000 + 3.0000i
>> c=a+b^2
c =
   -0.7000 -12.0000i
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
>>
Ready NUM
```

2.4.1 Information sur l'espace de travail.

Pour obtenir une liste des variables dans l'espace de travail, on utilise les instructions suivantes :

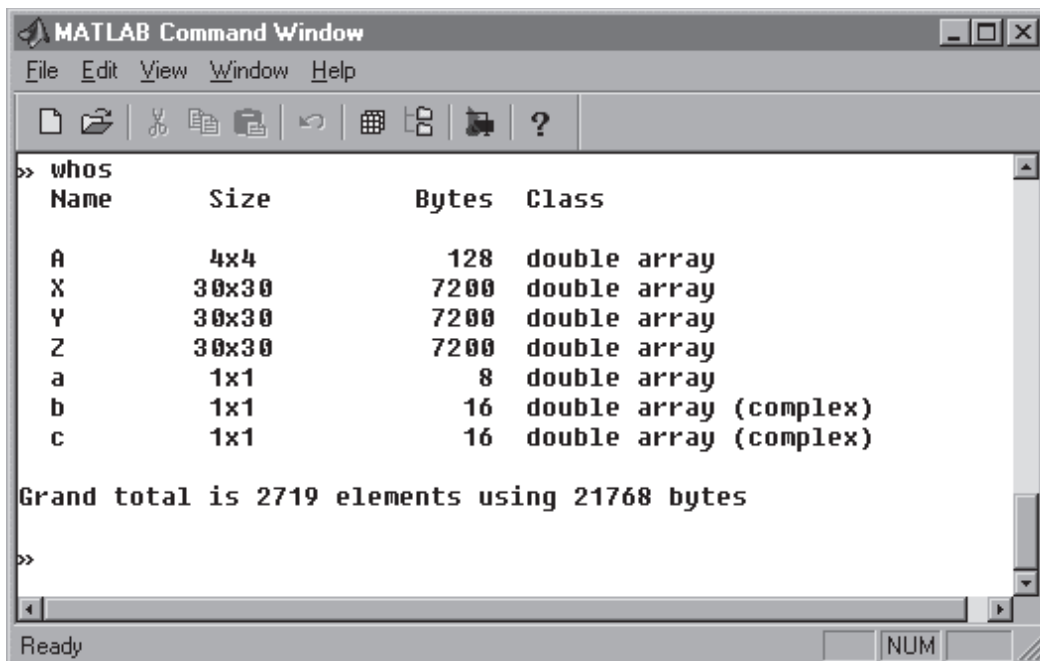
- who** Affichage des variables dans l'espace de travail.
whos Affichage détaillé des variables dans l'espace de travail.

La deuxième instruction (**whos**) donne beaucoup plus de détails.



```
MATLAB Command Window
File Edit View Window Help
[Icons]
>> who
Your variables are:
A          Y          a          c
X          Z          b
>> |
Ready NUM
```

Exemple d'utilisation de l'instruction « **who** »



```
MATLAB Command Window
File Edit View Window Help
[Icons]
>> whos
Name      Size      Bytes  Class
A         4x4       128    double array
X        30x30     7200   double array
Y        30x30     7200   double array
Z        30x30     7200   double array
a         1x1        8      double array
b         1x1       16     double array (complex)
c         1x1       16     double array (complex)

Grand total is 2719 elements using 21768 bytes
>>
Ready NUM
```

Exemple d'utilisation de l'instruction « **whos** »

2.4.2 *Enregistrement des variables de l'espace de travail dans un fichier.*

Pour enregistrer les variables de l'espace de travail dans un fichier, on utilise les instructions suivantes :

save Enregistrer toutes les variables dans un fichier matlab.mat. Dans une session ultérieure, taper **load** pour ramener l'espace de travail enregistrée.

save fichier1.mat x y z A X Enregistrer les variables x, y, z, A, X dans le fichier fichier1.mat. Dans une session ultérieure, taper **load fichier1** pour recharger le nom et le contenu des variables x, y, z, A, X dans l'espace de travail.

2.4.3 *Commandes DOS/UNIX.*

Il est possible d'utiliser certaines commandes des systèmes d'exploitation DOS/UNIX.

pwd permet de connaître le répertoire courant de travail
cd .. permet de remonter d'un niveau dans la hiérarchie des répertoires
cd *directory_name* permet d'accéder au répertoire '*directory_name*'
dir ou **ls** permettent de connaître le contenu du répertoire courant
mkdir *directory_name* permet de créer un répertoire *directory_name*
delete('file_name') supprime un fichier *file_name*

3 OPERATIONS MATHÉMATIQUES.

3.1 scalaire, vecteur, matrice.

L'élément de base de MATLAB est la matrice. Un scalaire est ainsi une matrice de dimension 1×1 , un vecteur colonne de dimension n est une matrice $n \times 1$, un vecteur ligne de dimension n , une matrice $1 \times n$.

Contrairement aux langages de programmation usuels, il n'est pas obligatoire de déclarer les variables avant de les utiliser et, de ce fait, il faut prendre toutes les précautions dans la manipulation de ces objets. Les variables sont affectés directement à l'aide du signe égal (=).

3.1.1 *Scalaire.*

Les scalaires se déclarent directement :

```
>> x = 0.12;
>> a = 1.7e-4;
```

Les nombres complexes peuvent être écrits sous forme cartésienne ou polaire :

Forme cartésienne : $0.5 + i*2.7$ $-1.2 + j*0.789$ $2.5 + 9.7i$
 Forme polaire : $1.25*\exp(j*0.246)$

Pour choisir le format d'affichage pour les nombres, on utilise l'instruction format :

format short	0.1234
format long	0.12345678901234
format short e	1.2341E+002
format long e	0.123456789012345E+002
format hex	0123456789ABCDEF

Nota : i) le format hexadécimal est utilisé pour les calculs en base 16.

ii) ne pas confondre format d'affichage et précision machine

3.1.2 *Vecteurs.*

Il existe différentes façons de définir un vecteur :

- soit en donnant la liste de ses éléments entre crochet « [] » :

```
>> V_ligne = [0 1 2]
V_ligne =
0     1     2
```

Pour les vecteurs colonne, on sépare les éléments par des points-virgules :

```
>> V_colonne = [0;1;2]
V_colonne =
0
1
2
```

- soit en utilisant l'opérateur d'incrémentation (:). Ainsi, pour créer un vecteur ligne constitué des valeurs de 0 à 1 par incrément de 0,2 :

```
>> V = [0:0.2:1]
V =
Columns 1 through 6
0     0.2000     0.4000     0.6000     0.8000     1.0000
```

Par défaut, l'incrément est de 1:

```
>> V = [0:5]
V =
0     1     2     3     4     5
```

- soit en utilisant une fonction qui génère un vecteur. Exemple :

```
>> x=linspace(1,10,6) % variation linéaire : 6 valeurs sont réparties linéairement
                        dans l'intervalle donné allant de 1 à 10.
x =
1.0000     2.8000     4.6000     6.4000     8.2000    10.0000
ou :
>> y=logspace(1,3,7) % variation logarithmique – 7 valeurs sont réparties
                      linéairement sur une échelle logarithmique dans l'intervalle
                      donné
y =
1.0e+003 *
0.0100     0.0215     0.0464     0.1000     0.2154     0.4642
1.0000
```

Remarque :

Lorsqu'on ajoute un «;» à la fin d'une instruction, elle est exécutée mais le résultat n'est pas affiché :

```
>> a=[1 2 3 4 5];
>> b=-2.5;
>> c=b*a;
>>
```

Lorsqu'il n'y a pas de «;» à la fin d'une instruction, elle est exécutée et le résultat est affiché :

```
>> a=[1 2 3 4 5]
a =
1     2     3     4     5
>> b=-2.5
b =
-2.5000
>> c=b*a
c =
-2.5000    -5.0000    -7.5000   -10.0000   -12.5000
>>
```

On se servira de cette remarque pour afficher les résultats de calcul à l'écran.

3.1.3 Matrices.

Les matrices peuvent être construites directement :

```
>> M = [1 2 3; 4 5 6 ; 7 8 9]
M =
     1     2     3
     4     5     6
     7     8     9
```

On peut avoir accès aux éléments de la matrice par :

```
>> m21 = M(2,1)    % 2e ligne, 1ère colonne
m21 =
     4
```

Il est aussi possible de stocker dans un vecteur une ou plusieurs lignes (ou colonnes). Ainsi, si l'on veut stocker la deuxième colonne de la matrice M :

```
>> V = M(:,2)      % ici, (:) signifie toutes les lignes
V =
     2
     5
     8
```

De la même manière, si l'on veut stocker les lignes 2 et 3 :

```
>> M2=M(2:3,:)     % (2:3) signifie ligne 2 à 3 et (:) signifie toutes les colonnes
M2 =
     4     5     6
     7     8     9
```

Il existe aussi des matrices prédéfinies dans Matlab, notamment :

Zeros	Matrice de 0
Ones	Matrice de 1
Eye	Matrice identité
Linspace	Vecteur linéairement espacé
Logspace	Vecteur logarithmiquement espacé
rand	Nombre aléatoire à répartition uniforme
randn	Nombre aléatoire à répartition normale

`ones(N,N)` définit une matrice carré contenant des 1 et de taille $N \times N$.

`zeros(N,N)` définit une matrice carré contenant des 0 et de taille $N \times N$.

Valeurs propres d'une matrice :

```
[v,d] = eig(X)
```

Produit une matrice diagonale d des valeurs propres de X et une matrice pleine v dont les colonnes correspondent aux vecteurs propres tel que $X \cdot \underline{d} = \underline{v} \cdot \underline{d}$.

```
>> [v,d]=eig(A)
```

```
v =
-0.4152  0.3001  -0.5584i  0.3001  +0.5584i
 0.8079  0.0363  -0.4525i  0.0363  +0.4525i
-0.4182  0.5330  -0.3285i  0.5330  +0.3285i
d =
-0.8260     0             0
```

```

0      5.6030 + 1.1550i    0
0      0      5.6030    - 1.1550i

```

Inversion d'une matrice.

```
inv(X)      inversion de la matrice X
```

```
» inv(A)
```

```
ans =
```

```

-0.3963  0.5456  0.2455
 1.2702 -0.9057 -0.6720
-0.5722  0.5571  0.4337

```

Matrice exponentielle.

```
expm(X)      Matrice exponentielle
```

Transposée d'une matrice.

```
B = A' La matrice B est égale à la matrice A transposée
```

Somme de deux matrices.

```
C = A + B
```

produit de deux matrices.

```
D=A*B
```

EMPLOI DES INDICES

Les éléments d'un vecteur ou d'une matrice peuvent être adressés en utilisant les indices sous la forme suivante :

```

t(10)      élément no. 10 du vecteur t
A(2,9)     élément se trouvant à ligne 2, colonne 9 de la matrice A
B(:,7)     la colonne 7 de la matrice B
B(3,:)     la ligne 3 de la matrice B

```

Addition

```

D = A - B   Soustraction
Z = X*Y     Multiplication
X = B/A     Équivalent à B*inv(A)

```

OPÉRATION «ÉLÉMENT PAR ÉLÉMENT»

Les opérations «élément par élément» des vecteurs et des matrices sont effectuées en ajoutant un point (.) avant les opérations * / \ ^ '.

Exemple 2 :

```

>> A=[1 2 3 4 5];
>> B=[6 7 8 9 10];
>> C=A.*B
C =
 6    14    24    36    50
>> D=A./B
D =
 0.1667    0.2857    0.3750    0.4444    0.5000

```

3.2 Opérations arithmétiques

3.2.1 Opérations arithmétiques usuelles.

Les opérations usuelles d'addition, de soustraction et de multiplication par scalaire sur les vecteurs et les matrices sont définies dans MATLAB. Les expressions mathématiques sont écrites de la façon habituelle :

$$z = 5 * \exp(-0.4 * x) * \sin(7.5 * y);$$

+	Addition
-	Soustraction
*	Multiplication
^	Elévation à la puissance
'	Transposée de la matrice
/	Division à droite
\	Division à gauche

En revanche, dans le cas de la multiplication (ou de la division) de vecteurs (ou de matrices), il faut faire attention à leurs dimensions. En effet :

```
>> V1 = [1 2];
>> V2 = [3 4];
>> V = V1 * V2    % multiplication de vecteurs
??? Error using mtimes
Inner matrix dimensions must agree.
```

MATLAB indique une erreur lorsque les dimensions ne concordent pas (lisez attentivement les messages d'erreur, ils sont parfois utiles pour corriger vos programmes). L'erreur provient ici du fait qu'on ne peut multiplier deux vecteurs lignes (relisez vos cours de calcul matriciel...).

Il est possible par contre de multiplier (ou de diviser) élément par élément. Il suffit d'ajouter un point (.) avant les opérations *, /, \, ^ :

```
>> V = V1 .* V2    % multiplication de vecteurs élément par élément
V =
3     8
```

3.2.2 Calcul matriciel.

Bien évidemment, Matlab recèle de nombreux moyens propres au calcul matriciel comme l'inversion de matrice (**inv**), l'extraction de la diagonale de la matrice (**diag**), la détermination des valeurs propres d'une matrice (**eig**), de son déterminant (**det**)... Vous retrouverez une partie de ces commandes à la fin de ce fascicule dans la rubrique résumant les commandes Matlab.

3.2.3 Analyse des données par colonnes.

De nombreuses commandes Matlab permettent de réaliser des opérations colonne par colonne (ou ligne par ligne), notamment :

Max	Valeur maximum
Min	Valeur minimum
Mean	Valeur moyenne
Sum	Somme des éléments
prod	Produits des éléments
Std	Ecart type
sort	Tri par ordre croissant

3.2.4 Fonctions mathématiques élémentaires.

Abs	Valeur absolue
Sqrt	Racine carrée
Real	Partie réelle
imag	Partie imaginaire
Round	Produits des éléments
Sign	Signe
exp	Exponentielle
Log / log10	Log base e/ log base 10

3.2.5 Fonctions trigonométriques.

Sin, asin, sinh, asinh	Sinus, Arcsin trigo et hyperbolique
Cos, acos, cosh, acosh	
tan, atan, tanh, atanh	

3.3 Variables et fonctions

VARIABLES

Exemples de variables en utilisant soit une affectation (signe =) soit une expression mathématique :

```
a = 1.25;  
x = 0:0.5:10;  
y = a*x;  
z = y.^2;
```

EXPRESSIONS MATHÉMATIQUES

On écrit les expressions mathématiques de la façon habituelle :

```
z = 5*exp(-0.4*x).*sin(7.5*y);
```

FONCTIONS MATHÉMATIQUES

Les fonctions mathématiques de base sont données dans le tableau suivant :

abs Valeur absolue Module (nb complexe)	angle Argument (nb complexe)	sqrt Racine carrée	real Partie réelle	imag Partie imaginaire
conj Conjuguée (nb complexe)	round arrondir	fix Arrondir (vers zéro)	floor Arrondir (vers $-\infty$)	ceil Arrondir (vers ∞)
sign signe	rem reste	exp exponentielle	log Logarithme base e	log10 Logarithme base 10

Les fonctions trigonométriques sont données dans le tableau suivant :

sin	cos	tan	asin	acos	atan
sinh	cosh	tanh	asinh	acosh	atanh

Exemple 3 :

```
>> x=-2+5i
x =
-2.0000 + 5.0000i
>> a=real(x)
a =
-2
>> b=imag(x)
b =
5
>> X=abs(x)
X =
5.3852
>> alfa=angle(x)
alfa =
1.9513
```

Exemple 4 :

```
>> w=50;
>> t=0.5e-3;
>> y=25*exp(-4*t)*cos(w*t)
y =
24.9423
```

CRÉATION DE FONCTIONS

L'utilisateur peut créer des fonctions particulières pour ses applications. Voir «Programmation avec MATLAB».

4 GRAPHIQUES.

Pour ouvrir une fenêtre dans laquelle seront tracés des graphiques, il faut utiliser l'instruction **figure**. S'il n'y a besoin que d'une seule fenêtre graphique, cette instruction est optionnelle.

4.1 Graphiques 2D.

4.1.1 Tracée de courbes

On utilise l'instruction **plot** pour tracer un graphique 2D :

plot(x,y)	Trace le vecteur y en fonction du vecteur x.
plot(t,x,t,y,t,z)	Trace x(t), y(t) et z(t) sur le même graphique.
plot(t,z,'r--')	Trace z(t) en trait mixte rouge.
stem(k,z)	Trace z(k.T) en échantillons (pour les fonctions discrètes).

La chaîne de caractères correspond à la définition du symbole et de la couleur utilisés selon le code suivant :

y	jaune	.	point	-	trait plein
m	magenta	o	cercle	:	trait pointillé
c	cyan	x	croix	-.	plein pointillé
r	rouge	+	plus	--	trait mixte
g	vert	*	*		
b	bleu	s	carré		
w	blanc	d	losange		
k	noir	v	triangle		
		^	triangle		
		<	triangle		
		>	triangle		
		p	étoile		
		h	étoile		

Nota : rien ne vaut mieux qu'un essai !

4.1.2 Format de graphique

On peut choisir le format du graphique :

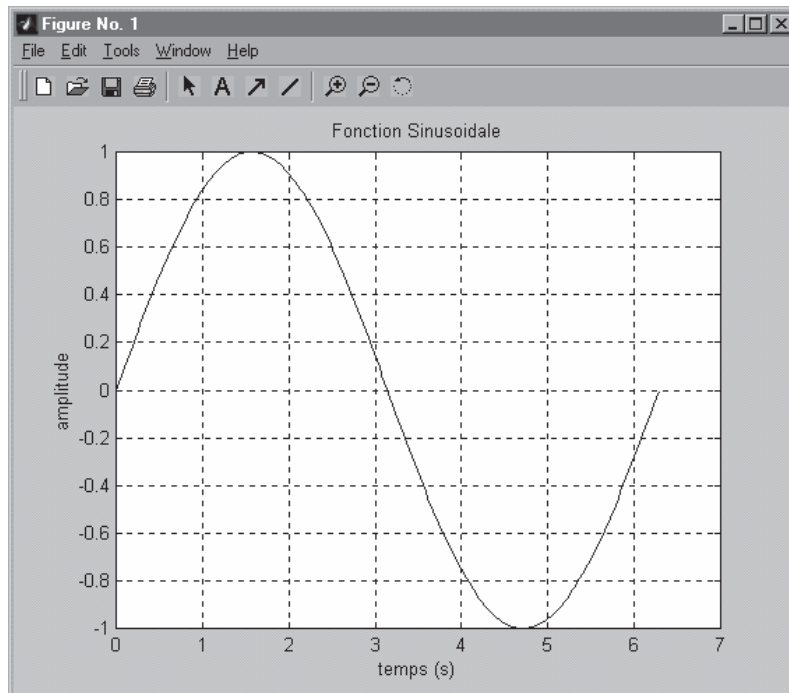
plot(x,y)	Trace y(x) avec des échelles linéaires
loglog(x,y)	Trace y(x) avec des échelles logarithmiques
semilogx(f,A)	Trace A(f) avec une échelle log(f)
semilogy(w,B)	Trace B(w) avec une échelle log(B)
polar(theta,r)	Trace r(theta) en coordonnées polaires
bar(x,y)	Trace y(x) sous forme des barres
grid	Quadrille le graphe de la fenêtre active

Exemple 5 :

```

» t = 0:pi/100:2*pi; % La variable  $\pi$  est prédéfinie et vaut  $\pi = 3.14159265358979$ 
y = sin(t);
plot(t,y)
» grid
» title('Fonction Sinusoidale')
» xlabel('temps (s)'), ylabel('amplitude')
»

```

**4.1.3 Graphique multiple**

On peut tracer plusieurs graphiques dans la même fenêtre en utilisant l'instruction **subplot** pour diviser la fenêtre en plusieurs parties.

Subplot(m,n,p) divise la fenêtre graphique en $m \times n$ sous graphique, m et n étant le nombre de lignes et de colonnes respectivement, la variable p désigne le numéro du graphe.

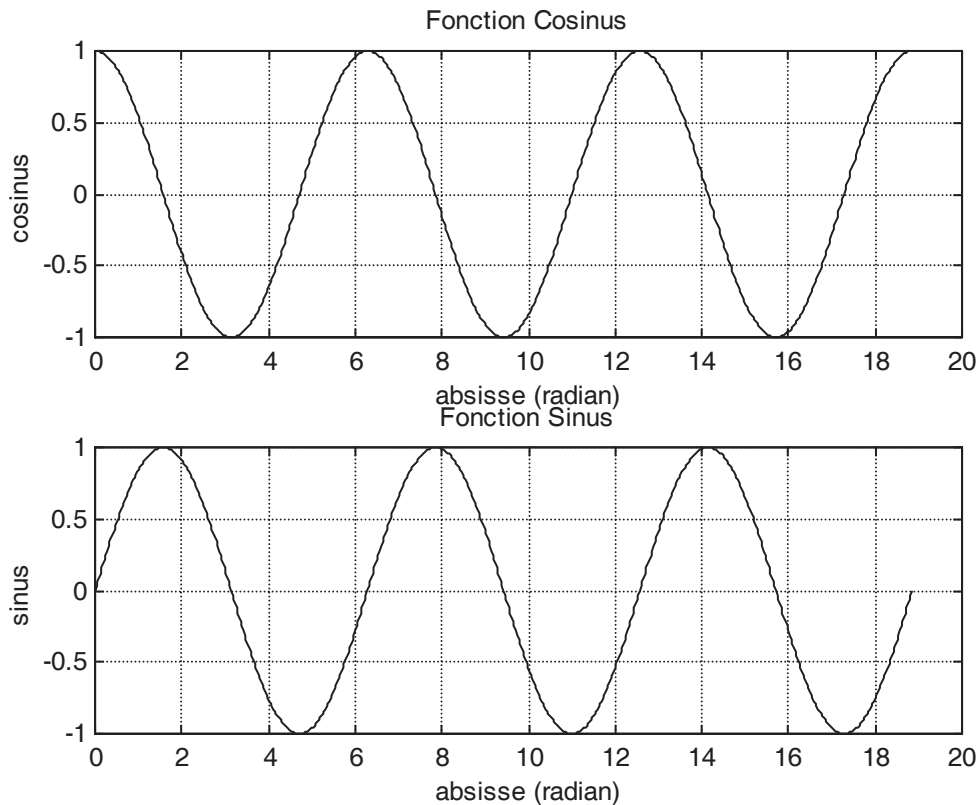
Exemple 6 :

```

>> t = 0:pi/10:6*pi;
>> y1=cos(t);
» y2=sin(t);
» subplot(2,1,1), plot(t,y1,'r')
» grid
» title('Fonction Cosinus')
» xlabel('abscisse (radian)')
» ylabel('cosinus')

» subplot(2,1,2),plot(t,y2,'b')
» grid
» title('Fonction Sinus')
» xlabel('abscisse (radian)')
» ylabel('sinus')

```



4.1.4 Ajout du texte au graphique

title ('Titre du graphique')	Ajoute un titre au graphique
xlabel ('Temps')	Étiquette de l'axe x
ylabel ('Tension')	Étiquette de l'axe y
gtext ('Valeur absolue')	Ajoute du texte au graphique à l'endroit pointé par la souris
legend ('legende 1','legende 2')	Insertion d'une légende

4.1.5 Manipulation de graphiques

axis ([-1 5 -10 10])	Choix des échelles x = (-1,5) et y = (-10,10)
hold	Garde le graphique actif à l'écran (et permet de tracer plusieurs courbes sur le même graphique)

Remarque : toutes les modifications du graphique (échelles, axes, titre, symboles, couleurs,...) peuvent être soit définies par des lignes de programmation (voir les instructions précédentes), soit modifiées à l'aide de la souris. Le premier mode de fonctionnement est intéressant s'il faut tracer automatiquement un grand nombre de figures.

4.1.6 Impression et enregistrement de graphiques

Impression d'un graphique

A partir du menu Fichier, choisir Imprimer. Il faut impérativement modifier le format défini par défaut de la taille de la feuille (Format lettre US en format A4).

Stockage d'un graphique sur le disque dur

Le contenu d'une fenêtre graphique peut-être stocké au format **.fig** qui est un format interne à MATLAB ; c'est le seul format de stockage qui permet de retoucher le graphique ultérieurement. Il est possible de copier une figure MATLAB dans le presse-papier pour l'insérer dans un autre logiciel (Word, Powerpoint,...). Le graphique peut également être stocké sous forme d'image aux différents formats usuels (**.bmp**, **.jpeg**, **.tif**, **.eps**,...).

4.2 Graphiques 3D.

Le traçage des graphiques 3D est illustré dans les deux exemples suivants.

Exemple 7 :

```
>> t = 0:0.05:25;  
>> x = exp(-0.05*t).*cos(t);  
>> y = exp(-0.05*t).*sin(t);  
>> z = t;  
>> plot3(x,y,z), grid
```

Exemple 8 :

```
» [x,y] = meshgrid(-2:.1:2, -2:.1:2);  
» z = x .* exp(-x.^2 - y.^2);  
» mesh(z)
```