Module : Informatique 3
Année : 2ème Année.

TP INFORMATIQUE 3

1- Objectif : Ce TP a pour but de vous familiariser avec l'usage de logiciel MATLAB de la compagnie Mathworks et à la programmation dans cet environnement. L'idée est de vous exposer les bases de cet outil de travail. Le nom MATLAB vient de la contraction MATrix LABoratory, ce qui signifie que toutes les variables sont considérées comme des matrices.

2- Présentation général

Une variable scalaire est vue par MATLAB comme une matrice 1x1 (une ligne, une colonne). Comme le montre l'exemple suivant dans lequel on affecte à la variable x la valeur 3 et on demande ensuite ses dimensions par la commande size :

```
Command Window
>> x=3

x =

     3

>> size(x)

ans =

     1     1

fx >>
```

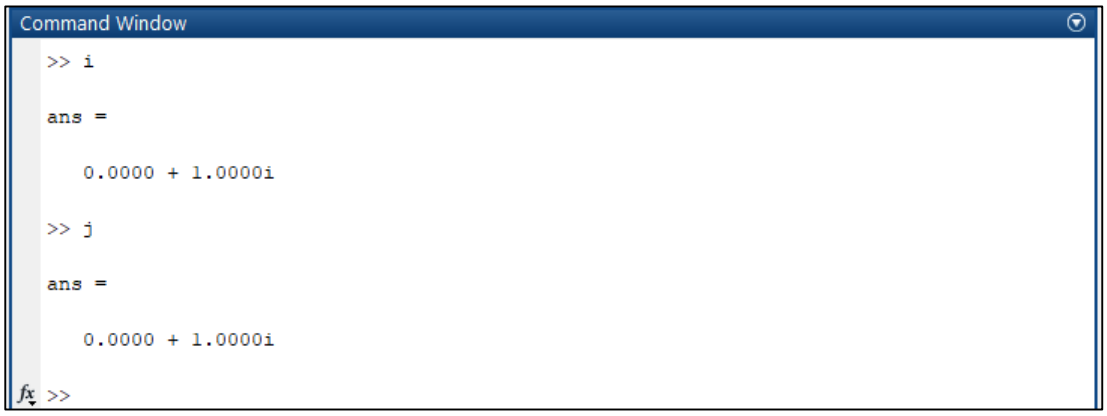
Afin d'éviter ceci, il suffit de suivre la commande par un point-virgule, ce qui permettra éventuellement des programmes plus rapides (pas de perte de temps du à l'affichage).

```
Command Window
>> x=5;
>> size(x)

ans =

     1     1

fx >>
```

MATLAB, en tant que langage scientifique, a prévu des constantes prédéfinies :

```
Command Window
>> pi

ans =

    3.1416

>> eps

ans =

   2.2204e-16

fx >>
```

Les variables i, j représentent le nombre imaginaire :



L'utilisateur peut affecter donc des valeurs à des variables et affecter des opérations à ces variables (+ - / *..)

```
>> x=3 ;
>> y=2 ;
>> x + y
ans =
       6
>> ans + 5
ans =
      11
```

Lorsque l'utilisateur ne fixe pas de variable de sortie, MATLAB place le résultat d'une opération dans « ans ». Cette variable temporaire peut bien sûr être utilisée pour un calcul suivant comme le montre l'exemple précédent. Il est toujours possible de connaître les variables utilisées et leur type à l'aide de la fonction whos.

La solution de x+y a donc été perdue. Il est donc préférable de toujours donner des noms aux variables de sortie :

```
>> a= x + y
 a =
      5
```

La fonction clear permet d'effacer des variables. Par exemple :

```
>> clc          % On efface le contenu de la fenêtre de commande
>> clear  x     % On efface x de la mémoire
>> x
   Undefined function or variable 'x'.
```

Le signe de pourcentage (%) permet de mettre ce qui suit sur une ligne en **commentaire** (MATLAB n'en tiendra pas compte à l'exécution).

3- Opérations mathématique avec MATLAB : Scalaires, vecteurs, matrices

L'élément de base de MATLAB est la **matrice**. C'est-à-dire qu'un scalaire est une matrice de dimension 1x1. Un vecteur colonne de dimension n est une matrice nx1. Une vectrice ligne de dimension n est une matrice 1 x n.

Les scalaires se déclarent directement, par exemple :

```
>> a = 3 ;
>> a
a =
   3
```

Les vecteurs lignes se déclarent de la manière suivante :

```
>> VL = [ 2   1   5 ]
VL =
    2     1     5
>> VL = [ 2, 1, 5]
VL =
    2     1     5
>> size ( VL )
ans =
    1     3
```

On sépare les éléments par des espaces ou des virgules.
Comme chaque élément de MATLAB, le vecteur est une matrice. Ici c'est une matrice 1x3.
1 : Nombre de ligne
3 : Nombre de colonne
Les vecteurs colonnes se déclarent de la manière suivante :

```
>> VC= [ 2 ; 0 ; 3 ]
VC =

   2
   0
   3
>> size ( VC )
ans =
   3    1
```

On sépare les éléments par des points-virgules ou on utilise le retour chariot.
Ici le vecteur est une matrice 3x1
La plus grande dimension d'un vecteur constitue sa longueur (length) :

```
>> length ( VC )
ans =
   3
```

Il est possible de transposer un vecteur à l'aide de la fonction « transpose » ou avec l'apostrophe (').

```
>> V = transpose (VC)

V =

   2    0    3
```

Ou

```
>> V = VC '
V =
   2    0    3
```

Le double point (:) est l'opérateur d'incrémentation dans MATLAB. Ainsi, pour créer un vecteur ligne de **valeurs de 0 à 10 par incrément de 2**, il suffit d'utiliser :

```
>> x = [ 0: 2: 10]
x =
   0    2    4    6    8   10
```

On peut éviter de mettre les crochets si les composants d'un vecteur varient d'un pas constant :

```
>> x =  0: 2: 10
x =
   0    2    4    6    8   10
```

Si l'incrément est de 1, le pas n'est pas noté. On met le (:) uniquement entre le premier et le dernier élément

```
>> y =  0: 10
y =
   0    1    2    3    4    5    6    7    8    9    10
```

On peut accéder à un élément d'un vecteur et même modifier celui-ci directement (ex : le troisième élément du vecteur ligne y et le remplacé par l'élément 7) :

```
>> y (3)
ans =
     2
>> y (3) = 7
y =
   1    2    7    4    5    6    7    8    9   10
```

La suppression d'un élément d'un vecteur, par exemple :

```
>>  x = [ 1 : 2 : 10 ]
x =
   1    3    5    7    9
>>  x ( 4 ) = [ ]              % suppression de l'élément d'indice 4
x  =
    1    3    5    9
```

On peut ajouter une valeur à l'élément d'indice i du vecteur, par exemple :

```
>>  x = [ 1  3  5  9 ]
>> x(2)=x(2)+10                % on ajoute la valeur 10 à l'élément d'indice 2 du vecteur x
x =
   1   13    5    9
```

On obtient la valeur moyenne d'un vecteur par la fonction « mean »:

```
>>  x = [ 1  3  5  9 ]
x =
   1    3    5    9
>>  m  = mean (x)              % la valeur moyenne du vecteur x
m  =
      4.5000
```

Les opérations usuelles d'addition, de soustraction et de multiplication par scalaire sur les vecteurs sont définies dans MATLAB :

```
>> x1 = [2   3];
>> x2 = [1   4];
>> x1+x2     % addition de vecteurs
ans =
    3     7
>> x2 - x1    % soustraction de vecteurs
ans =
   -1     1
>> x3 = 3 * x1   % multiplication par un scalaire
x3  =
    6     9
```

Dans le cas de la multiplication et de la division, il faut faire attention aux dimensions des vecteurs en cause. Pour la multiplication et la division élément par élément, on ajoute un point devant l'opérateur (.* et ./).
Exemple :

```
>> x1.*x2                  % multiplication élément par élément
ans =
   2   12
>> x1./x2                  % division élément par élément
ans =
   2.0000    0.7500
```

Cependant, MATLAB lance une erreur lorsque les dimensions ne concordent pas (remarquez les messages d'erreur, ils sont parfois utiles pour corriger vos programmes) :

```
>> x4=[1 3 4];
>> x5=x1.*x4
Error using  .*
Matrix dimensions must agree.
```

La multiplication de deux vecteurs est donnée par (*). Ici, l'ordre a de l'importance (et la taille aussi):

```
>> v1 =  [2   3];          % vecteur 1x2
>> v2 =  [1   4];
>> v3 =  v2' ;             % vecteur 2x1
>> v = v1 * v3             % (1x2) * (2x1) = (1x1)
 v  =
    14
>> v = v3 * v1             % (2x1) * (1x2) = (2x2)

v  =

    2     3
    8    12
```

Il est aussi possible de concaténer des vecteurs. Par exemple :

```
>> v1 = [2   3];
>> v2 = [1   4];
>> v = [v1   v2]
v =
    2    3    1    4
```

4- Polynômes

Pour MATLAB, un polynôme est une liste : la liste des coefficients ordonnés par ordre décroissant :

Exemple :

Le polynôme $p(x) = 1 - 2x + x^{2+} 4x^3$ est représenté par :

```
>> p = [ 4   1   -2   1]
p =
  4   1   -2   1
>> polyval (p, 0)  = [ 4   1   -2   1]     % polyval permet d'évaluer le polynôme en un point ou des point donnés.
ans =
       1
>> polyval (p , [1  2] )
ans =
   4   33
```

Le calcul approché de racines de polynôme s'effectue en Matlab avec la commande roots :

```
>> p = [ 4   1   -2   1] ;
>> roots(p)                    % Calcul des racines du polynôme p.
ans =
 -1.0000 + 0.0000i
  0.3750 + 0.3307i
  0.3750 - 0.3307i
```

La multiplication de deux polynômes se réalise via la commande **conv (x1,x2)**, c'est à dire les coefficients du produit des deux polynômes par exemple :

```
>> x1 = [2   3];
>> x2 = [1   4];
>> conv (x1, x2)
ans =
      2   11   12
```

5- Graphiques simples

MATLAB en plus de ses grandes possibilités de calcul numériques produit des graphiques en 2 ou 3 dimensions. On ne s'intéressera ici qu'au graphique 2D simple.
La fonction plot (x,y) permet de tracer une courbe liant un ensemble de valeurs (vecteur) y en fonction d'un une autre vecteur x (bien entendu de même dimension).
Dans l'exemple suivant on se propose de tracer la fonction suivante :

$y=\cos(2x) +2 \sin(0.1x)$.
La variable x est un vecteur dont les valeurs vont de $-\pi$ à $+\pi$ avec un pas de $\pi/100$.

```
>> x = - pi : pi/200 : pi;
>> y = cos ( 2 * x ) + 2 * sin ( 0.1 * x );
>> plot ( x , y )
>> grid
>> xlabel ( ' variable x ' )
>> ylabel ( ' variable y ' )
>> title ( ' y =cos ( 2 x)+2 sin ( 0.1 x ) ' )
```

Ce qui produit la sortie graphique de la figure suivante :