Université Mohamed Boudiaf, M'sila
Faculté de technologie
Socle commun

Module : TP-informatique 3
2ième année ST

TP3- Initiation à MATLAB

1. Introduction

Matlab est un environnement puissant au calcul scientifique. Il intègre le calcul matriciel et l'analyse numérique dans ses fonctions de base. De plus son architecture graphique orientée objets permet la génération de graphiques d'excellente qualité.
Matlab tire son nom de l'anglais et constitue une contraction de ***MAT***rix ***LAB***oratory. C'est donc un logiciel qui a été développé pour traiter spécifiquement les problèmes nécessitant un formalisme matriciel important.
Le logiciel *MATLAB* était intuitivement codé en Fortran. Au fil des années, *Matlab* a bien sûr s'est enrichi d'un grand nombre de fonction, les domaines couverts ont augmenté et le langage C a remplacé le Fortran. De plus pour optimiser les performances du logiciel, de nombreuses routines internes ont directement été écrites en assembleur.
On ce conçoit plus aujourd'hui d'enseignements d'automatique de traitement du signal, …, sans y associer des travaux pratiques sous *Matlab*.

2. Structure

Il comprend le langage *Matlab* et intègre de nombreuses fonctions mathématiques codées en langage *Matlab* sous forme *.m* (les *m-files*) par exemple, la fonction *abs.m* permet d'obtention de la valeur absolue d'un nombre.
Les bibliothèques de fonction de *Matlab* spécialisées permettent de personnaliser l'environnement de travail.

3. Commande *Help*

La commande help est l'une des commandes les plus utiles pour commencer l'apprentissage de *Matlab*, elle donne accès à l'aide en ligne. Pour l'utiliser, on a le choix entre plusieurs méthodes.
Exp.

```
>> help plot
PLOT   Linear plot.
   PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix
>> help graph2d
    Two dimensional graphs
```

Remarque : La connaissance du mot-clef anglais correspond à ce que l'on cherche.

4. Editeur ligne ou fichier

4.1. Editeur ligne : On peut travailler dans l'environnement *Matlab* en ligne (ligne à ligne) ou bien exécutant un fichier écrit à l'avance.
Le mode ligne à ligne est celui que l'on a rencontré jusqu'à présent, on tape à la suite du prompt ' »'dans la fenêtre de commande la commande que l'on souhaite exécuter suivie d'un retour chariot (entrée).

Matlab est un langage interprété et le résultat est immédiatement disponible dans l'espace de travail.

4.2. Création d'un fichier

Comme dans tout langage de programmation, on peut écrire un programme puis en demandant l'exécution. Exp : on réalise un programme en langage *Matlab* qui consiste à écrire toutes les commandes au moyen d'un éditeur de texte. Une fois le programme tapé le fichier est sauvé avec un certain nom et une extension *.m* pour que *Matlab* reconnaisse un *M-file* lorsqu'on demande l'exécution de ce programme.

5. Données par défaut

- Le langage *Matlab* est donc conçu pour une manipulation aisée des matrices. C'est dans cette optique que par défaut l'élément de base est la matrice.
- Les données peuvent être en majuscules ou minuscules, *Matlab* différencie les deux variables a et A.

```
>> a=5 ;
>> A=2 ;
>> c=A*a
c =
   10
>>        ←Matlab rend la main
```

- Toute donnée entrée en mode ligne est stockée dans l'espace de travail tant qu'on ne l'efface pas ou qu'on ne l'écrase pas (clear all).

```
>> clear all
>>        ←Matlab rend la main
```

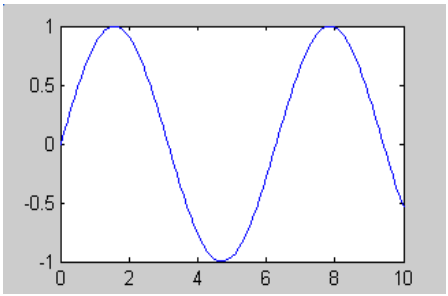
- On visualise une variable en tapant simplement le nom de la variable suivie de return '↵'.
- Pour créer une matrice, chaque élément est séparé de son voisin par un espace, chaque ligne est séparée de la précédente par un point virgule ' ;' ou un return '↵', tout les éléments dans deux crochets '[ ]'.

```
>> A=[2 -3;1 5]
A =
    2   -3
    1    5
>> A=[2 -3
1 5]
A =
    2   -3
    1    5
>>        ← Matlab rend la main
```

6. Affichage graphique

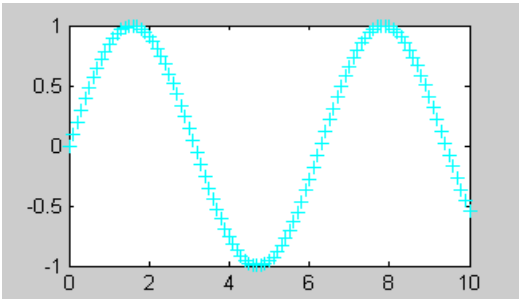
Les résultats présentés sous forme graphique, sous *Matlab* n'est pas un casse-tête les possibilités graphique sont énormes et de nombreuses fonctions toutes prêtes les rendent accessibles.
Voyons quelques commandes, traçons une courbe très simple $y(t) = \sin(t)$ sur un horizon temporel de 10 seconds.

Exp. *>> t=[0:0.1:10]; % t=[début:pas:fin]*
>> y=sin(t); % on donne la fonction y(t)=sin(t)
>> plot(t,y,'b'); % tracer la courbe y en fonction de t, 'b' c'est la couleur bleu
>> ←Matlab rend la main



En utilisant la commande plot pour tracer la courbe y(t) un extrait du help de cette commande nous indique comment l'utiliser *>> help plot.*
'b' indique la couleur de cette courbe (r : red, g : green, w : white, k :black, y : yellow, m : magenta ….).

Exp. *>> plot(t,y,'c+'); % tracer la courbe en cyan avec toute la ligne de (+)*



7. Commandes usuelles

7.1. Commande générales

a). Gestion de l'espace de travail

- Il arrive couramment que l'on se demande dans quel répertoire on se trouve et si ce dernier n'est pas le bon, que l'on veut changer ou bien que l'on cherche tel ou tel fichier.
- ***what*** : donne la liste des fichiers générés dans l'environnement *Matlab (.m, .mat, .mdl, .mex)* dans le répertoire courant.

Exp.

```
>> what
M-files in the current directory C:\MATLAB6p5\work
Untitled   mysqrt    zz       zzz
MDL-files in the current directory C:\MATLAB6p5\work
RLC     RLC2     RLC3
>>
```

- ***which*** : permet la localisation des fichier .m .mdl .mex …dans le répertoire courant.
- ***who*** : donne la liste des variables dans l'espace de travail.
- ***whos*** : donne la taille, la dimension des variables.
- ***size*** : donne la taille d'une variable présentée dans l'espace de travail.

```
>> A=12
A =
   12
>> size(A)
ans =
    1    1
>>      ←Matlab rend la main
```

- ***length*** : donne la longueur d'un vecteur (son nombre d'éléments)
- ***format*** : permet de choisir le format d'affichage des données l'écran.(format short, format long, ….)
- ***clc*** : rafraîchit la fenêtre de commande et ramène le curseur en haut de la page.
- ***clear*** : suppression des variables présentes dans l'espace de travail.
- ***clf*** : efface l'image présente sur la fenêtre graphique active.

- ***quit (exit)***: ferme une session *Matlab* sans sauver le contenu de l'espace de travail.

b). Gestion de cas

On a deux types d'outils : les instructions conditionnelles et les boucles que l'on peut utiliser séparément ou on l'imbriquer comme on va le voir.

b).1. Cas conditionnels

If : permet d'exécution conditionnelle d'une séquence d'actions.

```
If expression
   Action 1 ; Action 2 ; …. Action n
end
```

Si expression est vérifiée, les actions 1 à n sont exécutées sinon elles ne le sont pas et on passe directement à la commande située immédiatement après ***end***.

Else : permet la combinaison de deux séquences d'actions différentes au sein de ***if***.

```
if expression
   Action 1 ; Action 2;
Else Action 1 ; Action 2; end.
```

Remarque : la commande ***else*** est toujours utilisée avec ***if***.

else-if : permet l'indication de plusieurs cas de type if au sein d'un ***if***.

```
If expréssion1
   Action 1 ; Action 2;
else if expression2
   Action 1 ; Action 2;
end.
```

b).2. Boucles

For : permet de répéter une séquence d'action un nombre de fois fixé par l'installation.

```
For variable = … …
  … … … … ;
  … … … … ;
end
```

while : permet la répétition d'une séquence d'action un nombre de fois indéterminé mais conditionné à la réalisation d'une occurrence particulière

Exp.

```
while expression
   action1 ; action2 … ;end
```

```
i=1
while i~=5
disp('bonjour'); i=i+1; end.
```

8. Operations arithmétiques

Nous passons rapidement sur les opérateurs arithmétiques standards et nous attardons un peu plus sur ceux qui sont spécifiques.

Affectation(=) :

```
>> a=2;  % scalaire
>> A=[1 2;-3 -5];  % une matrice
>> t=[0:0.1:20];  % intervalle de temps.
```

Addition (+,-) :

```
>> A=C+B;  % même dimension, soit C et B connues
>> A=C-D;
```

Multiplication (* ou .*) :

Multiplication au sens habituel

Multiplication élément par élément.

*A=C*B* on multiplie la matrice B par C et on affecte le résultat à A.

*A=B.*C* on multiplie chaque élément de B par l'élément correspondant de C et on affecte le résultat à A.

9. Polynômes

Soit le polynôme de la forme $a_m x^m + a_{m-1}x^{m-1} + ... + a_1 x + a_0$ a est représenté par *Matlab* comme $[a_m \quad a_{m-1} \quad ... \quad a_0]$, par exemple $ax^2 + x - 5$ est représenté par $[4 \;\; 1 \;\; -5]$.

Pour chercher les solutions de ce polynôme en appliquant la commande ***roots*** qui permet de calculer les solutions de tel polynôme.

```
>> v=[1 -3 -4]; % le polynôme est : y(x)=x^2-3*x-4
>> roots(v)
ans =
    4
   -1
>>
>> polyval(v,4) % pour calculer y(4)
```

Résoudre un système d'équations

Soit le système des équations suivantes :

$$\begin{cases} a_1x_1 + b_1x_2 + c_1x_3 = d_1 \\ a_2x_1 + b_2x_2 + c_2x_3 = d_2 \\ a_3x_1 + b_3x_2 + c_3x_3 = d_3 \end{cases}$$

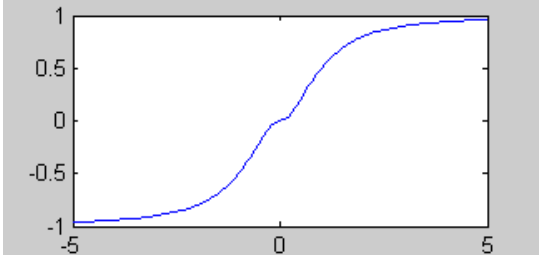
On peut calculer les solutions x_i comme suit $Ax = b$ *donc* $x = b/A = A\backslash b$ ou $x = A\text{^}{-1}*b$

Exp.
```
>> A=[1 1/2 1/3;1/2 1/3 1/4;1/3 1/4 1/5];
>> b=[1/4;1/5;1/6];
>> x=A^-1*b
x =
   0.0500
  -0.6000
   1.5000
>> x=A\b
x =
   0.0500
  -0.6000
   1.5000
>>
```

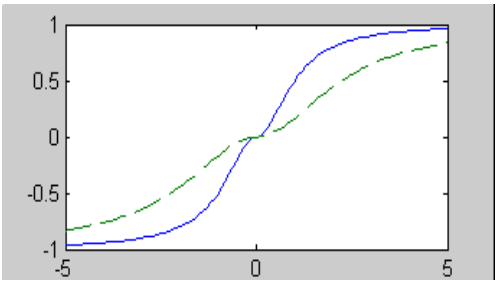
10. Graphismes

Considérons le problème de graphisme d'une fonction, par exemple la fonction suivante $f(x) = x|x|/(1+x^2)$ dans l'intervalle [-5 5].

```
>> x=(-5:.1:5)';
>> y=x.*abs(x)./(1+x.^2);
>> plot(x,y,'-')
>>
```



```
>> y1=x.*abs(x)./(5+x.^2);
>> plot(x,y,'-',x,y1,'--')
```



Exercice – Tracer la courbe de la fonction suivante :

$$f(x)=\frac{1}{(x-0.3)^2+0.01}+\frac{1}{(x-0.9)^2+0.04}-6$$