



الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة وهران 2 محمد بن احمد
Université Oran 2 Mohamed Ben Ahmed

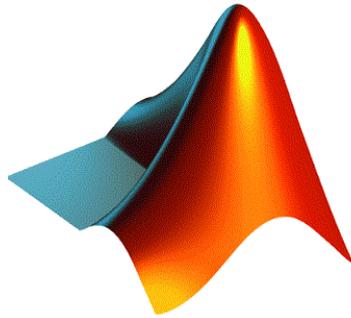
معهد الصيانة والأمن الصناعي
Institut de Maintenance et de Sécurité Industrielle

Polycopié de cours

INFORMATIQUE 3

Auteur : Dr Mohammed Chennoufi

2^{ème} année Licence



MATLAB

Avant-propos

Avant-propos

Ce document est un support pédagogique du cours et des TP destiné aux étudiants de 2^{ème} Année des licences assurées au trois départements: Maintenance en Electromécanique, Maintenance en Instrumentation et Hygiène et Sécurité Industrielle aux seins de l'institut de maintenance et de sécurité industrielle. Dans ce cours on s'intéresse a la programmation avec le logiciel Matlab, plusieurs chapitres sont présentés afin de bien maîtrisé ce langage. Matlab est un logiciel de calcul numérique produit par MathWorks ([http:// www.mathworks.com/](http://www.mathworks.com/)). Il est simple et très efficace, optimisé pour le traitement des matrices, il contient une interface graphique puissante. On peut enrichir Matlab en ajoutant des boîtes à outils (toolbox) qui sont des ensembles de fonctions supplémentaires, profilées pour des applications particulières (traitement de signaux, analyses statistiques, optimisation, etc.).

Le contenu de ce cours intitulé « Informatique 3 » résumé en cinq chapitres suit conformément le programme ministériel de l'enseignement supérieur et de la recherche scientifique proposé aux étudiants de 2^{ème} année

Le cours de module Informatique 3 est constitué de cinq chapitres organisés comme suit :

Chapitre 1: Initiation au langage Matlab

Chapitre 2: Les vecteurs et les matrices

Chapitre 3: La programmation avec Matlab

Chapitre 4: Graphisme

Chapitre 5: Annexe

A la fin de chaque chapitre, l'étudiant obtient des connaissances solides sur la programmation sous Matlab ou Scilab. De plus ce cours est une base au plusieurs cours de méthodes numérique qui seront enseignés durant le cursus de graduation de nos étudiants.

Table des matières

Table des figures	IV
Préambule	1
- Introduction.....	2
- Les mots clés.....	2
- L'organisation du polycopie.....	3

CHAPITRE 1

Initiation au langage Matlab.....	4
1.1 Introduction	5
1.2 L'environnement MATLAB	7
1.2.1 Première interaction avec MATLAB.....	8
1.2.2 Les nombres en MATLAB.....	10
1.2.3 Les principales constantes, fonctions et commandes	12
1.2.4 La priorité des opérations dans une expression.....	14
TP1.....	15

CHAPITRE 2

Les vecteurs et les matrices	17
2.1 Introduction	17
2.2 Les vecteurs.....	17
2.2.1 Définir un vecteur.....	12
2.2.2 Accés aux éléments d'un vecteur	22
2.2.3 La fonction Linespace	22
2.3 Les matrices.....	22
2.3.1 Extraction des éléments d'une matrice.....	24
2.3.2 Modifier les éléments d'une matrice	25

2.3.3	Arithmétique des matrices.....	25
2.3.3.1	<i>Opérations élément par élément.....</i>	26
2.3.3.2	<i>Comparaison des Matrices.....</i>	27
2.3.3.3	<i>Fonctions any et all.....</i>	27
2.3.3.4	<i>Matrices Particulière.....</i>	28
2.3.4	<i>Fonctions utiles pour le traitement des matrices.....</i>	31
2.3.5	<i>Opérations entre matrices.....</i>	33
2.3.6	<i>Caractéristiques des matrices.....</i>	34
TP 2	34

CHAPITRE 3

La programmation sous Matlab.....	37
3.1 Introduction.....	37
3.2 Fichier Script.....	38
3.2.1 <i>Définition.....</i>	38
3.2.2 <i>Utilisation de la fenêtre script ou « M.File ».....</i>	38
3.3 Entrée Sortie.....	41
3.3.2 <i>Lecture des données (les entrées).....</i>	41
3.3.2 <i>Ecriture des données (les sorties).....</i>	41
3.4 Les expressions logiques.....	42
3.4.1 <i>Les opérations de comparaison.....</i>	42
3.4.2 <i>les opérations logiques.....</i>	42
3.4.3 <i>Fonctionnement les opérations logiques.....</i>	43
3.4.4 <i>Comparaison des matrices et des vecteurs.....</i>	44
3.5 Les insructions de controles.....	44
3.6 Les boucles.....	46
6.6.1 <i>La boucle for.....</i>	46
6.6.2 <i>La boucle while.....</i>	49
TP 3.....	51

CHAPITRE 4

Les Graphiques sous Matlab	53
Introduction	55
4.1 Graphique 2 dimensions (2D).....	55
4.1.1 La fonction Plot.....	55
4.1.2 Modification de l'apparence d'une courbe.....	58
4.1.3 Anotation d'une figure.....	61
4.2 Tracage de plusieurs courbes sous Matlab	62
4.2.1 <i>Sur la même figure avec la même échelle.....</i>	<i>62</i>
4.2.2 <i>afficher plusieurs graphiques dans la même fenêtre.....</i>	<i>63</i>
<i>TP 4</i>	<i>66</i>

CHAPITRE 5

Annexe1	69
----------------------	-----------

<u>Bibliographie</u>	70
-----------------------------------	-----------

Table des figures

Chapitre 1) MAITRISE DU LOGICIEL MATLAB

1	Les fichiers Matlab.....	6
2	L'environnement Matlab	7

Chapitre 3) La programmation sous Matlab

3.1	Ouverture d'un nouveau fichier « M.file ».....	38
3.2	Fenêtre d'un fichier « script » ou « M.file ».....	39
3.3	Enregistrement d'un fichier script.....	39
3.4	Une fenêtre d'enregistrement.....	40
3.5	Exemple 3.1.....	40
3.6	Solution de l'exemple 3.6	45
3.7	Solution de 3.7	47
3.8	Résultat d'exécution de l'exemple 3.7	47
3.9	Solution de l'exemple 3.8.....	48
3.10	Résultat d'exécution de l'exemple 3.8.....	48

Chapitre 4) Les Graphiques sous Matlab

4.1	Fenetre graphique exemple 1	55
4.2	Fenêtre graphique (Exemple 2).....	56
4.3	Fenêtre graphique (Exemple 3).....	57
4.4	Fenêtre graphique (Exemple 4).....	58
4.5	Courbe avec couleur rouge, en ligne plein avec des losanges.....	60
4.6	Courbe avec couleur cyan, en pointillé avec des triangles supérieur.....	60
4.7	plusieurs courbes sur la même échelle.....	64
4.8	Plusieurs graphiques avec la commandesubplot.....	65

Préambule

Préambule

Introduction

Les mots clés

L'organisation du polycopié

Introduction

Afin de bien maîtriser les TP du module informatique 3, J'ai pensé de présenter un rappel de cours plus précisément sur le **langage Matlab , vecteurs et matrices, la programmation sous matlab et les graphismes**. A la fin de chaque chapitre des exercices avec solution et des TP sont présentés (une annexe qui résume un exemple élémentaire **de script** qui pourra être utile aux étudiants lors des séances de TP et des sujets **d'examens de synthèses et de rattrapages**).

Le contenu de ce polycopié de cours intitulé «Informatique 3 » résumé en cinq chapitres **suit conformément le programme ministériel** de l'enseignement supérieur et de la recherche scientifique proposé aux étudiants de 2^{ème} année LMD avec un Volume horaire hebdomadaire : 01h30mn de TP) ; Coefficient : 01 ; Crédit : 2.

Les mots clés

Matrice, Matlab, Vecteur, Plot, for, while , script

L'organisation du polycopié

Le cours et les TP du module Informatique 3 que j'ai enseigné durant ces 10 dernières années pour les étudiants de deuxième année licence et rapporté dans ce polycopie est décomposée en cinq chapitres organisés comme suit :

Une Initiation au langage Matlab est présentée au premier chapitre. J'ai abordé en deuxième chapitre Les vecteurs et les matrices. Le troisième chapitre est dédié à la programmation avec Matlab. Le quatrième chapitre présentera les graphismes. Nous terminerons par une annexe en cinquième chapitre qui résume des sujets d'examens de synthèses et de rattrapage et un script en Matlab que j'ai proposé dans notre établissement IMSI.

Chapitre 1: Initiation au langage Matlab

Sommaire

- 1.1** **Introduction**
- 1.2** **L'environnement MATLAB**
 - 1.2.1 Première interaction avec MATLAB
 - 1.2.2 Les nombres en MATLAB
 - 1.2.3 *Les principales constantes, fonctions et commandes*
 - 1.2.4 La priorité des opérations dans une expression
- 1.3** **TPI**

1.1 Introduction

MATLAB (MATrix LABoratory) est un environnement de programmation interactif pour le calcul scientifique, la programmation et la visualisation des données. Il est basé sur le type de variable matricielle (le type de donnée basic au niveau de MATLAB c'est la matrice). Il dispose d'une syntaxe spécifique avec ses fonctions spécialisées.

Il est très utilisé dans les domaines d'ingénierie et de recherche scientifique, ainsi qu'aux établissements d'enseignement supérieur. Sa popularité est due principalement à sa forte et simple interaction avec l'utilisateur mais aussi aux points suivants :

- Sa richesse fonctionnelle : avec MATLAB, il est possible de réaliser des manipulations mathématiques complexes en écrivant peu d'instructions. Il peut évaluer des expressions, dessiner des graphiques et exécuter des programmes classiques. Et surtout, il permet l'utilisation directe de plusieurs milliers de fonctions prédéfinie.
- La possibilité d'utiliser les boites à outils (toolboxes) : ce qui encourage son utilisation dans plusieurs disciplines (simulation, traitement de signal, imagerie, intelligence artificielle,...etc.).
- La simplicité de son langage de programmation : un programme écrit en MATLAB est plus facile à écrire et à lire comparé au même programme écrit en C ou en PASCAL.

- Sa manière de tout gérer comme étant des matrices.

A l'origine MATLAB était conçu pour faire principalement des calculs sur les vecteurs et les matrices d'où son nom 'Matrix Laboratory', mais par la suite il a été amélioré et augmenté pour pouvoir traiter beaucoup plus de domaines.

MATLAB n'est pas le seul environnement de calcul scientifique existant car il existe d'autres concurrents dont les plus importants sont MAPLE et MATHEMATICA. Il existe même des logiciels libres qui sont des clones de MATLAB comme SCILAB et OCTAVE.

Il existe deux modes de fonctionnement :

1. mode interactif : MATLAB exécute les instructions au fur et à mesure qu'elles sont données par l'utilisateur. C.-à-d. directement au clavier depuis la fenêtre de commande.
2. mode exécutif : MATLAB exécute ligne par ligne un "fichier M" (programme en langage MATLAB). C.- à-d. sous forme de séquences d'expressions ou scripts enregistrées dans des fichiers-texte appelés m-files et exécutées depuis la fenêtre de commande.

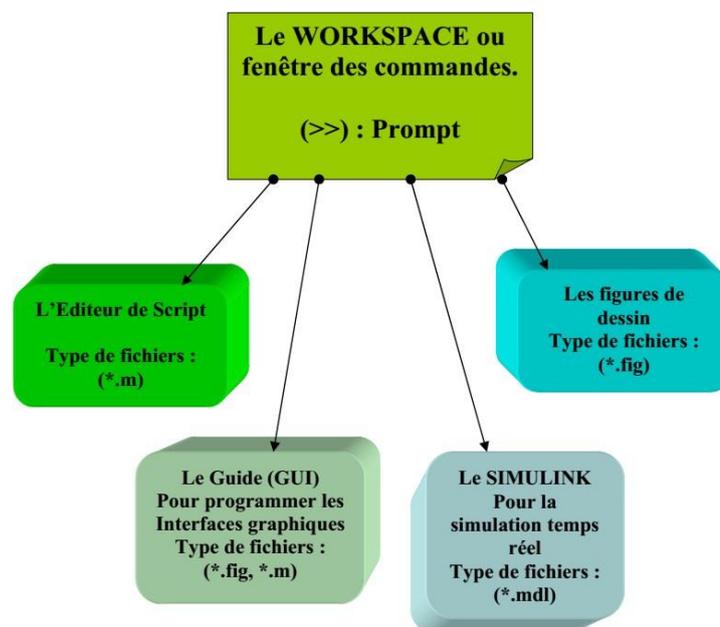


Figure 1 : Les fichiers Matlab

1.2 L'environnement MATLAB

Au démarrage de Matlab, il affiche plusieurs fenêtres. Selon la version on peut trouver les fenêtres suivantes :

- **Current Folder** : indique le répertoire courant ainsi que les fichiers existants.
- **Workspace** : indique toutes les variables existantes avec leurs types et valeurs.
- **Command History** : garde la trace de toutes les commandes entrées par l'utilisateur.
- **Command Window** : nous utilisons pour formuler nos expressions et interagir avec MATLAB, et c'est la fenêtre que nous utilisons tout au long de ce chapitre.

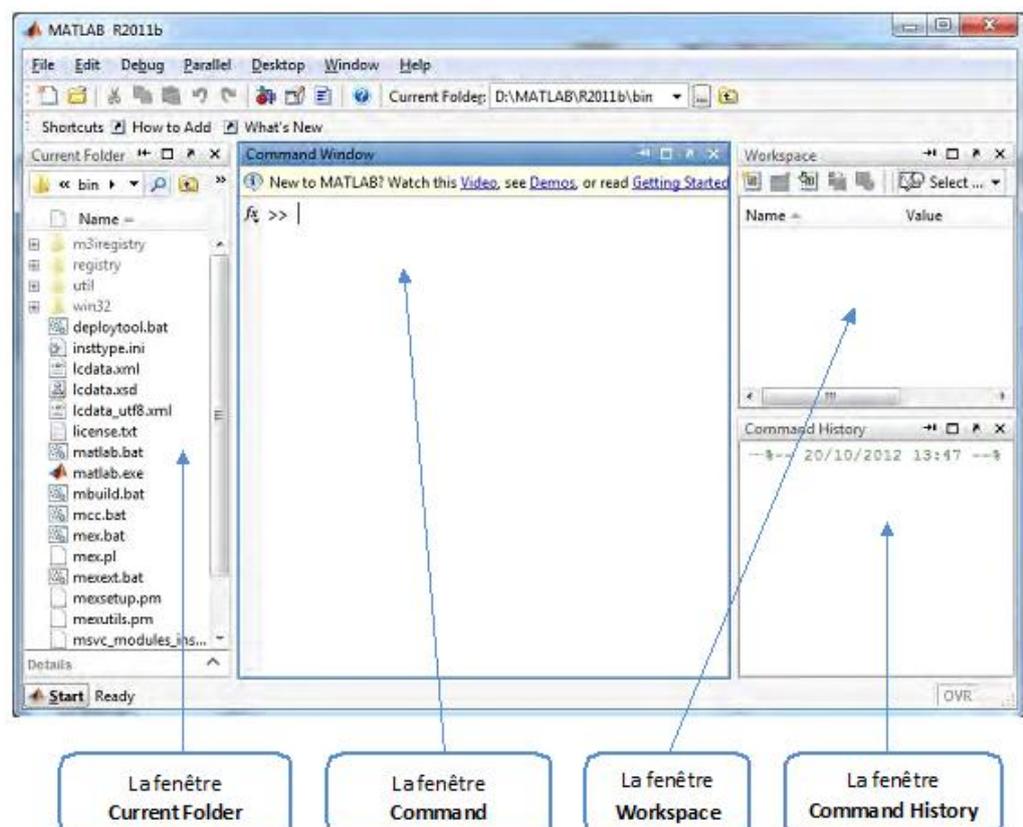


Figure 2 : L'environnement MATLAB

1.2.1 Première interaction avec MATLAB

Le moyen le plus simple d'utiliser MATLAB est d'écrire directement dans la fenêtre de commande (Command Window) juste après le curseur (prompt) >>

Pour calculer une expression mathématique il suffit de l'écrire comme ceci :

```
>> 5+6
```

Puis on clique sur la touche Entrer pour voir le

```
ans =  
11
```

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 :

```
>> 5+6 ;  
^^
```

Pour créer une variable on utilise la structure simple : '**variable** = **définition**' sans se préoccuper du type de la variable.

Par exemple :

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

```
ans =  
1
```

```
>> ans+10
```

```
ans =  
11
```

```
>>
```

Il est possible d'écrire plusieurs expressions dans la même ligne en les faisant séparées par des virgules ou des points virgules. Par exemple :

```
>> 5+6, 2*5-1, 12-4
```

```
ans =
11
ans =
9
ans =
8
```

```
>> 5+6; 2*5-1, 12-4;
```

```
ans =
9
```

```
>>
```

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). Les opérations de base dans une expression sont résumées dans le tableau suivant :

L'opération	La signification
+	L'addition
-	La soustraction
*	La multiplication
/	La division
\	La division gauche (ou la division inverse)
^	La puissance
'	Le transposé
(et)	Les parenthèses spécifient l'ordre d'évaluation

Pour visualiser la liste des variables utilisées, soit on regarde à la fenêtre '**Workspace**' soit on utilise les commandes '**whos**' ou '**who**'.

whos donne une description détaillée (le nom de la variable, son type et sa taille), par contre **who** donne juste les noms des variables.

Par exemple, dans ce cours on a utilisé 3 variables **a**, **u** et **v** :

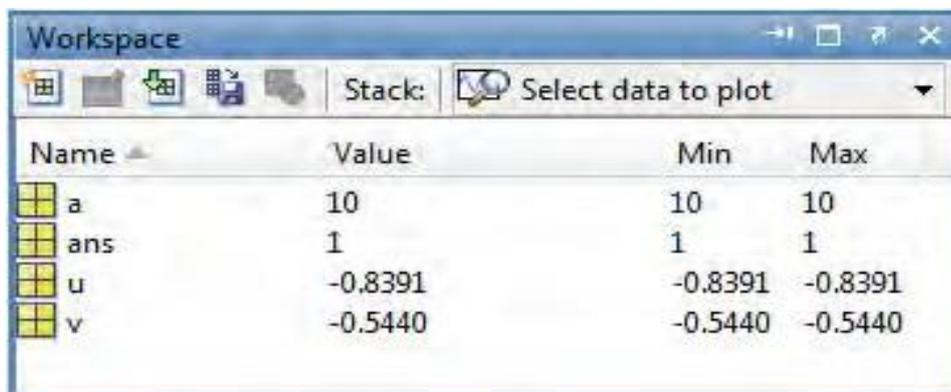
```
>> who
```

```
Your
variables
are:
a  ans  u  v
```

```
>> whos
```

Name	Size	Bytes	Class	Attribute
a	1x1	8	Double	s
ans	1x1	8	Double	
u	1x1	8	Double	
v	1x1	8	Double	

L'utilisation de ces deux commandes peut être omise car des informations sur les variables sont visibles directement dans la fenêtre workspace.



Name	Value	Min	Max
a	10	10	10
ans	1	1	1
u	-0.8391	-0.8391	-0.8391
v	-0.5440	-0.5440	-0.5440

1.2.2 Les nombres en MATLAB

MATLAB utilise une 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 désigner la partie imaginaire. Le tableau suivant donne un résumé:

Le type	Exemples
Entier	5 -83
Réel en notation décimale	0.0205 3.1415926
Réel en notation scientifique	1.60210e-20 6.02252e23 (1.60210x10 ⁻²⁰ et 6.02252x10 ²³)
Complexe	5+3i -3.14159j

MATLAB utilise toujours les nombres réels (double précision) pour faire les calculs, ce qui permet d'obtenir une précision de calcul allant jusqu'aux 16 chiffres significatifs. Mais il faut noter les points suivants :

- Le résultat d'une opération de calcul est par défaut affichée avec quatre chiffres après la virgule.
- Pour afficher davantage de chiffres utiliser la commande **format long** (14 chiffres après la virgule).
- Pour retourner à l'affichage par défaut, utiliser la commande **format short**.
- Pour afficher uniquement 02 chiffres après la virgule, utiliser la commande **format bank**.
- Pour afficher les nombres sous forme d'une ration, utiliser la commande **format rat**.

La commande	Signification
format short	affiche les nombres avec 04 chiffres après la virgule
format long	affiche les nombres avec 14 chiffres après la virgule
format bank	affiche les nombres avec 02 chiffres après la virgule
format rat	affiche les nombres sous forme d'une ration (a/b)

Exemple :

```
>> 8/3
```

```
ans =
     2.6667
```

```
>> format long
```

```
>> 8/3
```

```
ans =
 2.666666666666667
```

```
>> format bank
```

```
>> 8/3
```

```
ans =
     2.67
```

```
>> format short
```

```
>> 8/3
```

```
ans =
    2.6667
```

```
>> 7.2*3.1
```

```
ans =
    22.3200
```

```
>> format rat
```

```
>> 7.2*3.1
```

```
ans =
    558/25
```

```
>> 2.6667
```

```
ans =
    26667/10
    000
```

La fonction **vpa** peut être utilisé afin de forcer le calcul de présenter plus de décimaux significatifs en spécifiant le nombre de décimaux désirés.

Exemple :

```
>> sqrt(2)
```

```
ans =
    1.4142
```

```
>> vpa(sqrt(2),50)
```

```
ans =
1.4142135623730950488016887242096980785696718753769
```

1.2.3 Les principales constantes, fonctions et commandes

MATLAB définit les constantes suivantes :

La constant	Sa valeur
Pi	$\pi=3.1415\dots$
exp(1)	$e=2.7183\dots$
I	$=\sqrt{-1}$
J	$=\sqrt{-1}$
Inf	∞
NaN	Not a Number (Pas un numéro)
Eps	$\varepsilon \approx 2 \times 10^{-16}$.

Parmi les fonctions fréquemment utilisées, on peut noter les suivantes :

Fonction	signification
sin(x)	le sinus de x (en radian)
cos(x)	le cosinus de x (en radian)
tan(x)	le tangent de x (en radian)
asin(x)	l'arc sinus de x (en radian)
acos(x)	l'arc cosinus de x (en radian)
atan(x)	l'arc tangent de x (en radian)
sqrt(x)	la racine carrée de x $\rightarrow \sqrt{x}$.
abs(x)	la valeur absolue de x $\rightarrow x $
exp(x)	$= e^x$
log(x)	logarithme naturel de x $\rightarrow \ln(x)=\log_e(x)$
log10(x)	logarithme à base 10 de x $\rightarrow \log_{10}(x)$
imag(x)	la partie imaginaire du nombre complexe x
real(x)	la partie réelle du nombre complexe x
round(x)	arrondi un nombre vers l'entier le plus proche
floor(x)	arrondi un nombre vers l'entier le plus petit $\rightarrow \min\{n n \leq x, n \text{ entier}\}$
ceil(x)	arrondi un nombre vers l'entier le plus grand $\rightarrow \max\{n n \geq x, n \text{ entier}\}$

MATLAB offre beaucoup de commandes pour l'interaction avec l'utilisateur. Nous nous contentons pour l'instant d'un petit ensemble, et nous exposons les autres au fur et à mesure de l'avancement du cours.

La commande	Sa signification
Who	Affiche le nom des variables utilisées
Whos	Affiche des informations sur les variables utilisées
clear x y	Supprime les variables x et y
clear, clear all	Supprime toutes les variables
Clc	Efface l'écran des commandes
exit, quit	Fermer l'environnement MATLAB
Format	Définit le format de sortie pour les valeurs numériques format long : affiche les nombres avec 14 chiffres après la virgule format short: affiche les nombres avec 04 chiffres après la virgule format bank : affiche les nombres avec 02 chiffres après la virgule format rat : affiche les nombres sous forme d'une ration (a/b)

1.2.4 La priorité des opérations dans une expressions

L'évaluation d'une expression s'exécute de gauche à droite en considérant la priorité des opérations indiquée dans le tableau suivant :

Les opérations	La priorité (1=max, 4=min)
Les parenthèses (et)	1
La puissance et le transposé ^ et ' ^	2
La multiplication et la division * et /	3
L'addition et la soustraction + et -	4

Par exemple $5+2*3 = 11$ et $2*3^2 = 18$

Exercice :

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

- $3x^3-2x^2+4x$

- $\frac{e^{1+x}}{1-\sqrt{2x}}$

- $|\sin^{-1}(2x)|$

- $\frac{\ln(x)}{2x^3} - 1$

La solution :

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

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

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

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

```
>> log(x)/(2*x^3)-1 ;
```

Fiche TP 1

Exercice 1 :

1) Entrer les instructions suivantes dans la fenêtre command window

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

2) Donner le résultat d'affichage de chaque instruction

3) Qu'elle est l'utilité du « ; »

4) Que représente la variable ans

Exercice 2 :

Soit l'instruction suivante tapée dans la fenêtre command window :

```
>>5+6, 2*5-1; 12-4
```

1) Expliquez l'affichage du résultat de cette instruction

2) Tapez les deux commandes who et whos dans la fenêtre command window

3) Qu'elle est la différence entre les deux commandes

Exercice 3 :

1) Supprimez toutes les variables déjà utilisées et effacer le contenu de la fenêtre commandwindow

2) Tapez les instructions suivantes et expliquer le résultat affiché

```
>>8/3  
>>format bank  
>>8/3
```

3) Tapez une commande qui permet d'afficher plus de chiffres après la virgule

4) Expliquez le résultat d'affichage des instructions suivantes :

```
>>sqrt(2)  
>>vpa(sqrt(2),50)
```

Exercice 4 :

1) Supprimez toutes les variables déjà utilisées et effacer le contenu de la fenêtre commandwindow.

2) Entrez les instructions suivantes et expliquer les résultats affichés

```
>>5+2*3  
>>2*3^2  
>>5*2^4+(3-1)
```

3) Créez une variable x et donnez-la la valeur 2, puis écrivez les expressions suivantes :

- $3 * x^3 - 2 * x^2 + 4x$
- $\frac{e^{1+x}}{1-\sqrt{2x}}$
- $|\sin^{-1}(2x)|$
- $\frac{\ln(x)}{2x^3} - 1$

Exercice 5

I- Quelques manipulations basiques sous Matlab

Voici quelques exemples de calculs que vous pouvez taper :

```
>> 2*3+4e5
```

```
>> x=ans; y=sqrt(x)
```

```
>> z=1+i
```

```
>> z^2, w=p*i+j
```

Remarquez que la virgule et le point virgule permettent de séparer des commandes sur une ligne et que le point virgule supprime l'affichage du résultat.

Notez aussi π , i , j pour noter le nombre π , et i , j .

Voici quelques fonctions usuelles (essayez-les):

`abs`, `exp`, `log`, `cos`, `sin`, `tan`, `acos`, `asin`, `atan`, `cosh`, `sinh`, `tanh`, `acosh`, `asinh`, `atanh`, `sqrt`, `floor`, `ceil`.

1- Aide

Pour obtenir la description d'une commande Matlab, vous pouvez taper `help` commande ou utiliser la fenêtre d'aide. Par exemple,

```
>> help format.
```

Essayez ensuite

```
>> format long, pi
```

```
>> format long e, pi
```

puis revenez au format par défaut

Chapitre 2: Les vecteurs et les matrices

Sommaire

2.1 *Introduction*

2.2 *Les vecteurs*

2.2.1 *Définir un vecteur*

2.2.2 *Accès aux éléments d'un vecteur*

2.2.3 *La fonction linspace*

2.3 *Les matrices*

2.3.1 *Extraction des Eléments d'une Matrice*

2.3.2 *Modifier les Eléments d'une Matrice*

2.3.3 *Arithmétique des Matrices*

2.3.3.1 *Opérations élément par élément*

2.3.3.2 *Comparaison des Matrices*

2.3.3.3 *Fonctions any et all*

2.3.3.4 *Matrices Particulière*

2.3.4 *Fonctions utiles pour le traitement des matrices*

2.3.5 *Opérations entre matrices*

2.3.6 *Caractéristiques des matrices*

2.1 Introduction

Nous présentons dans ce chapitre les notions de base de création et de manipulation de vecteurs et de matrices avec des exemples sous MATLAB

2.2 Les vecteurs

Un vecteur sous Matlab est une collection d'éléments du même type. Un vecteur pourra représenter des valeurs expérimentales ou bien les valeurs discrètes d'une fonction continue.

Le vecteur est un cas spécial d'une matrice avec une seule ligne ou une seule colonne, tandis qu'un scalaire est considéré par MATLAB comme une matrice 1x1 et n'a pas besoin de crochets lors de sa saisie.

2.2.1 *Définir un vecteur*

La méthode la plus simple pour définir un vecteur est de donner sa description explicite à l'aide de la commande []

Par défaut, le vecteur est une ligne à plusieurs colonnes

- Vecteur ligne par énumération des composants

```
>> v=[1 3.4 5 -6]
```

```
v= 1.0000  3.4000  5.0000 -6.0000
```

- Vecteur ligne par description

```
>> T=[0 : 10 : 60] → [valeur initiale : incrément : valeur finale]
```

```
T = 0  10  20  30  40  50  60
```

- Vecteur colonne

```
>> co=T'
```

```
col =
```

```
0
10
20
30
40
50
60
```

Exemple :

$\text{vec} = [1 \ 2 \ 4 \ 7 \ 9]$

$\text{vec} =$

1.0000 2.0000 4.0000 7.0000 9.0000

On peut également définir un vecteur colonne en utilisant le ;

$\text{col} = [1 ; 2 ; 4 ; 7]$

$\text{col} =$

1

2

4

7

On peut concaténer deux vecteurs :

$\text{vec1} = [1 \ 3 \ 5];$

$\text{vec2} = [9 \ 10 \ 11];$

$\text{vec} = [\text{vec1} \ \text{vec2}]$

$\text{vec} =$

1 3 5 9 10 11

On peut prendre la transposée pour passer d'une ligne à une colonne ou réciproquement :

$\text{vec1} = [1 \ 3 \ 5];$

$\text{vec} = \text{vec1}'$

$\text{vec} =$

1

3

5

length() permet de retourner la longueur du vecteur.

length(vec)

ans =

3

Exemple:

```
>> Vligne = [2 6 3] %vecteur ligne
```

V =

2 6 3

```
>> Vcolonne = [6 ; 8 ; 5] %vecteur colonne
```

Vcolonne =

6

8

5

```
>> Vcolonne = [6 8 5]' %vecteur colonne également (utilisation de la transposée)
```

Vcolonne =

6

8

5

Si les composants d'un vecteur X sont ordonnés avec des valeurs consécutives, nous pouvons le noter avec la notation suivante :

**X = premier_élément :
dernier_élément**

Les crochets sont facultatifs dans ce cas)

Exemple :

```
>> X = 1:8
```

X =

1 2 3 4 5 6 7 8

```
>> X = [1:8]
```

X =

1 2 3 4 5 6 7 8

Si les composants d'un vecteur X sont ordonnés avec des valeurs consécutives mais avec un pas (d'incréméntation/décrémentation) différent de 1, nous pouvons spécifier le pas avec la notation suivante

**X = premier_élément : le_pas :
dernier_élément**

(Les crochets sont facultatifs)

Exemple:

```
>> X = [0:2:10]    % le vecteur X contient les nombres pairs < 12
```

X =

0 2 4 6 8 10

```
>> X = [-4:2:6]
```

X =

-4 -2 0 2 4 6

```
>> X = 0:0.2:1
```

X =

0 0.2000 0.4000 0.6000 0.8000 1.0000

```
>> V = [ 1:2:5 , -2:2:1 ]
```

V =

1 3 5 -2 0

```
>> A = [1 2 3]
```

A =

1 2 3

```
>> B = [A, 4, 5, 6]
```

B =

1 2 3 4 5 6

2.2.2 Accès aux éléments d'un vecteur

Une fois qu'un vecteur existe on peut accéder à leurs éléments en spécifiant les indices

Nom du Vecteur(indice_élément)

Exemple :

```
>> V = [5 7 9 6 3];
```

```
>> V(2) %donne le 2 ème élément du vecteur
```

```
>> V(3:5) %donne les éléments du 3ème jusqu'au 5ème
```

```
>> V(:) %retourne le vecteur sous forme de vecteur colonne
```

```
>> V(end) %retourne le dernier élément du vecteur
```

2.2.3 La fonction linspace

La création d'un vecteur dont les composants sont ordonnés par intervalle régulier et avec un nombre d'éléments bien déterminé peut se réaliser avec la fonction :

linspace (début, fin, nombre d'éléments)

Le pas d'incrémentation est calculé automatiquement par MATLAB selon la formule :

$$\text{Le pas} = (\text{fin} - \text{début}) / \text{nombre élément} - 1$$

Exemple:

```
>> X = linspace(1,10,4) % un vecteur de quatre élément de 1 à 10
```

```
X =
```

```
1 4 7 10
```

```
>> Y = linspace(13,40,4) % un vecteur de quatre élément de 13 à 40
```

```
Y =
```

```
13 22 31 40
```

2.3 Les matrices

Une matrice est un tableau rectangulaire d'éléments (bidimensionnels). Pour insérer une matrice, il faut respecter les règles suivantes :

Les éléments doivent être mis entre des crochets [et]

Les espaces ou les virgules sont utilisés pour séparer les éléments dans la même ligne

Un point-virgule (ou la touche entrer) est utilisé pour séparer les lignes

Les matrices sont saisies sur une seule ligne où les éléments consécutifs des lignes sont séparés par un espace ou une virgule et les lignes sont séparées par un point virgule, le tout doit être entre crochets.

Exemple:

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

M =

```
2 6 3
```

```
3 8 1
```

```
1 5 3
```

```
>> C1 = [10 20];
```

```
>> C2 = [30 40];
```

```
>> C = [C1 , C2 ]
```

C =

```
10 20 30 40
```

```
>> D = [C1 ; C2 ]
```

D =

```
10 20
```

```
30 40
```

```
>> D = [D ; 50 60]
```

D =

```
10 20
```

```
30 40
```

```
50 60
```

2.3.1 Extraction des Eléments d'une Matrice

L'accès aux éléments d'une matrice se fait en utilisant la syntaxe générale suivante :

nom_matrice (positions_lignes , positions_colonnes)

L'accès à un élément de la ligne i et la colonne j de la matrice A se fait par : $A(i,j)$

L'accès à toute la ligne numéro i se fait par : $A(i,:)$

L'accès à toute la colonne numéro j se fait par : $A(:,j)$

Exemple :

```
>> A = [ 1 2 3 ; 4 5 6]
```

```
A =
```

```
1  2  3
```

```
4  5  6
```

```
>> A(2,1)
```

```
ans =
```

```
4
```

```
>> A(1, :)
```

```
ans =
```

```
1  2  3
```

```
>> A(:, 3)
```

```
ans =
```

```
3  6
```

```
>> A(:, [1,3])
```

```
ans =
```

```
4 3
```

```
4 6
```

2.3.2 Modifier les Eléments d'une Matrice

Exemple :

```
>> B = A;
```

```
>> B(1, 2) = 200
```

```
B =
```

```
1 200 3
```

```
4 5 6
```

```
>> B(2, :) = [400 500 600]
```

```
B =
```

```
1 200 3
```

```
400 500 600
```

```
>> B(:, [1,3]) = [11 33 ; 44 66]
```

```
B =
```

```
11 200 33
```

```
44 500 66
```

2.3.3 Arithmétique des Matrices

Exemple:

```
>> 2 * A + 1
```

```
ans =
```

```
3 5 7
```

```
9 11 13
```

```
>> A + B
```

```
ans =
```

```
2 4 3
```

```
8 10 6
```

```
>> A * B'
```

```
Ans =
```

```
5 14
```

```
14 41
```

```
>> sin(A)
```

```
ans =
```

```
0.8415 0.9093 0.1411
```

```
-0.7568 -0.9589 -0.2794
```

2.3.3.1.1 Opérations élément par élément

Exemple:

```
>> A .* B
```

```
ans =
```

```
1 4 0
```

```
16 25 0
```

```
>> A.^2
```

```
ans =
```

```
1 4 9
```

```
16 25 36
```

```
>> A ./ 2
```

```
ans =
```

```
0.5000 1.0000 1.5000
```

```
2.0 2.5000 3.0000
```

2.3.3.2 Comparaison des Matrices

Exemple:

```
>> A >= 3
```

```
ans =
```

```
    0  0  1
```

```
    1  1  1
```

```
>> B = A ;
```

```
    B(:,3) = 0
```

```
B =
```

```
    1  2  0
```

```
    4  5  0
```

```
>> A == B
```

```
ans =
```

```
    1  1  0
```

```
    1  1  0
```

```
>> A >= B
```

```
ans =
```

```
    1  1  1
```

```
    1  1  1
```

2.3.3.3 Fonctions any et all

any(x) : renvoie 1 si au moins un des éléments de x n'est pas nul :

all(x) : renvoie 1 si tous les éléments de x ne sont pas nuls :

Exemples :

```
>> V = [0 0 4 0] ;
```

```
>> any(V)
```

```
ans =
```

```
1
```

```
>> W = 1: 4;
```

```
>> all(W)
```

```
ans =
```

```
1
```

```
>> all(V)
```

```
ans =
```

```
0
```

2.3.3.4 Matrices Particulière

Il existe des fonctions en MATLAB qui permettent de générer automatiquement des matrices particulières.

`zeros(n)` Génère une matrice $n \times n$ avec tous les éléments = 0

`zeros(m,n)` Génère une matrice $m \times n$ avec tous les éléments

= 0 `ones(n)` Génère une matrice $n \times n$ avec tous les éléments

= 1 `ones(m,n)` Génère une matrice $m \times n$ avec tous les

éléments = 1 `eye(n)` Génère une matrice identité de

dimension $n \times n$ `magic(n)` Génère une matrice magique de

dimension $n \times n$

`rand(m,n)` Génère une matrice de dimension $m \times n$ de valeurs aléatoires

Matrice identité :

Exemple:

```
>> I = eye(3,3)
```

```
I =
```

```
1 0 0
0 1 0
0 0 1
```

Matrice nulle :Exemple:

```
>> Z = zeros(3,2)
```

```
Z =
```

```
0 0 0
0 0 0
```

Matrice unitéExemple:

```
>> U = ones(2,3)
```

```
U =
```

```
1 1 1
1 1 1
```

Matrice vide :Exemple:

```
>> E = []
```

```
E =
```

```
[]
```

```
>> size(E)
```

```
ans =
```

```
0 0
```

```
>> A
```

```
A =
```

```
1 2 3
4 5 6
```

```
>> A(:, 2) = []
```

```
A =
```

```
1 3
```

2 4

Nombres Complexes

Pour Matlab, i (ou j) est le nombre complexe tel que $i^2 = -1$

Exemple:

```
>> i
ans =
    0 + 1.0000i
>> i^2
ans =
    -1
>> z1 = 4 - 3i
z1 =
    4.0000 - 3.0000i
>> conj(z1)
ans =
    4.0000 + 3.0000i
>> z1 * ans
ans =
    25
>> real(z1)
ans =
    4
```

```
>> imag(z1)
ans =
    -3
```

Matrice de nombre complexe

```
>> z1 * ones(2,2)
ans =
    4.0000 - 3.0000i 4.0000 - 3.0000i
    4.0000 - 3.0000i 4.0000 - 3.0000i
```

2.3.4 Fonctions utiles pour le traitement des matrices

Voici quelques fonctions parmi les plus utilisées concernant les matrices

det Calcule de déterminant d'une matrice

inv Calcule l'inverse d'une matrice

rank Calcule le rang d'une matrice

trace Calcule la trace d'une matrice

diag Renvoie le diagonal d'une matrice

tril Renvoie la partie triangulaire inferieure

triu Renvoie la partie triangulaire supérieure

```
>> A = [1,2;3,4] ;
```

```
>> det(A)
```

```
ans =
```

```
    -2
```

```
>> inv(A)
```

```
ans =
```

```
   -2.0000  1.0000
```

```
    1.5000 -0.5000
```

```
>> rank(A)
```

```
ans =
```

```
    2
```

```
>> trace(A)
```

```
ans =
```

```
    5
```

```
>> diag(A)
```

```
ans =
```

```
    1
```

```
    4
```

diag(V) Crée une matrice ayant le vecteur V dans le diagonal et 0 ailleurs.

```
>> V = [-5,1,3]
```

```
>> diag(V)
```

```
ans =
```

```
   -5  0  0
```

```
    0  1  0
```

```
    0  0  3
```

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

```
B =
```

```
1 2 3
4 5 6
7 8 9
```

```
>> tril(B)
```

```
ans =
```

```
1 0 0
4 5 0
7 8 9
```

```
>> tril(B,-1)
```

```
ans =
```

```
0 0 0
4 0 0
7 8 0
```

```
>> tril(B,-2)
```

```
ans =
```

```
0 0 0
0 0 0
7 0 0
```

```
>> triu(B)
```

```
ans =
```

```
1 2 3
0 5 6
0 0 9
```

```
>> triu(B,-1)
```

```
ans =
```

```
1 2 3
4 5 6
0 8 9
```

```
>> triu(B,1)
```

```
ans =
```

```
0 2 3
0 0 6
1 0 0
```

2.3.5 Opérations entre matrices

- Les **additions** ou **soustraction** entre matrices ne sont possibles que si les dimensions des matrices sont **les mêmes**
- Le **produit matriciel** n'est possible que lorsque les dimensions sont cohérentes :

Exemple

```
>> M1=[1 2 3;4 5 6;7 8 9]
```

M1 =

```
1 2 3
4 5 6
7 8 9
```

```
>>M2=[1 :1 : 3 ; 11 :1 :13]
```

M2 =

```
1  2  3
11 12 13
```

```
>> M1 * M2'
```

```
ans =
```

```
14  74
32  182
50  290
```

-**Multiplication** ou **division** élément par élément :

```
>> M2. * M3 → M2 et M3 ont les mêmes dimensions
```

```
>> M2. / M3 → chaque élément de M2 est divisé par l'élément équivalent de
```

M3

```
>> M2. \ M3 → chaque élément de M3 est divisé par l'élément équivalent de
```

M2

```
>> M3. / M2 → chaque élément de M3 est divisé par l'élément équivalent de
```

M2

2.3.6 Caractéristiques des matrices

```
>> size(M2)
```

```
ans =
```

```
2    3
```

2 lignes et 3 colonnes

```
>> length (M2)
```

équivalent à max (size (M2)) : dimension

maximale

```
ans = 3
```

Fiche tp2

Exercice 1 :

1. Créer un vecteur ligne de coordonnées contenant les nombres de -5 à 5 avec un pas=1 et déterminer sa taille.
2. Créer un vecteur colonne contenant les nombres -500, -499, ..., 499, 500 et déterminer sa taille
3. Que fait l'instruction suivante : `v = [0 :0.2 :1]`
4. Que fait

l'instruction

suivante : `X = [-`

`1.3, sqrt(3),`

`(1+2+3)*4/5]`

Quelle est la valeur de `x(2)`

Exercice 2 :

1. Créer le vecteur suivant : `v = (8 -1 13 -4 7 6)`
2. Afficher la valeur de la 2eme valeur de `v`
3. Afficher les valeurs de `v` entre la 2eme et la 4eme position
4. Afficher les valeurs de `v` en commençant depuis la 5 eme valeur jusqu'à la 1ere avec une décrémentation de (-2)
5. Expliquer l'affichage des

commandes suivantes : `V(3 :end)`

, `v(1)=-1` , `v(7)=-1` ,

`v(2)=[]`

Exercice 3:

Compléter les opérations suivantes en indiquant ce qu'elles réalisent:

```
>>x= [1 ; 2 ; 3]
>>y= [4 ; 5 ; 6]
>>x+y
>>x-y
>>z1=x.*y
>>z2=x./y
>>z3=x.\y
>>z4=x.^y
>>length(z1)
>>u=linspace(1,20,5)
```

Chapitre 3 La programmation sous Matlab

Sommaire

- 3.1 Introduction**
- 3.2 Fichier Script**
 - 3.2.1 Définition
 - 3.2.2 Utilisation de la fenêtre script ou « M.File »
- 3.3 Les Entrée sortie**
 - 3.3.1 Lecture des données (les entrées)
 - 3.3.2 Ecriture des données (les sorties)
- 3.4 les expressions logiques**
- 3.5 Instructions de contrôle**
- 3.6 Les boucles**
 - 3.6.1 La boucle for
 - 3.6.2 La boucle while
 - 3.6.3 La boucle do while

3.1 Introduction

Nous avons utilisé la fenêtre « Command Window » pour créer et manipuler les variables, ainsi que l'utilisation des différentes fonctions prédéfinies.

Cette partie de travail, ne permet pas d'exécuter des programmes avec beaucoup de lignes de commandes pour résoudre des problèmes plus complexes, qui demandent des commandes plus structurées ou plus nombreuses.

Ce présent chapitre, traite la partie programmation Matlab, qui explique comment utiliser Matlab comme un véritable langage de programmation en utilisant une nouvelle fenêtre appelée « Script » et afficher l'exécution dans la fenêtre « Command Window ».

3.2 Fichier Script

3.2.1 *Définition*

Il est possible d'enregistrer une séquence d'instructions dans un seul fichier appelé

« Script ». Un script ou « M.File » est un fichier texte qui regroupe plusieurs commandes Matlab, identiques à celles que l'on peut employer directement dans la fenêtre de commandes de MATLAB, enregistré sous Matlab avec l'extension « .m » et qui joue le rôle de programme principal.

3.2.2 *Utilisation de la fenêtre script ou « M.File »*

Ouverture de la fenêtre : on crée des fichiers scripts en utilisant une fenêtre qui s'ouvre de plusieurs façons.

- ✓ à partir de la barre d'outils en cliquant sur l'icône  ,
- ✓ en tapant la commande `>> edit`,
- ✓ à partir du menu fichier en cliquant sur File→New→ M-file (figure cidessous).

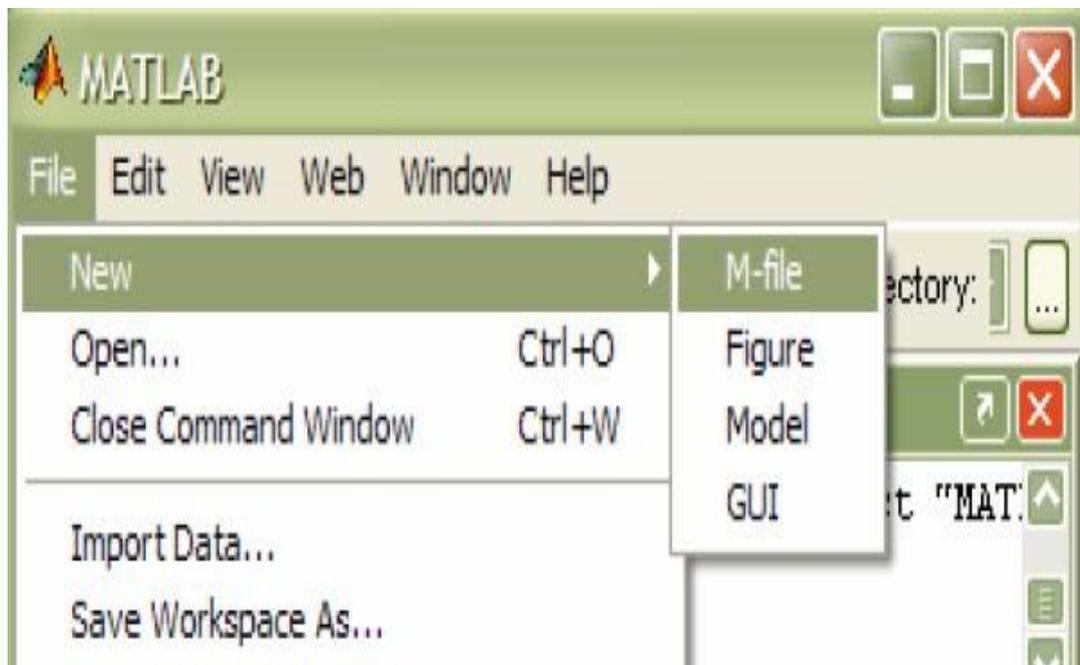


Figure 3.1 Ouverture d'un nouveau fichier « M.file ».

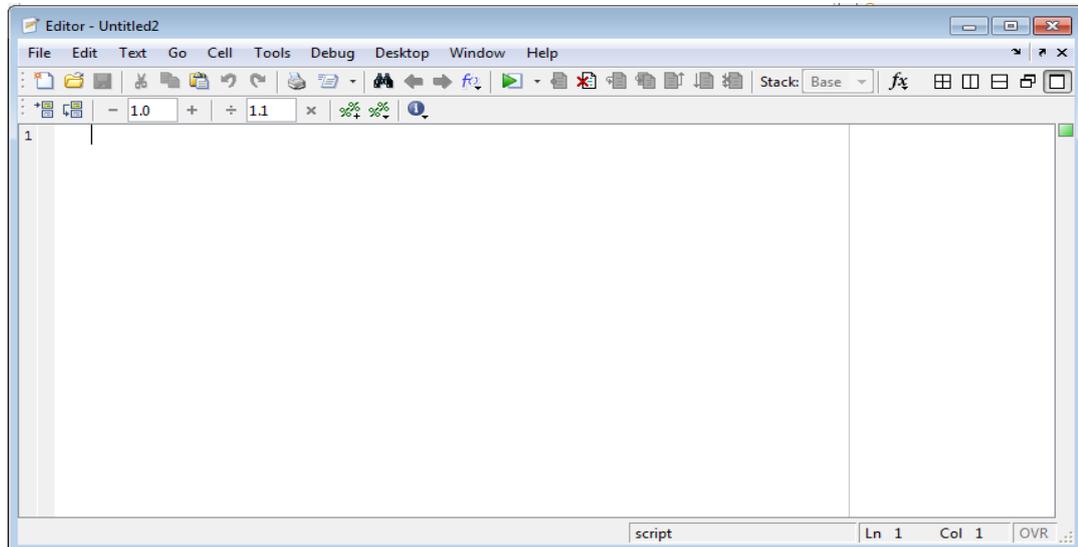


Figure 3.2 Fenêtre d'un fichier « script » ou « M.file ».

- Exécution d'un fichier script : Après l'enregistrement du fichier script, on peut l'exécuter de la façon suivante
 - ✓ à partir de la barre d'outils du fichier script, en cliquant sur l'icône ,
 - ✓ Si le script est écrit dans le fichier de nom **exemple.m** on l'exécute dans la fenêtre MATLAB en tapant son nom dans la fenêtre de commande (**>> exemple.m** et valider).

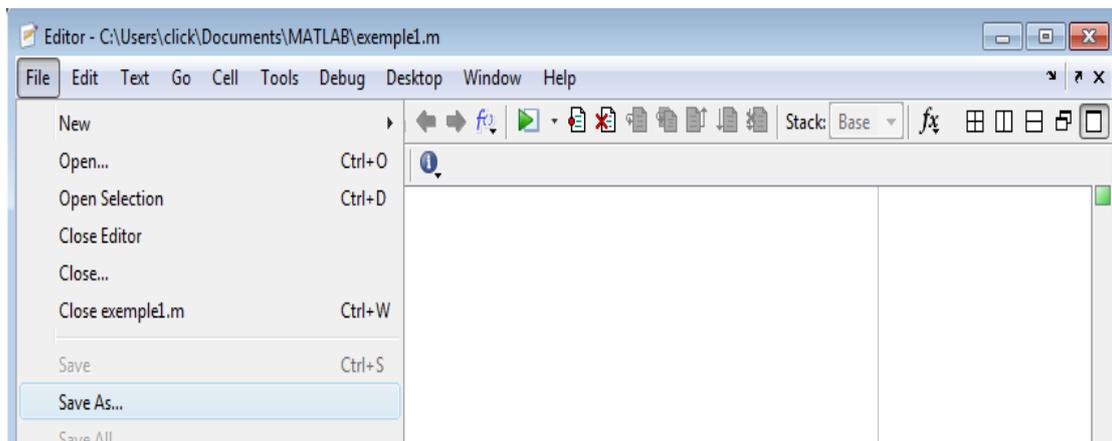


Figure 3.3 Enregistrement d'un fichier script.

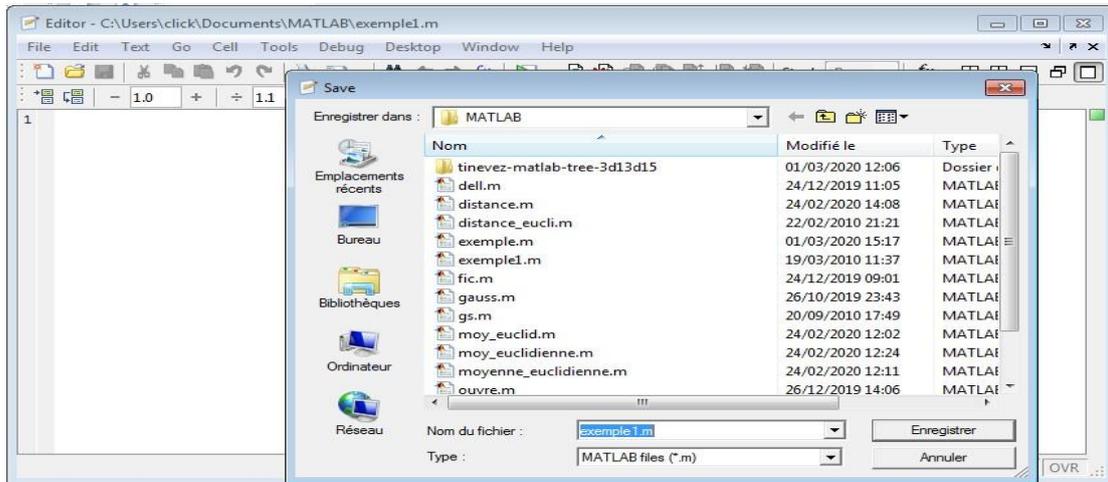


Figure 3.4 Une fenêtre d'enregistrement.

- Exécution d'un fichier script : Après l'enregistrement du fichier script, on peut l'exécuter de la façon suivante
 - ✓ à partir de la barre d'outils du fichier script, en cliquant sur l'icône ,
 - ✓ Si le script est écrit dans le fichier de nom **exemple.m** on l'exécute dans la fenêtre MATLAB en tapant son nom dans la fenêtre de commande (**>> exemple.m** et valider).

Exemple 3.1

1. Ouvrez l'éditeur de texte (ou de script) de Matlab. Reproduire alors le script présenté dans la fenêtre 3.5 ci-dessous.
2. Enregistrer le dans votre répertoire sous le nom « exemple.m ».
3. Exécuter le et afficher les résultats obtenus, c.-à-d. les valeurs des SC, SL, SD et SAD.

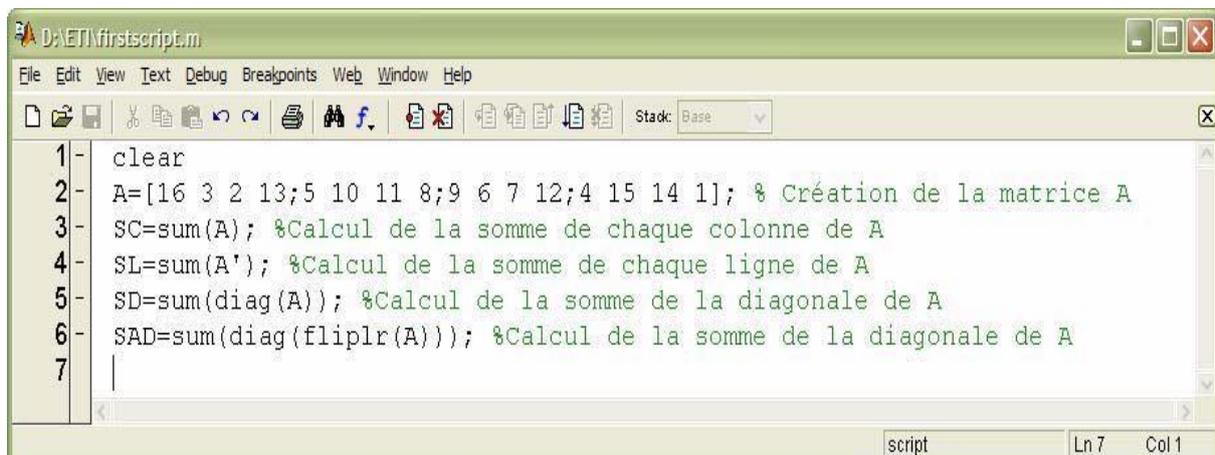


Figure 3.5 Exemple 3.1.

Remarque :

- Il est important de commencer un programme par l'instruction **clear**. Cette instruction effacera toutes les variables se trouvant dans l'espace. Ainsi, toutes les variables seront créées par le présent programme.
- Il est important de commenter abondamment un programme. Ceci permet de comprendre le programme lorsqu'on a besoin de le réutiliser après une longue période. Dans Matlab, une ligne commentaire commence par « % ».

3.3 Les Entrée et Sorties

3.3.1 Lecture des données (les entrées)

La commande « input » permet de demander à l'utilisateur d'un programme de fournir des données.

Syntaxe :

Variable=input('une phrase indicative');

Variable : une valeur déposée par l'utilisateur sera mise dans cette variable.

Input : une commande matlab permet de lire une valeur donnée par l'utilisateur.

Exemple 3.2:

```
>> A=input('entrer une valeur');
entrer une valeur 5
```

```
>> B=input('entrer un vecteur');
entrer un vecteur [1 2 3]
>> B
```

```
B =
```

```
    1    2    3
```

3.3.2 Ecriture des données (les sorties)

La commande « disp » permet d'afficher un tableau de valeurs numériques ou de caractères. L'autre façon d'afficher un tableau est de taper son nom. La commande « disp » se contente d'afficher le tableau sans écrire le nom de la variable ce qui peut améliorer certaines présentations.

Syntaxe :

Disp(objet)

- disp : commande matlab permet d'afficher à l'écran un objet.
- Objet : peut être un nombre, un vecteur, une matrice, une chaîne de caractère ou une expression.

Exemple 3.3:

```
>> A %affichage de la valeur de A
```

```
A =
```

```
5
```

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

```
5
```

```
>> disp(B)
```

```
1 2 3
```

3.4 les expressions logiques

3.4.1 *les opérations de comparaison*

L'opération de comparaison	signification
==	L'égalité
~=	L'inégalité
<, >, <=, >=	Inférieure, supérieure, inférieur égale, supérieur égale.

3.4.2 *les opérations logiques*

L'opération de comparaison	signification
&	Le et logique
	Le ou logique
~	La négation logique

3.4.3 Fonctionnement des opérations logiques

- Toute valeur égale à 0 sera considérée comme fausse.
- Toute valeur différente de 0 sera considérée comme vraie.

a	B	a&b	a b	~a
1 (vraie)	1 (vraie)	1	1	0
1 (vraie)	0 (faux)	0	1	0
0 (faux)	1 (vraie)	0	1	1
0 (faux)	0 (faux)	0	0	1

Exemple 3.4 :

```
>> x=10;
>> y=20;
>> x>y % affiche la valeur 0 (faux)
```

```
ans =
```

```
0
```

```
>> x==y
```

```
ans =
```

```
0
```

```
>> (x>9) & (y>10) % vraie et vraie le résultat 1 (vraie)
```

```
ans =
```

```
1
```

3.4.4 Comparaison des matrices et des vecteurs

La fonction	Description
isequal	Teste si deux (ou plusieurs) matrices sont égales (ayants les mêmes éléments partout). Renvoie 1 si c'est vraie et 0 sinon.
isempty	Teste si une matrice est vide ou non. Renvoie 1 si la matrice est vide et 0 sinon.

Exemple 3.5:

```
>> A=[1 2 3];
>> B=[1 2 3];
>> isequal(A,B)
```

ans =

1

```
>> C=[1 3 3];
>> A==C %comparer les éléments des deux vecteurs
```

ans =

1 0 1

```
>> D=[];
>> isempty(D)
```

ans =

1

3.5 Les Instructions de contrôle

3.5.1 Les instructions « if », « else » et « elseif »

Syntaxes:

Instruction “if”	Instruction “else”	Instruction “elseif”
if (condition) Instructions..... end	if (conditi on) Instructi ons..... else Instructions end	if (condition 1) Instructions..... elseif (conditio n 2) Instructi ons..... elseif (condition n) Instructions else Instructions end

Exemple 3.6:

Ecrire un programme sous matlab qui vérifie si un nombre donnée par l'utilisateur est divisible par la valeur 7 ou non. (Utiliser la fonction « mod »)

Solution :

```

Editor - C:\Users\click\Documents\MATLAB\exemple1.m*
File Edit Text Go Cell Tools Debug Desktop Window Help
[Icons]
+ 1.0 + ÷ 1.1 × % % ⓘ
1 - a=input('entrer une vleur:');
2 - if mod(a,7)==0
3 -     disp('a est divisible par 7');
4 - else
5 -     disp('a n'est pas divisible par 7');
6 - end
7 - |
    
```

Figure 3.6 Solution de l'exemple 3.6

Exécution :

```

    entrer une vleur:6
Dr. M a n'est pas divisible par 7
    
```

3.6 les boucles

3.6.1 *la boucle for*

Une boucle est une structure qui permet d'exécuter un certain nombre de fois un même bloc d'instructions.

- On peut créer une boucle en utilisant **for ... end**.
- On peut aussi réaliser des boucles **for** imbriquées.

Syntaxe :

Création d'une seule boucle for	Création de boucles for imbriquées
<u>for</u> (variable= expression_vecteur)groupes d'instructions <u>end</u>	<u>for</u> (variable1= expression_vecteur) <u>for</u> (variable2= expression_vecteur) groupes d'instructions <u>end</u> <u>end</u>

Remarque :

- Expression_vecteur : correspond à la définition d'un vecteur utilisé de la façon suivante (début : pas : fin).
- Variable : on l'appelle aussi indice. Il parcourt tous les éléments du vecteur défini par « expression_vecteur », où pour chaque indice on exécute un groupe d'instructions.

Exemple 3.7:

Ecrire un programme matlab qui permet de calculer l'expression suivante « $j=i*2$ », sachant que la variable i représente l'indice allant de la valeur 1 jusqu'à la valeur 4, ensuite afficher la valeur de la variable j .

Solution :


```

1 - a=input('entrer une mtrice');
2 - somme=0;
3 - for i=1:size(a,1)
4 -     for j=1:size(a,2)
5 -         somme=somme+a(i,j);
6 -     end
7 - end
8
9 - disp(somme);

```

Figure 3.9 Solution de l'exemple 3.8

Exécution du programme :

```

>> exemple1.m
entrer une mtrice [1 2 3;4 5 6;7 8 9]
45

```

Figure 3.10 Résultat d'exécution de l'exemple 3.8

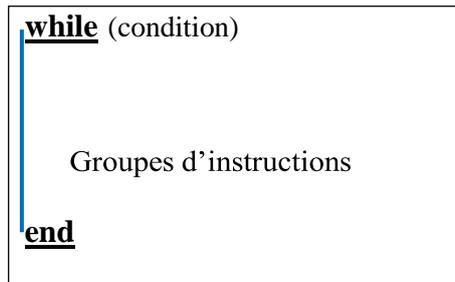
Table d'exécution (l'exemple 3.8) pas à pas :

A	somme	i	j	Disp(somme)
A=[1 2 3 ; 4 5 6; 7 8 9]	0	1	1	
	1+0=1	1	2	
	1+2=3	1	3	
	3+3=6	2	1	
	6+4=10	2	2	
	10+5=15	2	3	
	15+6=21	3	1	
	21+7=28	3	2	
	28+8=36	3	3	
	36+9=45			45

3.6.2 La boucle while

On peut créer une boucle en utilisant **while ... end**.

Syntaxe :



Exemple 3.9:

Ecrire un programme qui calcule le n factoriel

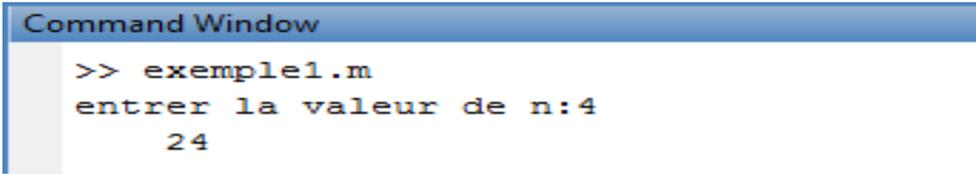
Solution :

```

Editor - C:\Users\click\Documents\MATLAB\exemple1.m
File Edit Text Go Cell Tools Debug Desktop Window Help
[Icons]
- 1.0 + ÷ 1.1 × %>% %>% ⓘ
1 - n=input('entrer la valeur de n:');
2 - m=1;
3 - i=1;
4 - while (i<=n)
5 -     m=m*i;
6 -     i=i+1;
7 - end
8
9 - disp(m);
  
```

Figure 3.11 Solution de l'exemple 3.9

Résultat d'exécution :



```
Command Window
>> exemple1.m
entrer la valeur de n:4
    24
```

Figure 3.12 Résultat d'exécution de l'exemple 3.9

Fiche TP 3

Exercice 1 :

1) Créez la matrice A à l'aide de matlab et calculez sa dimension

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

2) Expliquez l'affichage des deux syntaxes suivantes :

>>d1=size (A, 1)

>>d2 = size (A, 2)

3) Créez des syntaxes qui permettent d'afficher:

- L'élément de la 2eme ligne avec la 3eme colonne
- Tous les éléments de la 1ere ligne
- Tous les éléments de la 2eme colonne
- Tous les éléments de la 2eme et la 3eme ligne
- La sous matrice supérieure droite de taille 2x2
- La sous matrice : ligne (1,3) et colonne (2,3)
- Supprimer la 3eme colonne
- Supprimer la 2eme ligne
- Ajouter une nouvelle colonne qui contient des zéros
- Ajouter une nouvelle ligne qui contient que des uns

Exercice 2 :

1) Créez sous matlab la matrice B et le vecteur d :

$$B = \begin{bmatrix} 1 & -4 \\ 1 & 7 \\ 1 & 4 \end{bmatrix} \quad d = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

2) Ecrire à l'aide de matlab la matrice ordonné C composée de la matrice B et le vecteur d

3) Ecrire à l'aide de matlab la matrice D définie par :

D = Id - C.*Ct ou Id désigne la matrice identité et Ct la matrice transposé de C

Dr. Mohammed Chennoufi

Université Omar El-Mohamed Ben Ahmed
Institut de Maintenance et de Sécurité Industrielle

Exercice 3 :

- 1) Créez avec la ligne de commande la plus courte possible la matrice B suivante :

- 2) Est-il possible de calculer le déterminant de la matrice B ? si oui calculer le sinon trouver une solution pour pouvoir le calculer

Exercice 4 :

Soit la matrice A :
$$A = \begin{pmatrix} 1 & -4 & -1 & 1 \\ 1 & 7 & 1 & -2 \\ 1 & 4 & -1 & 2 \\ 3 & -10 & -2 & 5 \end{pmatrix}$$

- 1) créer la matrice A sous matlab
- 2) extraire les blocs suivants de la matrice A :
$$b1 = \begin{pmatrix} 4 & -1 \\ -10 & -2 \end{pmatrix}$$

$$b2 = \begin{pmatrix} 1 & -4 & -1 \\ 1 & 7 & 1 \\ 1 & 4 & -1 \end{pmatrix}$$

- 3) donner les valeurs de : A(2:4,3), A(3,end), diag(A,1), diag(tril(A)), diag(diag(A))

donner la ligne de commande permettant

$$C = \begin{pmatrix} 1 & -4 & -1 \\ 1 & 7 & 1 \\ 1 & 4 & -1 \end{pmatrix} + 3 * \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - 2 * \begin{pmatrix} 1 & -4 & -1 \\ 0 & 7 & 1 \\ 0 & 0 & -1 \end{pmatrix}$$

Chapitre 4: Les Graphiques sous Matlab

Introduction

4.1 Graphique 2 dimensions (2D)

4.1.1 la fonction PLOT

4.1.2 Modification de l'apparence d'une courbe

4.1.3 Annotation d'une figure

4.2 Traçage de plusieurs courbes sous Matlab

4.2.1 Sur la même figure avec la même échelle

4.2.2 afficher plusieurs graphiques dans la même fenêtre

Sommaire

Introduction

Matlab offre un puissant système de visualisation pour la présentation et l'affichage graphique des données d'une manière à la fois efficace et facile.

Dans cette partie, nous allons présenter les principes de base indispensables pour dessiner des courbes en Matlab.

4.1 Graphique 2 dimensions (2D)

4.1.1 La fonction plot

La fonction la plus simple pour la création des graphiques du type 2D est la fonction «plot», elle peut prendre en arguments deux vecteurs ou bien deux matrices de même taille, permettant de tracer une courbe affichée sur un graphique à deux axes (axe des abscisses et axe des ordonnées), en reliant les points de coordonnées définis dans ses arguments ayant plusieurs formes.

Par exemple, plot(x, y) marquera un point pour chaque couple [x(i), y(i)] avec i allant de 1 à length(x). On représente ainsi les valeurs de y en fonction de x, sachant que x et y ayant la même taille.

a) Formes des arguments

1. Deux vecteurs de la même taille comme

arguments :

2. Exemple 1 :

```
>> x= [1 2 3]

x =

     1     2     3

>> y= [-2 1 3]

y =

    -2     1     3

>> plot(x,y)
```

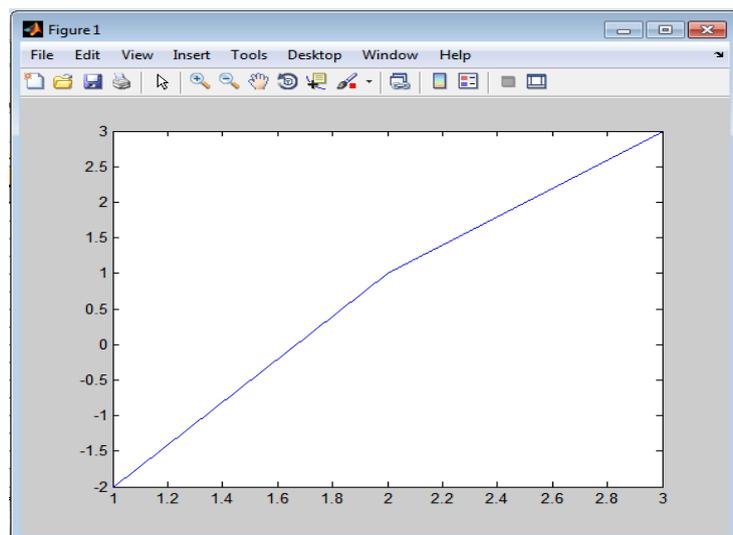


Figure 4.1 Fenêtre graphique (Exemple 1).

Remarque :

La variable x représente les valeurs du premier vecteur (axe des abscisses) et la variable y représente les valeurs du deuxième vecteur (axe des ordonnées).

Un seul vecteur comme argument :**Exemple :**

```
>> plot ([1 3 5 2]);
```

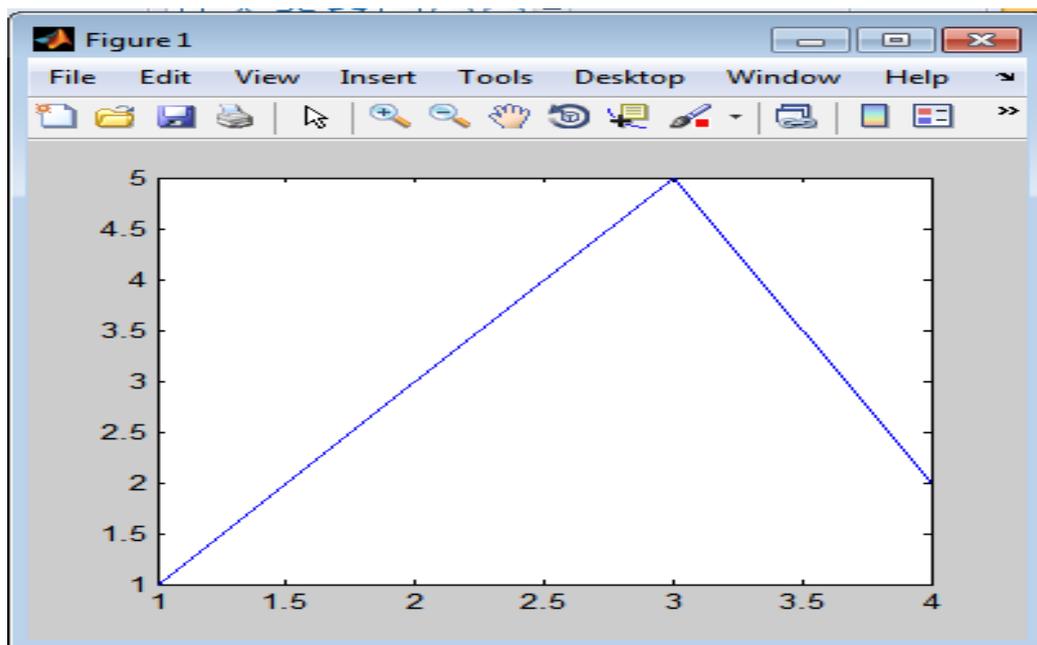


Figure 4.2 Fenêtre graphique (Exemple 2).

Remarque :

Les valeurs du vecteur [1 3 5 2] représentent les éléments de l'axe y (axe des ordonnées), et leurs positions représentent les éléments de l'axe x (axe des abscisses).

3. Une seule matrice comme argument**Exemple 3:**

```
>> M=[1 0 2;2 3 1;0 1 2]
```

M =

```

1     0     2
2     3     1
0     1     2
```

```
>> plot(M)
```

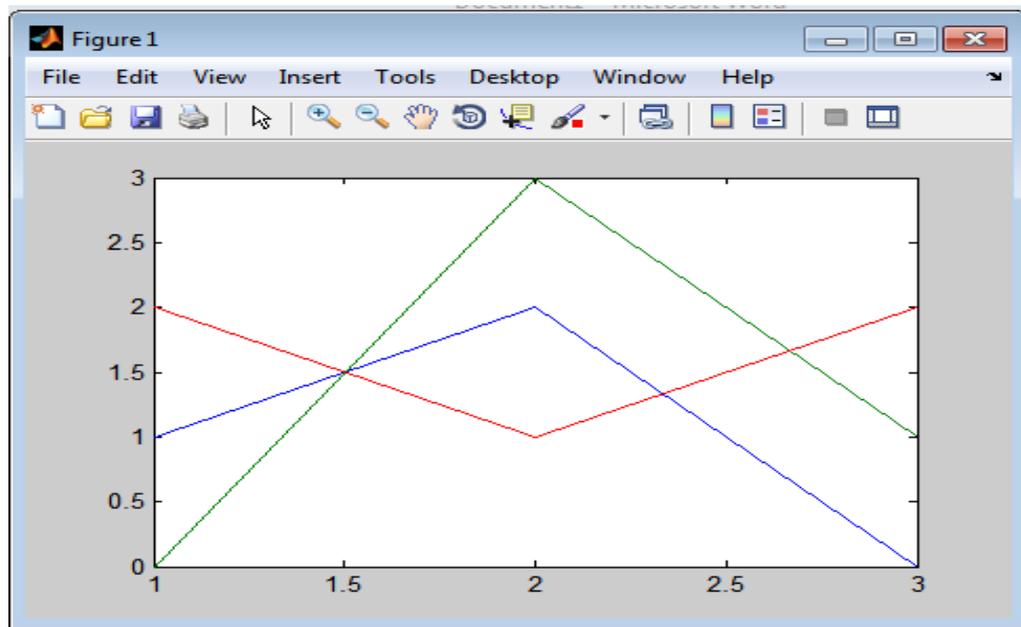


Figure 4.3 Fenêtre graphique (Exemple 3)

Remarque :

- Chaque colonne de la matrice M sera représentée par une courbe.
- La fonction `plot(M)` donnera plusieurs courbes avec des couleurs différentes, sachant que la première colonne est représentée par la courbe bleue, la deuxième colonne est représentée par la courbe rouge et la troisième colonne est représentée par la courbe verte.
- Les valeurs de chaque colonne de la matrice M représentent les éléments de l'axe y et leurs positions (le numéro de ligne) comme les valeurs de l'axe x.

4. Deux matrices comme arguments :

Exemple 4 :

```
>> K=[1 1 1;2 2 2;3 3 3]
```

```
K =
```

```
    1    1    1
    2    2    2
    3    3    3
```

```
>> L=[0 -2 1;2 0 3;-3 3 -2]
```

```
L =
```

```
    0   -2    1
    2    0    3
   -3    3   -2
```

```
>> plot(K,L)
```

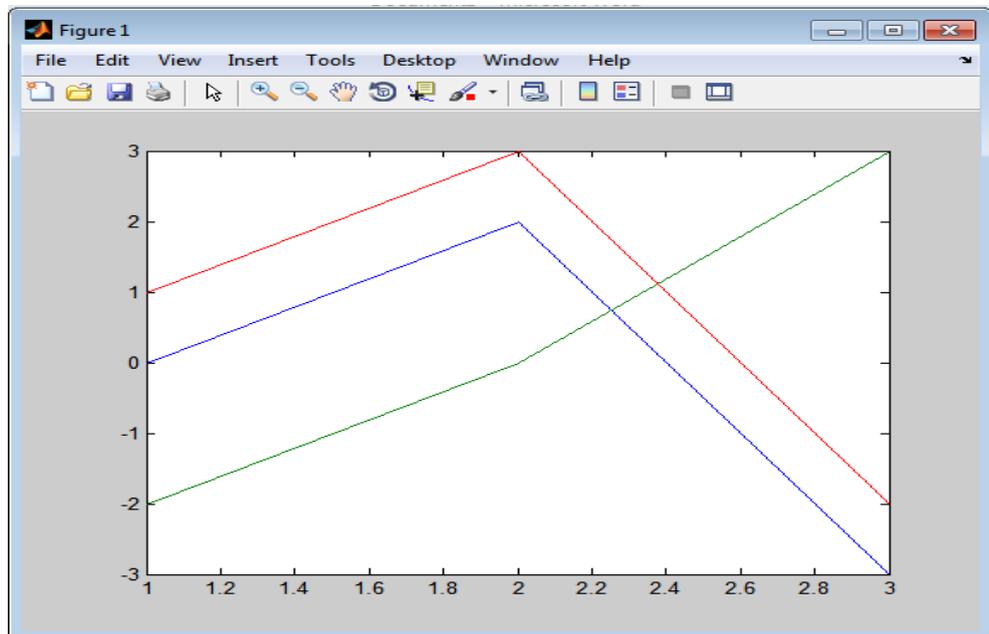


Figure 4.4 Fenêtre graphique (Exemple 4).

Remarque :

- Les valeurs de chaque colonne de la première matrice représentent les éléments de l'axe x.
- Les valeurs de chaque colonne de la deuxième matrice représentent les éléments de l'axe y.

4.1.2 Modification de l'apparence d'une courbe

Le traçage des courbes sous Matlab s'affiche par défaut en bleu, mais il est possible de changer l'apparence d'une courbe en modifiant sa couleur, la forme des points de coordonnées et le type de ligne reliant les points.

Pour cela, on ajoute un nouveau argument de type chaîne de caractère appelé « marquer » à la

fonction plot en utilisant la syntaxe suivante :

plot (x, y, 'marquer')

Le mot marquer peut contenir un ou plusieurs caractères parmi les caractères rassemblés dans les tableaux suivants :

Couleur de la courbe	
Le caractère	Son effet
b ou blue	Courbe en bleu
g ou green	Courbe en vert
r ou red	Courbe en rouge
c ou cyan	Entre le vert et le bleu
m ou magenta	En rouge violacé vif
y ou yellow	Courbe en jaune
k ou black	Courbe en noir

Style de la courbe	
Le caractère	Son effet
-	En ligne plein
:	En pointillé
-.	En point tiret -.-.-.-.
--	En tiret - - - - -

Représentation des points	
Le caractère	Son effet
.	Un point
o	Un cercle
x	Le symbole ×
+	Le symbole +
*	Le symbole *
S	Un carré
D	Un losange
v	Triangle inférieur
^	Triangle supérieur
<	Triangle gauche
>	Triangle droit

Exemple :

- Pour $x = [0 \dots 6]$ tracer le graphe de la fonction $y = \cos(x \cdot \pi)$ avec un pas = 0,1.
- Changer l'apparence de la courbe en utilisant les deux marqueurs suivants : ('rd-', 'c:^')

Solution :

```
>> x=[0:0.1:6];  
>> y=cos(x*pi);  
>> plot(x,y,'rd-')
```

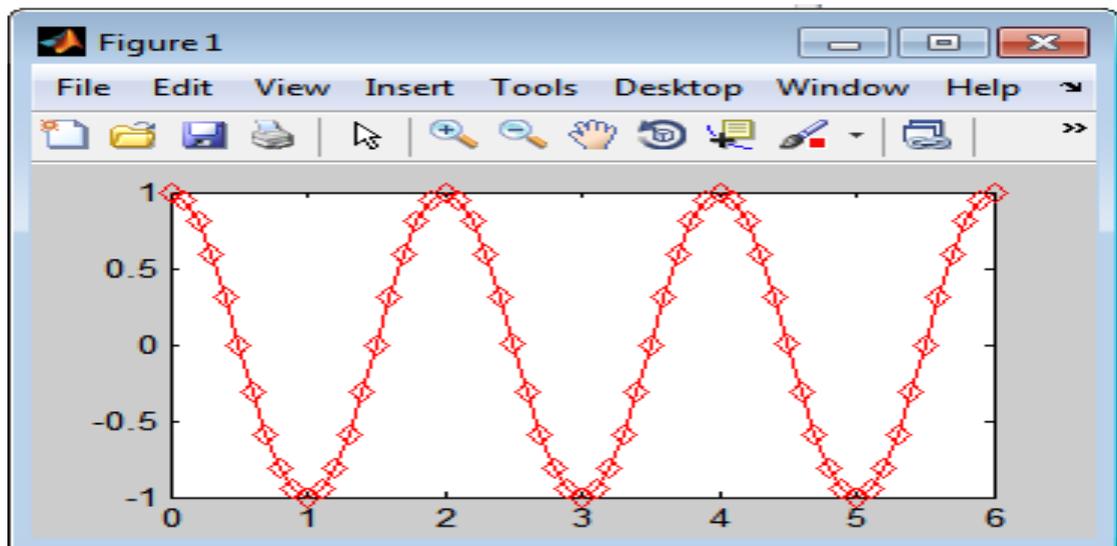


Figure 4.5 Courbe avec couleur rouge, en ligne plein avec des losanges.

```
>> x=[0:0.1:6];  
>> y=cos(x*pi);  
>> plot(x,y,'c:^')
```

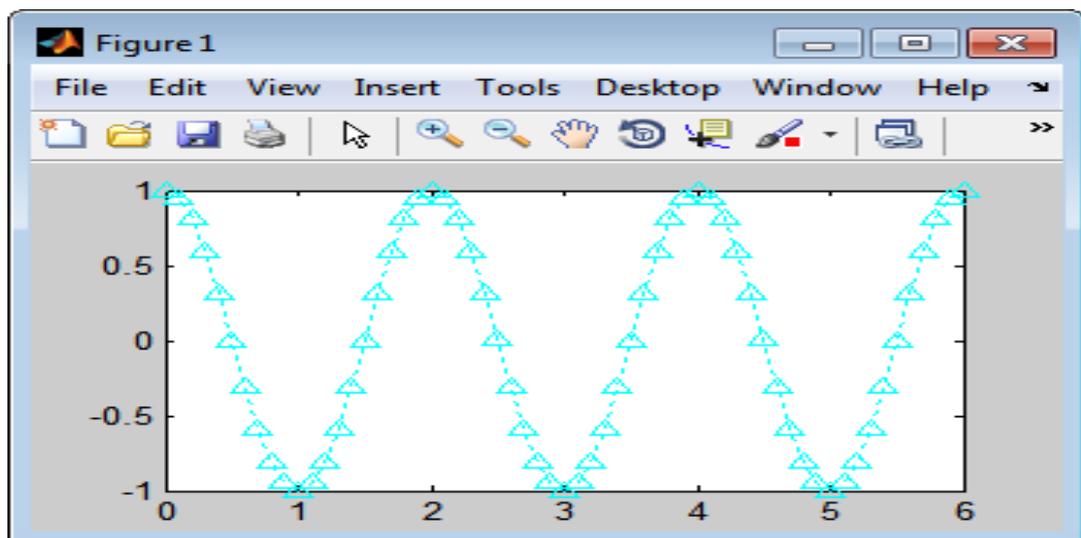


Figure 4.6 Courbe avec couleur cyan, en pointillé avec des triangles supérieur.

4.1.3 Annotation d'une figure

Pour une meilleure compréhension du graphe, il est préférable d'ajouter des informations dans le graphique.

1. Titre de la figure :

On utilise la fonction « **title** » pour donner un titre à la figure utilisé comme ceci :

```
>> title ('titre de la figure')
```

2. La commande « xlabel » et « ylabel » :

On utilise la commande « **xlabel** » pour donner un nom à l'axe des abscisses et « **ylabel** » pour donner un nom à l'axe des ordonnées de la figure utilisé comme ceci :

```
>> xlabel ('Ceci est l'axe des abscisses X')
```

```
>> ylabel ('Ceci est l'axe des ordonnées Y')
```

3. La commande « text » :

Permet d'introduire un texte dans la fenêtre du graphique à une position indiquée par les coordonnées X et Y utilisée comme ceci :

```
>> text (x, y, 'veuillez respecter le message suivant')
```

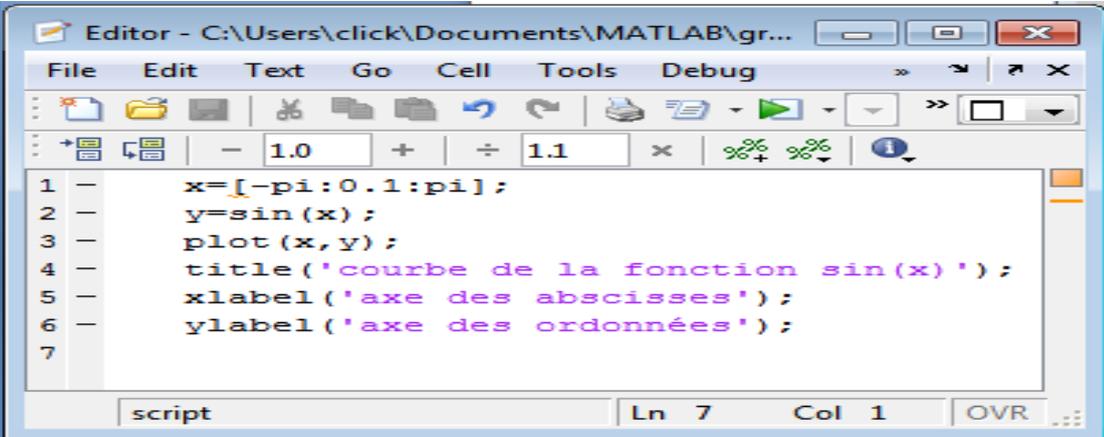
4. La commande « grid » :

Pour représenter une figure sous forme de grille on utilise la commande **grid** (ou **grid on**) et pour l'enlever on utilise la commande **grid** (ou **grid off**).

Exemple :

Dans un nouveau script, écrire un programme qui permet de tracer le graphe de la fonction $y=\sin(x)$ avec $x=[-\pi \dots \pi]$ et un pas=0,1. Donnez un titre à la figure, un nom à l'axe des abscisses et l'axe des ordonnées.

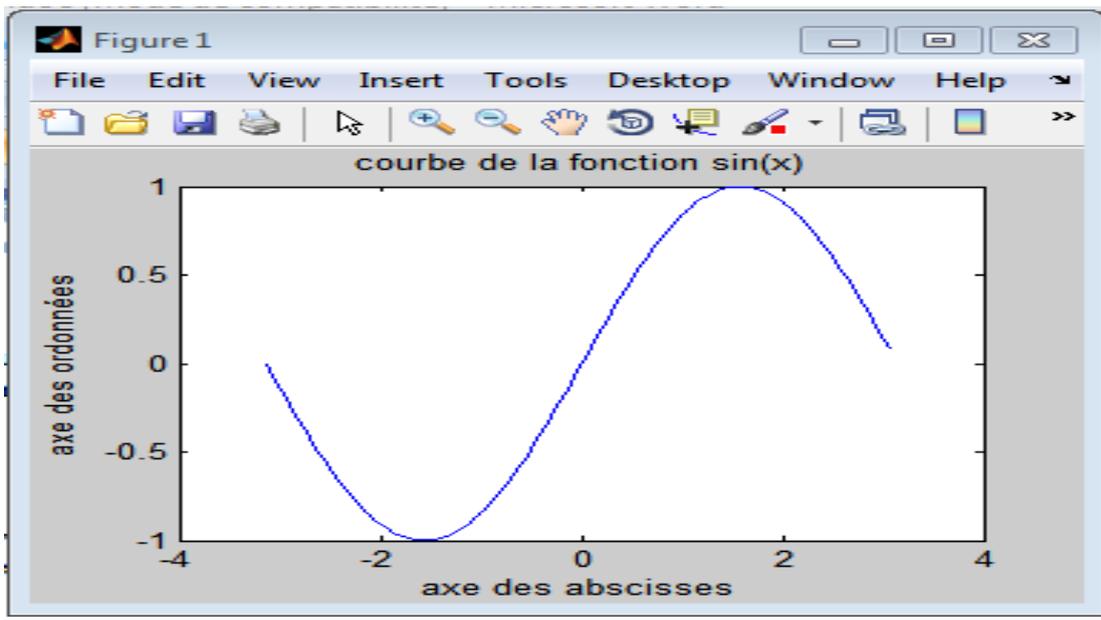
Solution :



```

Editor - C:\Users\click\Documents\MATLAB\gr...
File Edit Text Go Cell Tools Debug
1 - x=[-pi:0.1:pi];
2 - y=sin(x);
3 - plot(x,y);
4 - title('courbe de la fonction sin(x)');
5 - xlabel('axe des abscisses');
6 - ylabel('axe des ordonnées');
7
script Ln 7 Col 1 OVR ...

```



Légende des courbes d'un graphique

Dès que l'on trace plusieurs courbes sur le même graphique, il devient indispensable d'ajouter une légende, pour spécifier à quoi correspond chacune des courbes.

L'instruction **legend** permet d'ajouter cet élément. Il faut lui communiquer autant de chaînes de caractères que de courbes tracées. Un cadre est alors ajouté sur le graphique, et affiche en face du style de chaque courbe, le texte correspondant

4.2 Traçage de plusieurs courbes sous Matlab

4.2.1 Sur la même figure avec la même échelle

Pour tracer plusieurs courbes sur le même graphique, il suffit de spécifier autant de couples de vecteurs (abscisses, ordonnées) qu'il y a de courbes à tracer.

Syntaxe :

```
plot(Vx1,Vy1,Vx2,Vy2,Vx3,Vy3, ..., Vxn,Vyn)
```

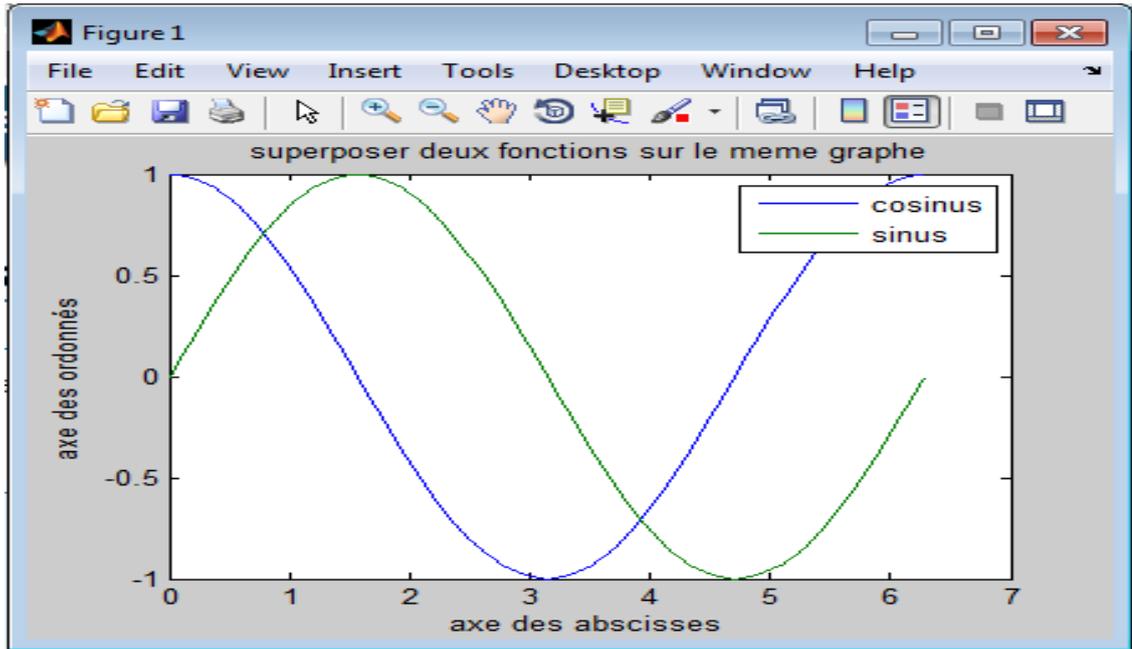
Où les V_{xi} sont les vecteurs d'abscisses, et V_{yi} les vecteurs d'ordonnées.

Exemple :

Superposer les courbes des deux fonctions $\sin(x)$ et $\cos(x)$ sur le même graphe pour $x=0:2*\pi/100:2*\pi$

Solution :

```
>> x=[0:2*pi/100:2*pi];  
>> plot(x,cos(x),x,sin(x));  
>> xlabel('axe des abscisses');  
>> ylabel('axe des ordonnées');  
>> title('superposer deux fonctions sur le meme graphe');  
>> legend('cosinus','sinus');
```



e
r plusieurs courbes sur la même échelle.

4.2.2 afficher plusieurs graphiques dans la même fenêtre

L'idée générale est de découper la fenêtre graphique en zones, et d'afficher un graphe dans chacune des zones. On utilise l'instruction subplot en lui spécifiant le nombre de zones sur la hauteur, le nombre de zones sur la largeur, et le numéro de la zone que l'on considère (et dans laquelle on va tracer une courbe).

Syntaxe :

Subplot (m , n , p)

Où m représente le nombre de colonnes, n le nombre de lignes, et P est le numéro de la zone à laquelle on s'intéresse (la position de la courbe).

Exemple :

```
>> x = 0:2*pi/100:2*pi;
    >> subplot(2,2,1);
>> plot(x,sin(x));
    >> subplot(2,2,2);
>> plot(x,cos(x),x,sin(x),'-.');
    >> subplot(2,2,3);
>> plot(cos(x),sin(x));
    >> subplot(2,2,4);
```

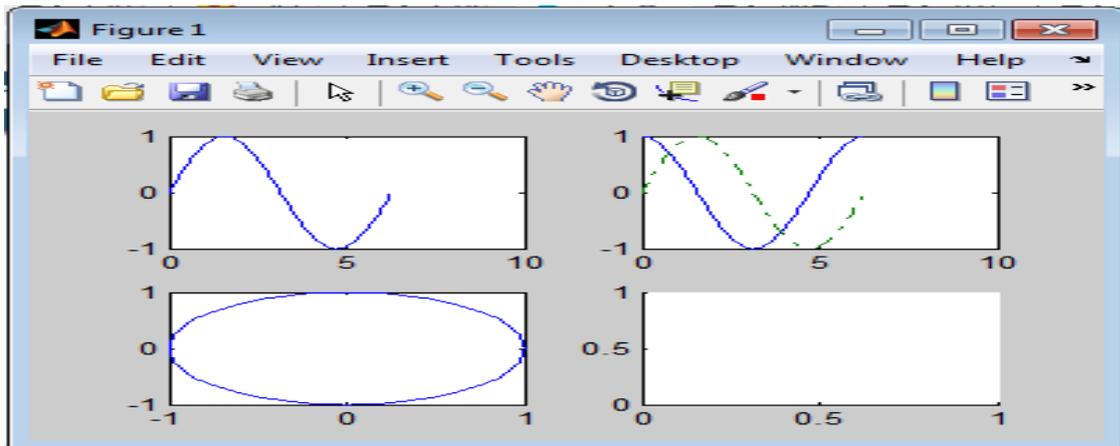


Figure 4.8 Plusieurs graphiques avec la commande subplot

TP 4

Exercice 1 :

Soit les trois fonctions suivantes :

1. $y_1 = \sin(x)$;
2. $y_2 = \sin(x + \pi/3)$;
3. $y_3 = \sin(x + 2\pi/3)$;

Tracer les courbes des trois fonctions y_1, y_2, y_3 sur la même fenêtre graphique en utilisant la commande subplot dans l'intervalle $x = [-\pi, \pi]$, avec un pas de 0.1.

Exercice 2 :

Soit les trois fonctions suivantes :

1. $z_1 = \exp(-2x) \cdot \sin(x)$;
2. $z_2 = \exp(-3x) \cdot \sin(x)$;
3. $z_3 = \exp(-4x) \cdot \sin(x)$;

En utilisant la commande plot, tracer les courbes des trois fonctions z_1, z_2, z_3 dans une même figure, dans l'intervalle $x = [1, 2\pi]$ avec un pas de 0.2 en utilisant la convention graphique:

1. Pour z_1 couleur rouge avec des tirets.
2. Pour z_2 couleur vert avec des tirets.
3. Ajouter une légende à chaque courbe.
4. Ajouter le titre suivant « Comparaison entre trois fonctions ».

Solutions fiche TP

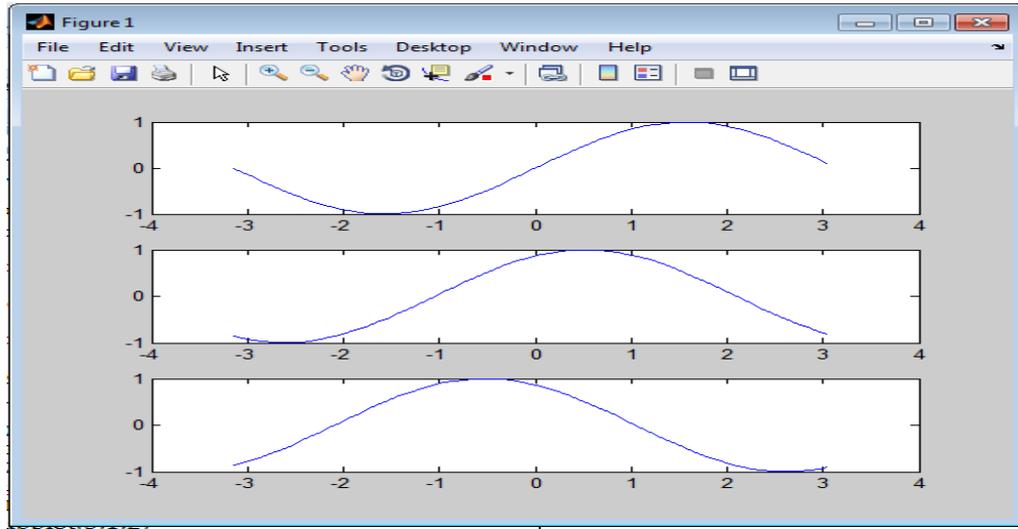
Exercice 1 :

Script :

```

x = -pi:.1:pi;
y0 = sin(x);
y1 = sin(x+pi/3);
y2 = sin(x+2*pi/3);
subplot(3,1,1) plot(x,y0);
subplot(3,1,2) plot(x,y1);
subplot(3,1,3) plot(x,y2);

```



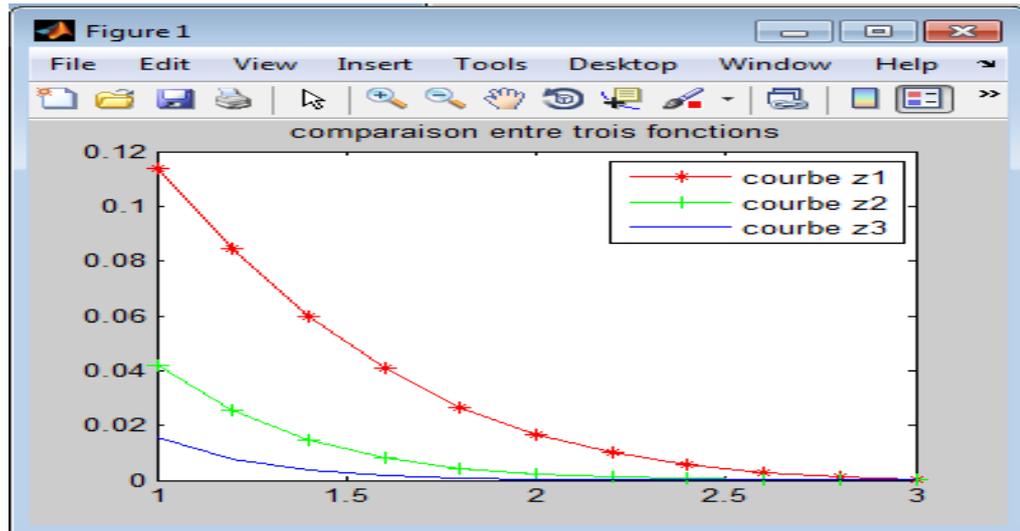
Exercice 2 :

Script :

```

x = 1:.2:pi;
z1 = exp(-2*x).*sin(x);
z2 = exp(-3*x).*sin(x);
z3 = exp(-
4*x).*sin(x);
plot(x,z1,'r-
*',x,z2,'g-
+',x,z3);
title('comparaison entre trois
fonctions') legend('courbe z1',
'courbe z2', 'courbe z3')

```



Annexe 1

Examen de synthèse
Informatique 3
2021

Exercice 1 : (8 points)

Soit la matrice A et les deux vecteurs B et C,

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad B = [10 \ 8 \ 1] \quad C = \begin{bmatrix} -9 \\ -3.5 \\ 4 \\ 5 \end{bmatrix}$$

1) Ecrivez des commandes Matlab permettant de :

- Extraire la 2ème ligne de la matrice A.
- Extraire la 1ère et la 3ème colonne de la matrice A.
- Supprimer la 3ème ligne de la matrice A.
- Ajouter le vecteur B à la matrice A.
- Créer un vecteur colonne qui contient 51 éléments entre -25 et 25.

2) A partir des vecteurs B et C créer le vecteur ligne D suivant, en utilisant la ligne de commande Matlab la plus courte :

$$D = [8, 1, -9, -3.5, 4, 10, 8]$$

3) Avec une instruction Matlab, définir la matrice E suivante :

$$E = \begin{bmatrix} -3 & -3 & -3 & 0 & 0 \\ -3 & -3 & -3 & 0 & 0 \\ -3 & -3 & -3 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Exercice 2 : (5 points)

4) Donnez le résultat d'exécution des commandes Matlab suivantes :

```
>> t = [1 : 0.5 : 3]';
>> F = [t, t+2, t-1]
>> G = F (2, [1, 3])
>> H = G * eye (2,2)
>> L = (diag ( t ), F (:, 2))
```

Exercice 3 : (7points)

En utilisant les boucles, écrire un Script Matlab qui calcule la somme suivante (x ,n donnée par l'utilisateur).

$$s = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} \dots \dots \dots \pm \frac{x^n}{n}$$

Bibliographie

Bibliographie du polycopié du cours

[1] : Abdellah MECHAQRANE, (2008) Introduction à Matlab et simulation. Université Sidi Mohammed Ben Abdallah, Faculté Des Sciences et Techniques Fès, Département Génie Electrique.

[2] : John Chaussard, (2017), Introduction à Matlab, Ecole Sup Galilée - Cours Ingénieur - 1ère année.

[3] : Yassine Ariba - Jérôme Cadieux, Manuel Matlab, Départements GEI & Mécanique, Icam de Toulouse.

[4] : Matlab help

[5] : Introduction à Matlab, J.-T. Lapresté (Ellipses 1999)

[6] : Mastering Matlab 6, D.Hanselman B. Littlefield (Prentice Hall 2001)

[7] : Apprendre et maîtriser Matlab, M.Mokhtari A.Mesbah (Springer 1997)