وزارة التعليم العالي والبحث العلمي
جامعة محمد بوضياف بالمسيلة
كلية الرياضيات والإعلام الآلي

قسم الرياضيات

اللجنة العلمية

رقم: 250 ق.ر/2021

المسيلة في: 08- 11 - 2021

مستخلص محضر اللجنة العلمية ليوم: 2021/11/08
بخصوص إعتماد مطبوعة دروس

وافقت اللجنة العلمية على إعتماد المطبوعة الخاصة بالأستاذ: **جعيجع النوي** والتي عنوانها:

Notes de cours: outils de programmation 2

كمرجع لدروس طلبة السنة الثانية ليسانس رياضيات
وهذا بعد الإطلاع على التقرير الإيجابي المنجز من طرف الأستاذة: **خيراني آمنة** (أستاذ محاضر – أ جامعة المسيلة).

رئيس اللجنة العلمية

جامعة المسيلة
رئيس اللجنة العلمية
لقسم الرياضيات
كلية الرياضيات و الإعلام الآلي

دريهم الذوادي

Résumé

Le but de ce cours est de permettre aux étudiants de deuxième année licence mathématiques de :

1- Découvrir les bases du langage Matlab;

2- Apprendre la syntaxe de base du langage Matlab ;

3- Se familiariser rapidement avec Matlab ;

4- L'apprentissage de la programmation et des fonctionnalités principales de MATLAB.

Mots -clés : Langage Matlab, Fonctionnalités de base, Outil graphique, Programmation.

Mot clés

REPUBLIQUEALGERIENNEDEMOCRATIQUEETPOPULAIRE
MINISTEREDEL'ENSEIGNEMENTSUPERIEURETDELARECHERCH
ESCIENTIFIQUE

Université Mohamed Boudiaf de M'sila

Faculté des Mathématiques et de l'Informatique
Département des Mathématiques

Notes de cours

Outils de programmation 2

Licence 2 Mathématiques-Semestre 3

Noui DJAIDJA

Année universitaire 2020/2021

Table des matières

INTRODUCTION

Ce polycopié de cours, est un guide à la découverte des différentes fonctionnalités de base du logiciel Matlab.
Matlab nous permet de résoudre numériquement de nombreux problèmes mathématiques. En outre, Matlab dispose de développement avec l'outil graphique.
Le but de ce cours est de permettre aux étudiants de deuxième année licence mathématiques de :

1- Découvrir les bases du langage Matlab;
2- Apprendre la syntaxe de base du langage Matlab ;
3- Se familiariser rapidement avec Matlab ;
4- L'apprentissage de la programmation et des fonctionnalités principales de MATLAB.

Les notions abordées dans ce polycopié de cours sont organisées en six chapitres.

Chapitre I : présentation et prise en main de Matlab.

Chapitre II : Les Variables.

Chapitre III : Les Vecteurs et les Matrices.

Chapitre IV : Programmation avec Matlab.

Chapitre V : Graphisme en Matlab.

Chapitre VI : Les polynômes

Chapitre I

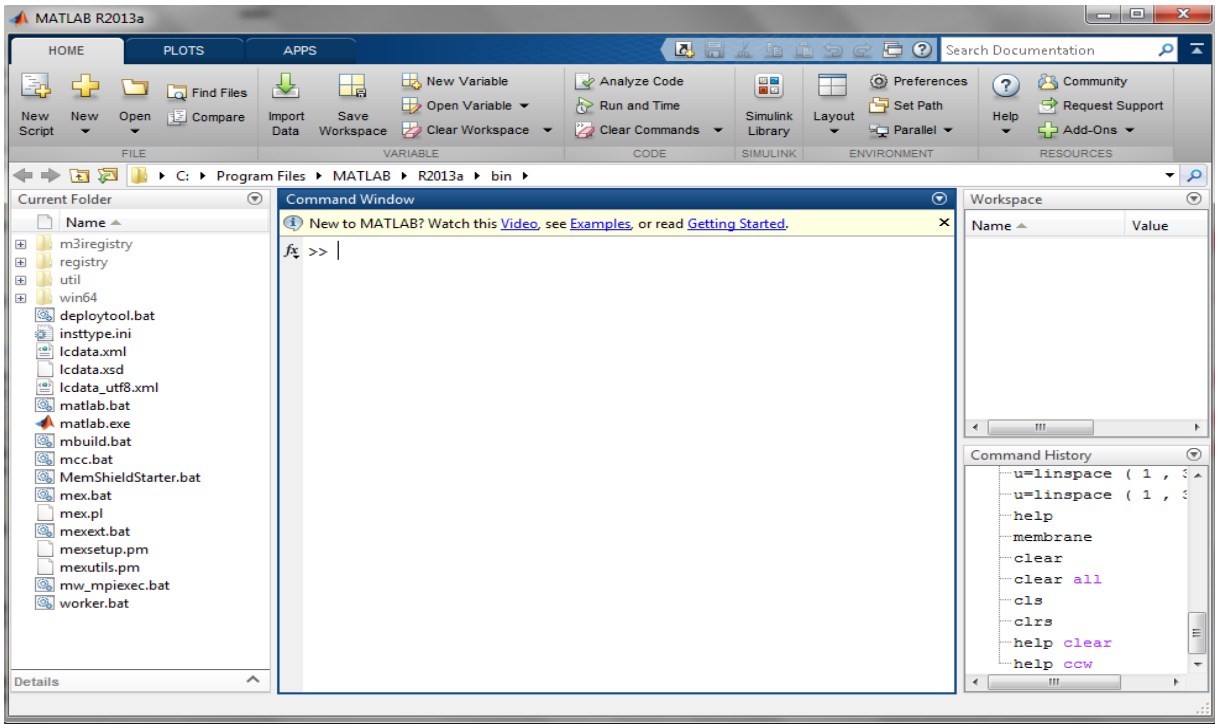
I Présentation et prise en main de Matlab

I.1 Introduction à Matlab

Matlab pour « MATtrix LABoratory », est un logiciel qui a été conçu pour fournir un environnement de calcul numérique. Il est particulièrement performant pour le calcul matriciel car sa structure de données interne est basée sur les matrices.

I.2 Interface de Matlab

Figure 1



L'interface Matlab se compose d'une fenêtre principale divisée en quatre sous-fenêtres.

1-A gauche, il y a la fenêtre **Current Folder** indique le répertoire courant, ainsi que les fichiers existants.

2- Au centre, il y a une grande fenêtre : **Command Window** formuler les expressions et interagir avec Matlab, c'est la fenêtre qui nous utilisons tout au long de ce module.
3- En haut à droite, il y a la fenêtre **Workspace** qui permet de gérer les variables utilisées.
4- En bas à droite la fenêtre **Command History** indique les dernières commandes effectuées.

I.3 Lancement de MATLA

Pour lancer Matlab il suffit de cliquer sur l'icône de l'application. La fenêtre de commande de Matlab s'ouvre alors et on tape les commandes ou les expressions à évaluer à droite du prompt ">>". Le processus d'évaluation est déclenché par la frappe de la touche "enter".
Le symbole >> indique à l'utilisateur où il faut rentrer la commande.
Pour quitter MATLAB, tapez quit suivi de la touche "enter" ou utilisez le menu "File" avec l'option "Exit MATLAB".

I.4 L'aide dans Matlab

Matlab comporte un très grand nombre d'opérateurs, de commandes et de fonctions. Tous ne seront pas décrits dans ce document d'autant qu'une aide en ligne efficace peut être utilisée.
On peut taper les commandes suivantes :
>> help : produit une liste de domaines MATLAB par catégories,
>> help <fonction ou commande> : fournit de l'aide sur l'utilisation de la fonction ou de la commande indiquée.
>> look for "mot-clé": fournit la liste des fonctions et commandes contenant le mot-clé spécifié dans la première ligne de leur texte d'aide.

On obtient la liste des fonctions MATLAB usuelles en classées par thème en tapant helpwin :
>> helpwin elfun : affiche la liste des fonctions mathématiques élémentaires
>> helpwin specfun : affiche la liste des fonctions mathématiques avancées
>> Helpwin elmat : affiche la liste des fonctions matricielles élémentaires

Exemple
>> help cos

cos Cosine of argument in radians.
cos(x) is the cosine of the elements of x.

Syntax
y = cos(x)
y = cos(x) returns the cosine for each element of x The cosfunction operates element-wise on arrays. The function accepts both real and complex inputs. For purely real values or imaginary values of x,

cos returns real values in the interval [-1 ,1]. For complex values of x, cos returns complex values. All angles are in radians.

Exemple >> helpwin elfun
Elfun
  Elementary math functions.

  Trigonometric

sin	Sine.
sinh	Hyperbolic sine.
asin	Inverse sine.
asinh	Inverse hyperbolic sine.
cos	Cosine.
cosh	Hyperbolic cosine.
acos	Inverse cosine.
acosh	Inverse hyperbolic cosine.
tan	Tangent.
tanh	Hyperbolic tangent.
atan	Inverse tangent.
atanh	Inverse hyperbolic tangent.
cot	Cotangent.
coth	Hyperbolic cotangent.
acot	Inverse cotangent.
acoth	Inverse hyperbolic cotangent.

deg2rad	Convert angles from degrees to radians.
rad2deg	Convert angles from radians to degrees

Exponential

exp	Exponential
log	Natural logarithm
log10	Common (base 10) logarithm
log2	Base 2 logarithm and dissect floating point number.
sqrt	Square root
realsqrt	Square root of number greater than or equal to zero.
nthroot	Real n-th root of real numbers.

Complex

abs	Absolute value.
angle	Phase angle.
complex	Construct complex data from real and imaginary parts.
conj	Complex conjugate.
imag	Complex imaginary part.
real	Complex real part.
isreal	True for real array.

Rounding and remainder

fix	Round towards zero.
floor	Round towards minus infinity.
ceil	Round towards plus infinity.
round	Round towards nearest integer.
mod	Modulus (signed remainder after division).
rem	Remainder after division.
sign	Signum.

Chapitre II

II Les variables

II.1 Notions de base de MATLAB

II.1.1 Types de données

Dans MATLAB, il y a un seul type de données : le type matrice (Matrix). Tout est matrice, un scalaire est une matrice carrée d'ordre 1. Il n'y a donc pas de déclaration de types. De même, l'utilisateur ne s'occupe pas de l'allocation mémoire. Les variables matrices et vecteurs peuvent être redimensionnés et même changer de type.

II.1.2 Le mode interactif

Dans le mode interactif, MATLAB peut être utilisé comme une "super-puissante" calculatrice scientifique. On dispose des opérations arithmétiques et d'un ensemble important de fonctions de calcul numérique et de visualisation graphique. L'affichage des résultats est automatique en fin de traitement.

Exemple

```
>> 3+7
ans =
    10
```

Le résultat est stocké par défaut dans la variable prédéfinie ans, par contre si on veut créer une variable on utilise la structure simple : variable = définition

Exemple

```
>> x=3+7
x=
   10
```

L'affichage des résultats peut être annulé, en ajoutant un point virgule " ; " à la fin de l'expression.

Exemple

```
>> 3+7 ;
>> x= 4+9 ;
>>
```

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

Exemple

```
>> 5+6,x= 2*5-3, y=3^2 ;
ans =
    11
x =
    7
>>
```

Remarque : l'ancienne valeur de x est systématiquement écrasée.

II.1.3 Variables spéciales et constantes

Dans MATLAB, on trouve des constantes prédéfinies :

pi : 3.1415926535 8979. , pi=4atan(1)= imag(log(-1)).
eps : 2.2204e-016 (distance entre 1.0 et le flottant le plus proche) ; cette valeur peut être modifiée,
Inf (Infinite) : nombre infini,
NaN (Not a Number) : n'est pas un nombre, exprime parfois une indétermination.
ans : variable contenant la dernière réponse.
i et j représentent tous deux le nombre imaginaire unité $(-1)^{1/2}$.

realmin : désigne le petit nombre réel positif ;
realmax : désigne le plus grand nombre réel positif.

II.2 Variable d'environnement

who : affiche la liste des variables présentes dans l'espace de travail
whos : affiche la liste des variables présentes dans l'espace de travail ainsi que leurs propriétés
class (var) : affiche classe (type) de variable (nommée var).
size(A) : affiche les dimensions de la matrice A.
clear var1 var2… efface les variables var1 var2…
clear **(ou clear all)** : efface toutes les variables crées dans l'espace de travail.
Clc : efface Command Window.

Exemple

```
>> x=2*6-9; y=[1 2 3]; z=[1 2 3;4 5 6]; t='bonjour';
>> who
Your variables are:
t  x  y  z
>> whos
  Name      Size              Bytes  Class     Attributes
  t         1x7                  14  char
  x         1x1                   8  double
  y         1x3                  24  double
  z         2x3                  48  double
```

II.2.1 Types de variables

Il existe quatre types de variables sous Matlab : les réels, les complexes, les chaînes de caractères et le type logique.

II.3 Opérations arithmétiques

Les opérations arithmétiques de base dans Matlab sont : + pour l'addition (x+y), - pour la soustraction, (x-y), * pour la multiplication (x*y), et / ou \ pour la division (x/y), (x\y). La division à droite et la division à gauche ne donnent pas

les mêmes résultats, ce sont donc deux opérations différentes que ce soit pour les scalaires ou pour les vecteurs et matrices. ^ Pour la puissance (x^y).
Les opérations peuvent être enchaînées en respectant les priorités usuelles des opérations et en utilisant des parenthèses.

Exemples

```
>> x=2+(6-4*5)
x =
   -12
>> y=2*x+5^3
y =
   101
>> z=x/y
z =
   -0.1188
>> t=x\y
t =
   -8.4167
```

II.4 Format des nombres et précision des calculs

MATLAB a la possibilité d'afficher les valeurs des variables dans différents formats :
Flottants courts (short), flottants longs (long), flottants courts en notation scientifique
(short e), flottants longs en notation scientifique (long e), ainsi que les formats monétaire, hexadécimal et rationnel (notation sous forme d'une fraction).

Exemple

```
>> format  short
>> 22/7
ans =
    3.1429
>> format  long
>> 22/7
ans =
```

```
   3.142857142857143
>> format  short e
>> pi
ans =
   3.1416e+00
>> format  long e
>> pi
ans =
    3.14159265358979e+00
>> format  rat
>> 0.564
ans =
     141/250
```

Chapitre III

III Vecteurs et Matrices

III.1 Les vecteurs

L'élément de base pour MATLAB est une matrice à éléments complexes.

-Tout nombre réel est considéré comme une matrice à une ligne et une colonne dont la partie imaginaire nulle.

- Un vecteur est une matrice à une ligne ou à une colonne.

-Les vecteurs servent aussi à représenter les polynômes et les chaînes de Caractères.

-Vecteur ligne : les composantes sont arrangées horizontalement

-Vecteur colonne : les composantes sont arrangées verticalement.

III.1.1 Création de vecteurs

-Pour créer un vecteur ligne il suffit d'écrire ses composantes entre crochets **[ ]** et de les séparés par des blancs ou des virgules (,).

Exemple

```
>> V1=[1  -7   1/2   0]
V1 =
   1   -7   0.5   0
```

```
>>V2=[10,6,3]
V2 =
   10    6    3
```

-Pour créer un vecteur colonne il suffit d'écrire ses composantes entre crochets **[ ]** et de les séparés par des points-virgules (;).

Exemple

```
>> V3=[1 ;-5 ;9]
V3 =
    1
   -5
    9
```

```
On peut écrire le vecteur verticalement,
>>V3=[1
        -5
         9 ] ;
```

Remarques

1) Si les composantes d'un vecteur sont équidistantes (espacées d'un pas constant) et si la première et la dernière valeur sont connues, alors ce vecteur peut être décrit en fonction d'un outil puissant de Matlab est l'opérateur « : » de la manière suivante :

V=début : pas : fin

Exemple

```
>>V=-1: 0.5 : 1
   V=
      -1.0000   -0. 5000  0.0000   0. 5000   1.0000
```

Si le pas est égal à un, on écrit simplement,

V=début : fin

Exemple

```
>>W=-2 : 2
    W=
      -2.0000   -1.0000  0.0000   1. 0000   2.0000
```

2) La commande linspace

La commande linspace permet de générer un ensemble de n éléments (vecteur ligne) en spécifiant la première et la dernière valeur.

syntaxe : linspace (début, fin, nombre d'éléments).

Le pas d'incrémentation est calculé automatiquement par Matlab selon la formule : le pas=(fin-début)/(nombre d'éléments-1) .

Exemple

```
>>t=linspace(1,2,5)
      t=
        1.0000   1.2500   1.5000   1.7500   2.0000
```

Ou bien la syntaxe : linspace (début, fin) avec le nombre d'éléments =100 par défaut.

3) La taille d'un vecteur (le nombre de ses composants) peut être obtenue avec la fonction length comme suit :

Exemple

```
>>t=linspace(1,2,5) ;
>> length(t)
 ans =
          5
```

```
>> length (linspace(0 , 5) )
ans =
     100
% length : longueur du vecteur
```

III.1.2 Opérations élémentaires

Pour réaliser des calcules élément par élément sur les vecteurs en utilisant les opérations suivantes :

1) Addition et soustraction : +, -

Exemple

```
>> V = [3  5  -2  11];  W = [1  8  4  0];
 >> V+W
ans =
     4   13  2  11
```

```
>> V-W
 ans =
      2   -3  -6  11
```

2) Multiplication et division : *, /, \, ^

Lorsqu'on veut réaliser des opérations entre les éléments de deux vecteurs, pris un à un. Les opérateurs *, /, \, ^ sont précédés du signe "point(.)"

Exemples

```
>> x = [3  1  -2  2];  y = [-1  5  4  0];
>> m=x.*y
m=
    -3     5    -8     0
```

```
>> x*y
Error using  *
Inner matrix dimensions must agree.
```

```
>> d=x./y
d =
   -3     0.2   -0.5       Inf
```

```
>> t=x.\y
t =
   -0.3333     5   -2.       0
```

```
>> p=x.^2
p =
     9     1     4     4
```

```
>> q=x.^y
q =
   0.3333    1  16   1
```

```
>> x^2
Error using ^ Inputs must be a scalar and a square matrix.
To compute elementwise POWER, use POWER (. ^) instead.
```

Attention L'écriture: x^2 génère une erreur car cette expression réfère a une multiplication de matrices x*x.

3) Transposition

Pour réaliser certaines opérations vectorielles, on est amené à transformer un vecteur ligne en vecteur colonne et inversement.

La transposition d'un vecteur est réalisée en faisant suivre son nom par une apostrophe

Exemple

```
>> x = [3  1  -2];
>> T=x'
T =
     3
     1
    -2
```

```
>> x''
ans =
     3     1    -2
```

III.2 Les matrices

Les matrices représentent le cœur du Matlab.

III.2.1 Création de matrice

Pour saisir manuellement des (petites) matrices, il faut respecter les règles suivantes :

a) Les s éléments doivent être mises entre des crochets **[ ]**.

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

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

On peut insérer une matrice avec une des syntaxes suivantes :

Soit la matrice $M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

```
>> M= [1 ,2 ,3; 4 ,5 ,6; 7, 8 ,9] ;
>> M= [1     2     3                    %Retour à la ligne avec la touche« ENTER »
     4     5     6
     7     8     9] ;
```

```
>> M= [[1 ,2 ,3];[4 ,5 ,6];[7, 8 ,9]] ;          % Création par lignes
```

```
>> M= [[1 ; 4 ; 7], [2 ; 5 ; 8], [3 ; 6 ; 9]] ;          % Création par colonnes
```

```
>>x= [1 ,2 ,3]; y= [4 ,5 ,6]; z= [7, 8 ,9] ;
>>M=[x ; y ; z] ;                    % M est formée par les vecteurs lignes x, y et z
```

```
>>M= [x', y', z'] ;                % M est formée par les vecteurs colonnes x, y et z
```

Attention

Une erreur sera signalée par MATLAB, si le nombre d'éléments dans chaque ligne (colonne) n'est pas identique dans toutes les lignes (colonnes) de la matrice.

Exemple

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

Error: Dimensions of matrices being concatenated are not consistent.

III.2.2 L'accès aux éléments d'une matrice

Soit M une matrice de m ligne et n colonne

- M(I, j) désigne l' élément de la ligne **i** et la colonne **j** de la matrice M.
- M (i, :) désigne toute la ligne numéro **i** de la matrice M.
- M (:, j) désigne toute la colonne numéro **j** de la matrice M.

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

Nom matrice (positions lignes, positions colonnes)

Exemples

Soit la matrice $M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

>> M= [1 2 3;4 5 6; 7 8 9] ;	
>> M(2,3)%L'élément sur la ligne 2 et la colonne 3 ans = 6	>> M (2, :) % Tous les éléments de la ligne 2 ans = 4 5 6
>> M (:,1) % Tous les éléments de la colonne 1 ans = 1 4 7	

Remarque

La position peut être un simple numéro, ou une liste de numéro (un vecteur de positions)

Exemple

```
>> M (1:2, :)                    %Tous les éléments de la 1 ère et la 2ème ligne .
ans =
    1    2    3
    4    5    6
>> M (: , [1,3])                 %Tous les éléments de la 1 ère et la 3ème colonne .
ans =
    1    3
    4    6
```

```
    7     9
>> M (2:3, [1,3])
ans =
    4     6
    7     9
```

III.2.3 Génération automatique des matrices

En Matlab, il existe des fonctions qui permettent de générer automatiquement des matrices particulières. Certaines matrices apparaissent souvent dans les calculs scientifiques. Matlab offre la possibilité de générer ce type de matrices spéciales afin de faciliter la création des matrices surtout pour les problèmes de dimension élevé.

-Pour les matrices rectangulaires, on précisera le nombre de lignes et de colonne et pour les matrices carrées, on spécifiera uniquement l'ordre.

La fonction	Signification
eye (n,m)	Génère une matrice n × m avec tous les éléments si l'indice (i=j) =1 sinon 0
eye(n)	Génère une matrice n × n avec tous les éléments si l'indice (i=j) =1 sinon 0
zeros(n,m)	Génère une matrice n × m avec tous les éléments = 0
zeros(n)	Génère une matrice n × n avec tous les éléments = 0
ones(n,m)	Génère une matrice n × m avec tous les éléments = 1
ones(n)	Génère une matrice n × n avec tous les éléments = 1
diag (v), v :vecteur	Génère une matrice diagonale avec v comme diagonale (0 ailleurs)
diag (M), M:matrice	extraction de la diagonale de la matrice M sous forme de vecteur colonne
rand(n,m)	Génère une matrice de n × m de valeurs aléatoires (entre 0 et 1)
magic(n)	Génère une matrice magique de dimension n × n

Exemples

```
>> eye(2,3)
ans =
    1     0     0
```

```
>> eye(3)
ans =
    1     0     0
```

0 1 0	0 1 0 0 0 1
>> zeros(2,3) ans = 0 0 0 0 0 0	>> zeros(2) ans = 0 0 0 0
>> ones(2,3) ans = 1 1 1 1 1 1	>> ones(3) ans = 1 1 1 1 1 1 1 1 1
>> v=[1 2 3]; >> diag(v) ans = 1 0 0 0 2 0 0 0 3	>> M=[14 5 3 7;4 5 -1 6; 7 8 -4 0 ; 9 3 11 8]; >> diag(M) ans = 14 5 -4 8
>> rand(2,3) ans = 0.8147 0.1270 0.6324 0.9058 0.9134 0.0975	>> magic(4) ans = 16 2 3 13 5 11 10 8 9 7 6 12 4 14 15 1

III.2.4 Opérations de base sur les Matrices

1) Les opérations élément par éléments +, -, .*, .^, ./ sur les matrices de mêmes dimensions sont les mêmes que ceux pour les vecteurs.
2) Les opérations habituelles +, -, *, ^, / sont matricielles. Si une des matrices (pour les opérations *,^, /) est un scalaire. Dans ce cas, les opérations s'appliqueront élément par éléments.

Exemples

>> A=[1 2; 3 4]; B=[5 6;7 8]; >> A+B	>> A-B ans =

ans = 6 8 10 12	-4 -4 -4 -4
>> A.*B ans = 5 12 21 32	>> A*B ans = 19 22 43 50
>> A./B ans = 0.2000 0.3333 0.4286 0.5000	>> A/B % A/B=A*B^{-1} ans = 3.0000 -2.0000 2.0000 -1.0000
>> A.^2 ans = 1 4 9 16	>> A^2 % A^2=A*A ans = 7 10 15 22
>> A.\B ans = 5.0000 3.0000 2.3333 2.0000	>> A\B % A\B=A^{-1}*B ans = -3 -4 4 5

III.2.5 Manipulation de matrices

A'	transposée de la matrice A
A (:)	écrire A sous forme d'un vecteur colonne
rechape (A, n, m)	retourne une matrice de dimension n×m, par les colonnes de A
rot90(A)	rotation 90° de matrice A
tril(A)	extraire la partie triangulaire inférieure.
triu(A)	extraire la partie triangulaire supérieure.
fliplr(A)	retournement gauche-droite

Exemples

>> A = [2 0 5 4; 1 -8 6 12; 4 9 3 7;0 -5 8 3]	>> tril(A) ans =	>> reshape(A,8,2) ans =

```
A =
    2    0    5    4
    1   -8    6   12
    4    9    3    7
    0   -5    8    3
>> A'
ans =
    2    1    4    0
    0   -8    9   -5
    5    6    3    8
    4   12    7    3
```

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

```
    2    5
    1    6
    4    3
    0    8
    0    4
   -8   12
    9    7
```

```
>> triu(A)
ans =
    2    0    5    4
    0   -8    6   12
    0    0    3    7
    0    0    0    3
```

```
>> fliplr(A)
ans =
    4    5    0    2
   12    6   -8    1
    7    3    9    4
    3    8   -5    0
```

```
>> rot90(A)
ans =
    4   12    7    3
    5    6    3    8
    0   -8    9   -5
    2    1    4    0
```

III.2.6 Factorisation de matrices

1) Factorisation de Cholesky

La factorisation de Cholesky d'une matrice A définie positive consiste en une décomposition de type : A = B'*B

B : matrice triangulaire inférieure régulière,

B' : transposée de B.

MATLAB prévoit la fonction chol pour cette factorisation.

Exemple

```
>> A = [1 1 1 ; 1 2 3 ; 1 1 6];
>> B=chol(A)
B =
    1    1    1
    0    1    2
    0    0    1
```

2) Factorisation LU

Pour toute matrice carrée régulière A, on peut écrire la décomposition suivante :

P * A = L * U

U : matrice triangulaire supérieure,

L : matrice triangulaire inférieure à diagonale unité,

P : matrice de permutation.

Cette décomposition peut être obtenue par la fonction lu

Exemple

```
>> A = [1 0 1 ; 1 2 3 ; 1 1 6];
>> [L, U, P] = lu(A)
L =
    1.0000         0         0
    1.0000    1.0000         0
    1.0000    0.5000    1.0000
U =
     1     0     1
     0     2     2
     0     0     4
P =
     1     0     0
     0     1     0
     0     0     1
```

III.2.7 Fonctions Matricielles

Nous présentons quelques fonctions les plus utilisées concernant les matrices, définies dans Matlab

La fonction	L'utilité	Exemple
det(A)	déterminant de A, (A carrée)	>> A=[2 5;7 9]; >> det(A) ans = -17
Inv(A) ou A^(-1)	A^{-1}: l'inverse de A	>> inv(A) ans = -0.5294 0.2941 0.4118 -0.1176
sum(A)	la somme des éléments de chaque colonne.	>> sum(A) ans = 9 14

prod(A)	la produit des éléments de chaque colonne.	>> prod(A) ans = 14 45
trace (A)	la trace de A,(A carrée), la somme des éléments sur la diagonale	>> trace(A) ans = 11
norm(A)	la norme (euclidienne) de A	>> norm(A) ans = 12.5364
eig(A)	les valeurs propres de A, (A carrée)	>> eig(A) ans = -1.3739 12.3739
poly(A)	les coefficients du polynôme caractéristique associé à la matrice A.	>> poly(A) ans = 1 -11 -17.

III.3 Serie d'exercices

III.4 Création et utilisation des fichiers-M (M-file)

- Définir la variable $x=\cos(2\pi/3)$ dans la Command Window.

- Créer un fichier script:

Aller dans le menu→ File →New →Script (ou M-file)

(1) Ecrire : $y=\log(x^2+1/2)$ dans ce script.

(2) Sauvegarder le fichier avec le nom TP01 par exemple dans un répertoire différent de celui proposé par Matlab (dans un dossier se trouvant sur le bureau).

(3) Revenir à la Command Window, puis appeler le fichier script en utilisant la commande : >> TP01, si Matlab ne reconnaît pas le fichier, alors il faut ajouter le répertoire qu'on vient de créer au Matlab Path comme suit:

a) Aller dans le menu →File→Set Path,

b) Cliquer sur le bouton Add Folder,

c) Sélectionner le répertoire,

d) Cliquer sur le bouton Close (Si la fenêtre de dialogue Save Path s'ouvre, cliquer sur le bouton non).

Créer un fichier fonction:

(1) Aller dans le menu →File→New→function (ou M-file)

(2) Ecrire:

```
function [y]=fct1(x)
y= log(x^2+1/2) ;
 end
```

(3) Sauvegarder le fichier avec le nom fct1 (obligatoire : le même nom de fonction, pour l'appel. . .) sur le bureau.

(4) Revenir à la Command Window, puis appeler le fichier function en utilisant la commande :>> fct1(cos(2π/3))

Exercice 01

Utiliser Matlab pour faire les calculs suivants

$x=\sqrt{3}+1/2$, $y=x^2-1$, $z=e^{\sqrt{x}}$, $t=(x+1/y)/Z$

$a=(k^3+2ki)$, pour $k=1+i$, $(i^2=-1)$

Exercice 02

1-Définir les vecteurs V = [0, 1, 2 ,…,99 ,100] et w = [1 , 3 ,5, …,99]
Quelle est la taille de V ? de w ?
-Définir le vecteur x contenant les dix premiers éléments de V, et le vecteur y contenant les dix premiers et les dix derniers éléments de w.
-Définir le vecteur z = [0, 4, 8,…,92 ,96 100] à partir de V

Exercice 03

1-Créez les vecteurs suivants en utilisant l'opérateur (:).

$$u=\frac{1}{100},\frac{2}{99},\frac{3}{98},\dots,\frac{98}{3},\frac{99}{2},100\,,\ v=1,\frac{1}{2},\frac{1}{3},\dots,\frac{1}{98},\frac{1}{99},\frac{1}{100}\,.$$

2-Soit n un entier naturel. Donner les commandes Matlab permettant de calculer les expressions suivantes.

$$V=\ \frac{1}{n}+\frac{2}{n-1}+\frac{3}{n-2}+\cdots+\frac{n-2}{3}+\frac{n-1}{2}+n\,.$$

$$w=1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n}\ .$$

Exercice 04

Créez deux vecteurs x et y, et regardez ce que donne les opérations suivantes

x - y; x * y; x .*y; x^2; x.^3; x/y; x./y
Faites de même avec deux matrices A et B. Que ce passe-t-il si elles ne sont pas de même taille ?

Exercice 05

Soit la matrice $M=\begin{pmatrix}3 & 9 & 0 & -5\\ 4 & 2 & 1 & -6\\ 2 & 1 & 5 & 2\\ 0 & 8 & 11 & 3\end{pmatrix}$

1-Donner les réponses à chacune des commandes suivantes :

>>M(1:2, 2 :3)	>>M (: ,end)
>>M(:,3 :4)	>>M([2 3],end)
>> size(M)	>>M(end, end)
>>fliplr(M(end,:))	>> reshape(M(3 :4, :),1,8)
>> length (diag(M))	>>sum(M(: ,end))

>> size (M ([2,3], [2:end]) ,2) >> diag (diag (A ([3,4],[2,3])	>>rot90(M([3,4],[2,3])

2 -Ecrire les commandes matlab qui permettent de

Extraire les éléments diagonaux de M Supprimer la deuxième colonne de M Supprimer la quatrième ligne de M Extraire la sous matrice A obtenue par suppression de la troisième ligne et deuxième colonne de M Permuter les lignes 2 et 4 Insérer la ligne (9 8 7 6) en deuxième position Ajouter la colonne (2 ;3 ;4 ;5) en fin

Exercice 06

1 -Créez les matrices suivantes en utilisant les matrices particulières

$$A=\begin{pmatrix} 3 & 3 & 3 \\ -5 & 6 & 6 \\ -5 & 6 & 6 \end{pmatrix}, \quad B=\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}$$

2 -**Testez les fonctions suivantes :**

diag(A), det(A), det(B), A^(-1), inv(A), tril(A) , triu(A), fliplr(A) sum(A), prod(A), trace (B), norm(A), eig(B), poly(A)

Chapitre IV

IV Programmation avec MATLAB

Matlab peut exécuter une suite d'instructions stockées dans un fichier appelé fichier M (M-file). Il ya deux types de fichiers M : les fichiers scripts (fichiers de commandes) et les fichiers de fonctions.

IV.1Lecture et affichage des données dans un programme

IV.1.1 Les entrées :

-Saisie au clavier : La fonction input

La fonction input effectue la saisie de valeurs numériques ou de textes (chaîne de caractères).

- Pour les valeurs numériques, n = input ('message'), affiche message et Affecté à la variable n la valeur numérique entrée au clavier.
- Pour les chaînes de caractères, str = input ('message','s') affiche message et affecte à la variable str la valeur entrée au clavier considérée comme une chaîne de caractères.

Exemples

```
>> n = input ('Entrez la valeur de n ')
>> nom = input('Entrez votre nom ','s')
```

IV.1.2 Les sorties (disp - error - num2str - fprintf-sprintf)

-Le moyen le plus simple pour afficher la valeur d'une variable est en tapant seulement le nom de cette dernière.

Exemple

```
>> x=12 ;
>>x
>>x=
    12
```

Remarque : Le texte venant après le signe % est un commentaire, donc il n'est pas traité par Matlab.

-La commande disp

La fonction disp permet d'afficher proprement des valeurs numériques ou bien du texte (chaine de caractères) à l'écran, sans faire référence au nom de la variable et n'affiche rien si le tableau est vide. La commande disp sera souvent utilisée avec num2str pour afficher les valeurs des expressions numériques.

Exemple

>> x=[1 0 ;3 7] ; >>x x=1 0 3 7	>>disp(x) 1 0 3 7
>> disp(['la valeur de exp(1)est': ,num2str(exp(1))]); la valeur de exp(1) est : 2.7183	

-La commande : error :

Dans le cas d'une commande illégale, Matlab retourne un message d'erreur généralement très précis et instructif. L'utilisateur peut également ajouter d'autres messages d'erreur qui dépend du son propre script en utilisant la commande error, qui a la syntaxe suivante :

Error ('message').

- La commande :fprintf

La commande fprintf( ) accepte tous genre de variable en utilisant leurs formats de types :

%d pour les entiers, %f pour les réels, %s pour les chaines de caractères, %c pour les caractères, … . On peut ajouter un retour à la ligne via '\n' et une tabulation via '\t'.

-La commande : sprintf

La commande sprintf ( ) sauvegarde tous genre de variables de la même manière que fprintf , mais sans affichage sur la fenêtre des commandes. Sprintf ( ) retourne une chaîne de caractères qui peut être affichée ultérieurement en utilisant la commande disp ( ).

Exemples

>> x=2; y=0.125; z=x/y;

```
>> fprintf('le quotient de %d et %f égale à %f \n', x, y, z);
le quotient de 2 et 0.125000 égale à 16.000000
```

```
>> x=2;y=0.125;z=x*y;
>> str = sprintf('%d * %f= %f \n', x, y, z);
>> disp(str);
2 * 0.125000= 0.25000
```

IV.2Scripts et fonctions

IV.2.1 Les fichiers scripts

-Un script ou un programme, est une suite d'instructions Matlab (commandes) séparées par une virgule, un point-virgule ou par le symbole de saut de ligne.
-Ces instructions sont écrites dans un fichier texte, enregistré avec l'extension .m
-Pour créer un fichier-M (fichiers scripts) aller dans le menu File→New→Script ou M-file ou cliquez sur le bouton de la page blanche.

-Pour appeler le fichier script en utilisant la commande : >> nondefichier,
- Pour exécuter un script, il faut évoquer son nom de fichier dans la ligne de commande de Matlab.

Remarque

Les variables définies dans le scripts restent dans la mémoire de Matlab après l'exécution du script.

Exemple

Ecrire un programme MATLAB qui permet de calculer les éléments de la matrice S, la somme de deux matrices A et B de dimensions m*n chacune.

```
%Ce Programme (script) calcule la somme de deux matrices A et B.
% Ce programme enregistré avec le nom : somme
1   A=input( 'Entrez la matrice A =') ;
2   B=input( 'Entrez la matrice B =') ;
3   S=A+B ;
4    S
```

Pour excuter ce programme taper somme dans la fenêtre de commande suivi de < entrer > comme suit :

```
>>  somme
>> Entrez la matrice A =[1  3  8; 0 -5  9; 2  3  6]   % suivi de < entrer >
>> Entrez la matrice B =[-1  5  9; 1  5  2; 0  13  4]  % suivi de < entrer >
```

```
S =
    0     8    17
    1     0    11
    2    16    10
```

IV.2.2 Les fichiers de fonctions

Une fonction est un script (programme) qui :

- reçoit des arguments en entrée
- renvoie des résultats
- les variables définies localement seront effacées après l'exécution de la fonction

-Syntaxe

```
function [Y1,Y2,...] = NomFonc (X1,X2,...)
.....
end
```

où

- X1, X2, … : sont les arguments de la fonction (les entrées)
- Y1, Y2, … : sont les résultats (les sorties)
- La fonction doit être enregistrée dans un fichier portant le nom de la fonction et l'extension « .m » (NomFonc.m)
- Les commentaires rédigés juste après la première ligne seront affichés lorsqu'on évoque.

Exemple

```
%Ce Programme (script) calcule la somme de deux matrices A et B.
% Ce programme enregistré avec le nom : sommeAB
1  function [S] = sommeAB( A,B )
2  S=A+B;
3  end
```

Pour excuter ce programme entrez les matrices A et B et puis taper sommeAB(A,B) dans la fenêtre de commande suivi de < entrer > comme suit :

```
>> A=[1  3  8; 0 -5 9; 2  3 6];
>> B=[-1  5  9; 1 5  2; 0  13  4];
>> somme(A,B)
S =
    0     8    17
    1     0    11
```

```
    2    16    10
```

IV.3 Opérateurs de comparaison et opérateurs logiques

Les opérateurs de comparaison sont :

==	égal à (x == y)
<	strictement plus petit que (x < y)
>=	plus grand ou égal à (x >= y)
<=	plus petit ou égal à (x <= y)
~ =	différent de (x ~ = y)

Les opérateurs logiques sont:

&	et (x & y)
|	ou (x | y)
~	non (~ x)

Pour la comparaison des vecteurs et des matrices on utilise les deux fonctions isequal et isempty.

isequal	teste si deux (ou plusieurs) matrices sont égales (ayant les mêmes éléments partout). Elle renvoie une valeur logique 1ou 0.
isempty	teste si une matrice est vide.Elle renvoie 1 si c'est le cas, et 0 sinon.

Exemples

```
>> A=[1  2  3 ;0  9  8 ;0  7  5]
A =
    1    2    3
    0    9    8
    0    7    5
```

```
>> B=[7  2  3 ;0  7  5 ;1  7  2]
B =
    7    2    3
    0    7    5
    1    7    2
```

```
>> A==B
ans =
    0    1    1
    1    0    0
    0    1    0
```

```
>> A~=B
ans =
    1    0    0
    0    1    1
    1    0    1
```

```
>> isequal(A,B)
ans =
```

```
>> isempty(A)
ans =
```

0	0
>>C= [ ] % C est une matrice vide	>> isempty(C) ans = 1

IV.4Les instructions de contrôle

Plusieurs commandes permettent de contrôler l'exécution des instructions.

MATLAB peut être vu comme un langage de programmation de haut niveau.

IV.4.1 l'instruction conditionnelle : if

Cette instruction utilisée qu'on a besoin d'exécuter une séquence (suite) d'instructions (commandes) seulement dans le cas où une condition donnée est vérifiée au préalable.

Il y a plusieurs formes d'instruction conditionnée existent sous Matlab.

1-Forme : if ...end

Syntaxe :

```
if  expression logique   % expression dont le résultat peut être vrai ou faux.
Liste de commandes  % Le traitement effectuer si expression logique est vraie.
end
```

Remarque : Si l'expression logique est faux, en passe directement à l'instruction qui suit le mot clé end.

2-Forme: if ...else...end

-Syntaxe:

```
if    expression logique (conditions)
 Liste 1 de commandes (instructions 1)
  else
Liste 2 de commandes (instructions 2)
 end
```

Remarques1- Si l'expression logique est vrai, on exécute Liste 1 de commandes
Si l'expression logique est faux, on exécute Liste 2 de commandes.

3- Forme : if...elseif...else....end

```
if    expression logique 1 (conditions 1)
```

```
 Liste 1 de commandes (instructions 1)
  elseif  expression logique 2 (conditions 2)% pas d'espace entre else et if, elseif
Liste 2 de commandes (instructions 2)
else
Liste 3 de commandes (instructions 3)  %liste de commandes par défaut
end
```

Remarque

Si Condition1 est vraie, alors Liste1 est exécutée, sinon et si condition2 est vérifiée alors il y a exécution de Liste2, autrement c'est Liste3.

Exemple

```
1  x=input ('entrez la valeur de x=')
 2   if mod(x, 2) ==0            %reste de la division de x par 2 égal à 0
 3      r='x est pair'
 4   elseif  mod(x, 2) ==1       %reste de la division de x par 2 égal à 1
 5      r='x est impair'
6    else
7     r='x n'est pas un entier'
8    end
```

IV.4.2 L'instruction conditionnelle switch

-Syntaxe

```
switch   Var      % var est une variable  numérique ou une chaîne de caractères.
 case const1

instructions 1

case const2  %const i (i=1:n) sont des constantes numérique ou chaîne de
caractères;
instructions 2
...
otherwise
 instructions n              %liste de commandes par défaut
end
```

Exemple

```
1  x=input ('entrez la valeur de x=')
 2   switch   mod(x, 4)          %reste de la division de x par 4
```

```
3   case 0,
4      r=0
5   case 1,
6      r=1
7   case 2,
8     r=2
9  case 3,
10     r=3
11   otherwise
12  disp('r n"est pas un entier')
end
```

IV.4.3 Boucles for et while

Deux commandes sont utilisées pour répéter l'exécution d'instructions : for et while.

-La syntaxe de la boucle for et while:

for variable = expression instructions end	while expression logique(condition) instructions end

-expression est usuellement de la forme: variable= i : h :k ou variable= i : j.

Remarque On peut sortir d'une boucle à l'aide de l'instruction break.

Exemple.

Ecrivez un programme qui calcule le plus grand cube inférieur ou égal à n.

```
1  n= input( 'Donnez la valeur de n=')
2    i=0;
3   while i^3<=n
4     i=i+1;
5    end
6   r=(i-1)^3
```

Exemple

```
for j=1:10
x=2^(j)
y=(1/2 )*log(j)
z=x*exp(y)
end
```

Chapitre V

V GRAPHISME EN MATLAB

Fonctions inline

Nous avons vu un certain nombre de fonctions prédéfinies. Il est possible de définir ses propres fonctions dans un fichier (fichier de fonction).

Matlab possède une commande en ligne employée pour définir ses propres fonctions, dites inline.

Exemple

```
>> f=inline('x^2-3*x+1','x')
f =
     Inline function:
     f(x) = x^2-3*x+1
```

On peut maintenant utiliser cette nouvelle fonction :

```
>>f(2)
ans=
     -1
>>f(0)
ans=
      1
```

```
>> f(1/2)
ans =
   -0.2500

>> f(sqrt(2))

ans =
   -1.2426
```

On peut appliquer cette fonction à un tableau de valeurs :

Exemple

```
>> x=[-2 -1  0 ] ;
>> f(x)
ans =
   11     5     1
```

```
>> f(0:4)
Error in inline expression ==> x^2-3*x+1
 Inputs must be a scalar and a square matrix.
To compute elementwise POWER, use POWER (.^)
instead.
```

MATLAB essaye de multiplier x vecteur ligne par x aussi vecteur ligne au sens de la multiplication de matrice ! Par conséquent il faut bien utiliser une multiplication élément par élément (.^, .*, ./) dans la définition de la fonction comme suit

```
>> f=inline('x.^2-3*x+1','x')
f =
```

```
>> f(0:4)
ans =
```

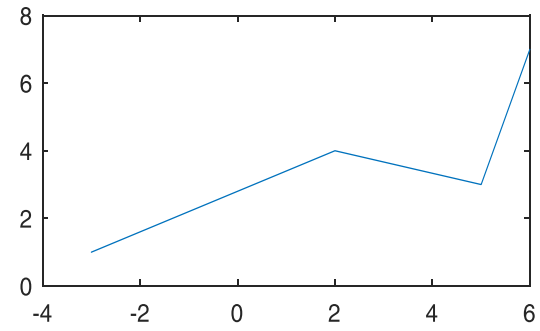
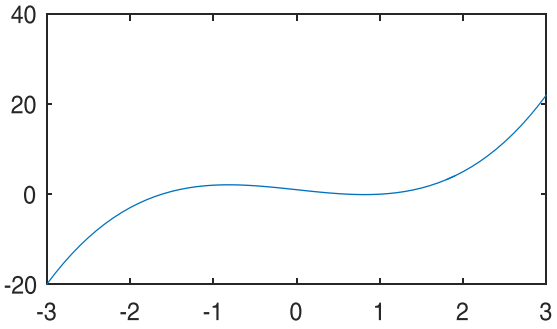
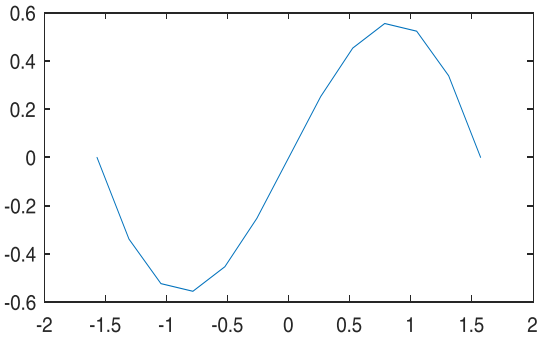
Inline function: f(x) = x.^2-3*x+1	1 -1 -1 1 5

V.1 Graphisme 2D :

V.1.1 La commande plot

La commande plot permet de tracer un ensemble de points de coordonnées (x(i),y(i)), i=1,2, …,N. Les points (xi, yi) sont reliés entre eux par des segments de droites.

Exemples

1 x= [-3 2 5 6]; 2 y= [1 4 3 7]; 3 plot(x, y)	
1 x= [-3:0.1:3]; 2 f=x.^3-2*x+1; 3 plot(x,y)	
1 x=[-pi/2:pi/12:pi/2]; 2 f=inline('x.*cos(x)'); 3 plot(x,f(x))	

V.1.2 La commande fplot

La commande fplot permet de tracer le graphe d'une fonction sur un intervalle donné, elle assure une représentation plus fine que plot.

syntaxe

fplot (' fonction ' , [xmin , xmax])

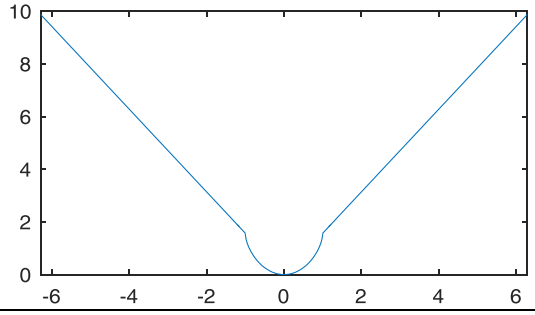
ou

fonction : est une chaine de caractères qui décrit la fonction à tracer.

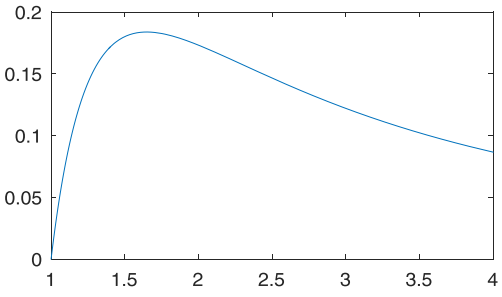
[xmin , xmax] est l'intervalle pour lequel est tracé le graphe de la fonction.

Exemples

```
1 fplot('x.*asin(x)',[-2*pi,2*pi])
```




```
1 function y=fon(x)
2 y=(1./(x.^2)).*log(x)
3 end
>>fplot('fon',[1,4])
```



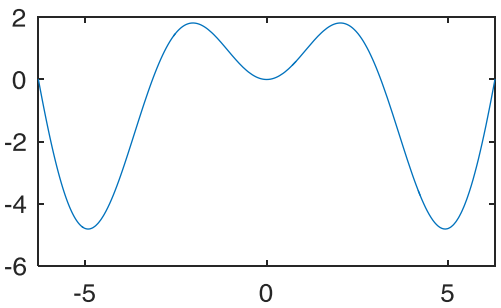
V.2 Tracer plusieurs graphes

-La commande figure

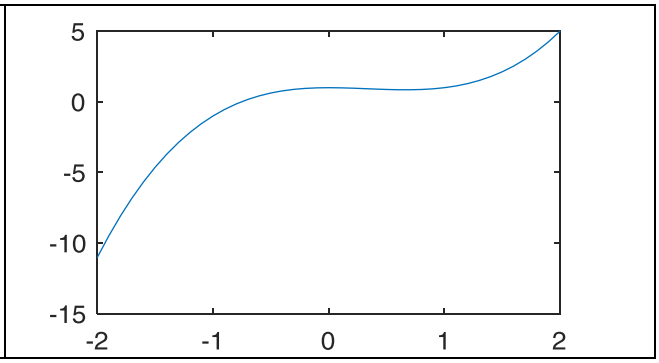
Pour tracer plusieurs graphes dans des fenêtres différentes on utilise la commande figure pour chaque nouveau graphe.

Exemple

```
1 fplot('x.*sin(x)',[-2*pi,2*pi])
2 figure
3 x= [-2:0.1:2];
4 y=x.^3-x.^2+1;
```



```
5  plot(x,y)
```

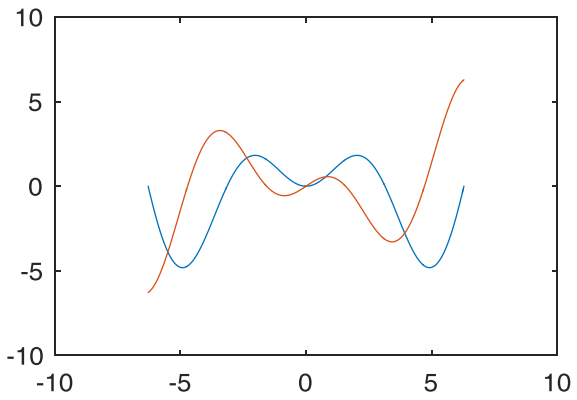


-La commande hold on

Pour tracer plusieurs graphes dans la même fenêtre on utilise la commande hold on

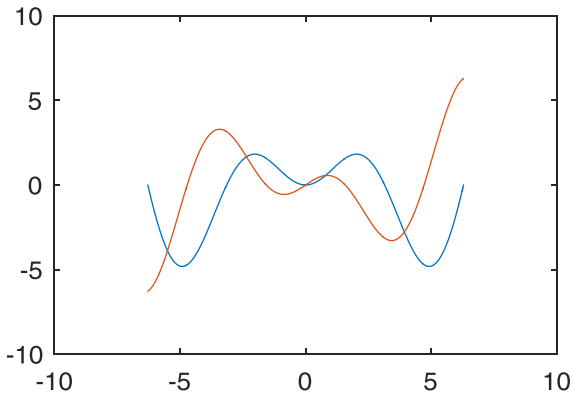
Exemple

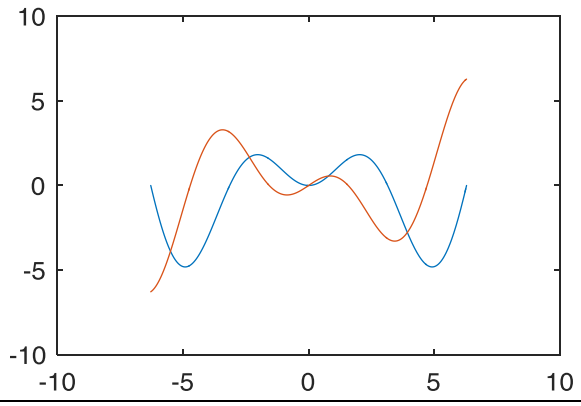
```
1  x=[-2*pi:pi/100:2*pi];
2  f1=inline('x.*sin(x)');
3  f2=x.*cos(x)
4  plot(x,f1(x))
5  hold on
6  plot(x,f2)
7  hold of
```



On peut également tracer plusieurs fonctions sur la même figure la commande plot et fplot de la manière suivante :

```
1  x=[-2*pi:pi/100:2*pi];
2  f1=inline('x.*sin(x)');
3  f2=x.*cos(x);
4  plot(x,f1(x),x,f2)
```



fplot('[x.*sin(x),x.*cos(x)]',[-2*pi,2*pi])	

V.2.1 Modification de l'apparence d'une courbe

Titres, labels, grille et legend

La commande title ajoute un titre au graphique de la fenêtre courante.

```
>>title ('Graphe de y = x*sin(x)')
```

Pour rajouter des labels d'axes à notre figures, on utilise les commandes :
xlabel, ylabel

```
>>xlabel ('Axe x')                %l'axe des abscisses

>>ylabel ('Axe y')                %l'axe des ordonnées
```

Pour ajouter des grilles, on fait appel à la commande :grid on

```
>>grid  on                %affichage de la grille

>>grid off                %masquage de la grille
```

Il est indispensable d'ajouter une légende en utilisant la commande legend

```
>>legend( 'y=f(x)')
```

Couleurs et styles du tracés

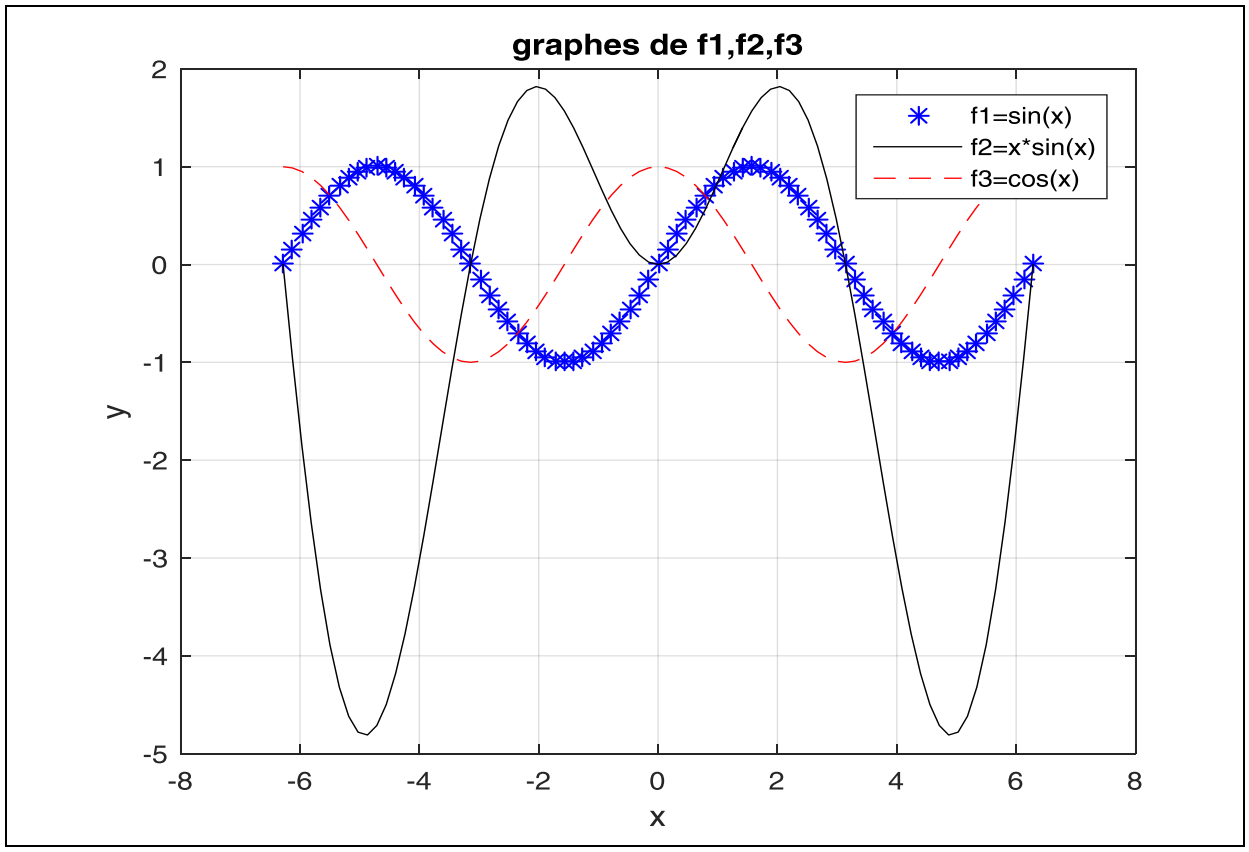
Par défaut, les tracés sont bleus en trait plein. Il est possible de modifier le style et la couleur de la tracé en rajoutant un 3_eme argument à la fonction plot ( ).

Les couleurs et styles du tracé

Couleurs			Styles du tracé	
B	Bleu	blue	.	point
g	Vert	green	°	cercle
r	Rouge	red	×	x-mark
c	cyan	cyan	+	plus
m	Magenta	magenta	-	Ligne continue
y	Jaune	yellow	*	étoile
k	Noir	black	--	tirets
w	Blanc	white	- .	Tirets point

Exemple

```
x=[-2*pi:pi/20:2*pi];

f1=sin(x);

f2=x.*sin(x);

f3=cos(x);

plot(x,f1,'*b',x,f2,'- k',x,f3,'--r')

grid on

xlabel('x')

ylabel('y')

legend('f1=sin(x)','f2=x*sin(x)','f3=cos(x)')

title('graphes de f1,f2,f3')
```


graphes de f1,f2,f3
f1=sin(x)
f2=x*sin(x)
f3=cos(x)
y
x
2
1
0
-1
-2
-3
-4
-5
-8
-6
-4
-2
0
2
4
6
8

Chapitre VI

VI LES POLYNOMES

VI.1 Les polynômes

MATLAB représente les polynômes de degré n sous la forme d'un vecteur ligne $P=[a_n,a_{n-1},\ldots,a_0]$ de dimension (n+1), contenant les coefficients classés par puissances décroissantes. Dans l'exemple suivant, nous présentons le polynôme : $P(x)=3x^4-2x^3-5x-1$

```
>> p= [3 -2  0  -5  -1]

p=

    3  -2  0  -5  -1
```

Remarque Le nombre d'éléments du vecteur est égal au degré du polynôme + 1.

Il y a des problèmes liés aux polynômes comme :

- la recherche de racines;
- l'évaluation;
- l'adaptation à des données.

VI.1.1 Racine d'un polynôme

On peut déterminer les racines de polynôme P à l'aide de la fonction ***roots***.

Exemple

```
>> p=[1 0 -1] ;

>> roots(p)

ans =

     -1

      1
```

```
>> p=[5 2  1  6] ;
>> roots(p)
ans =
   -1.1432 + 0.0000i
    0.3716 + 0.9548i
    0.3716 - 0.9548i
```

```
>> p=[1 -4  0  5  -3] ;
roots(p)
ans =
    3.6929 + 0.0000i
   -1.2007 + 0.0000i
    0.7539 + 0.3289i
    0.7539 - 0.3289i
```

Inversement, on peut construire un polynôme à partir de ses racines en utilisant la fonction *poly*. Les racines du polynôme sont passées à la fonction sous forme d'un vecteur.

Exemple

On cherche, le polynôme qui a pour racines : -2, 1 et 4.

Celles-ci peuvent être définies comme les éléments d'un vecteur r.

```
>> r=[-2  1  4]
r =
   -2     1     4
>> p=poly(r)
p =
     1    -3    -6     8
```

Le polynôme cherché est alors : $P(x)=x^3-3x^2-6x+8$.

On vérifie bien que les racines du polynôme p sont -2 1 et 4

```
p =

   1    -3    -6     8

>> racines=roots(p)

racines =

      4.0000

     -2.0000

      1.0000
```

Remarque La fonction *poly* accepte aussi une matrice comme argument dont elle retourne le polynôme caractéristique.

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

```
>> poly(M)
ans =
   1.0000   -6.0000   -1.0000
```

VI.2 Manipulation de fonctions polynomiales

VI.2.1 Évaluation de polynômes

Pour évaluer un polynôme en un point, on utilise la fonction polyval.

Soit le polynôme $P(x)=x^3-3x^2-6x+8$.

Exemple

Essayons de trouver la valeur P(-2) , p(0).

```
>>p =[1 -3  -6  8] ;
>> polyval(p,-2)
ans=
  0
```

```
>>p =[1 -3  -6  8] ;
>> polyval(p, 0)
ans=
     8
```

VI.2.2 Dérivation et intégration d'un polynôme

a) Dérivation d'un polynôme

La fonction polyder (p) retourne la dérivée d'un polynôme p.
La commande polyder (p) retourne les coefficients de la dérivée du polynôme p dont les coefficients sont donnés dans le vecteur p.

Exemple

```
>> p1=[1  2 -3 -1];
>> polyder(p)
ans =
    3   -6   -6
```

b) Intégration d'un polynôme

La fonction polyint (p) retourne l'intégrale d'un polynôme p.
La commande polyint (p) retourne les coefficients de l'intégrale (la primitive) du polynôme avec la constante le l'intégration c=0 par défaut.

Exemple

```
>> p2=[3  2 -1];
>> polyint (p2)
ans =
    1    1   -1    0
```

VI.3 Opérations sur les polynômes

1-Addition et soustraction de polynômes

Comme la représentation d'un polynôme en Matlab se faite sous forme d'un vecteur alors l'addition (soustraction) de deux polynômes p(x)et q(x) est également l'addition (soustraction) de deux vecteurs p et q s'ils sont de même degré. Sinon il faut compléter celui au degré le plus faible avec des zéros, même chose pour l'addition (soustraction) de plusieurs polynômes.

2-Multiplication

Le résultat de la multiplication de P1 par P2 est le polynôme P3 qui s'obtient avec la fonction ***conv.***

Exemple

```
>> p1 =[1 -3  -6  ]; p2=[1  -4  2];

>> p3=conv(p1,p2)

p3 =

     1    -7     8    18   -12
```

```
>> p1 =[5  7  9  1] ;p2=[1  9  7  5] ;

>> p3=conv(p1,p2)

p3 =

   5   52  107  156  107   52    5
```

3-Division

La division de deux polynômes se fait par la fonction ***deconv.*** Le quotient **Q** et le reste **R** de la division peuvent être obtenus sous forme d'élément d'un tableau.

Exemple

```
>> p1=[1  2 -3 -1];
>> p2=[1  -1 2];
>> [Q R]=deconv (p1,p2)
Q =
    1    3
R =
    0    0   -2   -7
```

```
>> p1=[1  2 -3];
>> p2=[1  -1 ];
>> [Q R]=deconv(p1,p2)
Q =
    1    3
R =
    0    0    0
```

VI.4 Représentation graphique

Pour tracer la représentation graphique du polynôme p (x), définissons un domaine pour la variable x qui contient les racines de p.

Exemple

```
X=-2 :0.1 :2 ;

p=[3  2 -1];

y=polyval (p, x) ;

plot(x,y)

grid on

xlabel ('x')

ylabel('y')

title('y=3x²+2x-1')
```

VI.5 Série d'exercices

Exercice 01

Ecrire un script permettant de résoudre une équation du second degré : $ax^2+bx+c=0$. Ce script devra demander à l'utilisateur les valeurs de a, b et c ; puis donner le résultat.

Exercice 02

1-Ecrire des scripts qui permettent de calculer n ! (factoriel de n) en utilisant la boucle while puis la boucle for.

2-Ecrire un programme qui estime la valeur de la constante mathématique e en utilisant la formule : $e = \sum_{k=0}^{n} \frac{1}{k!}$

Exercice 03

A l'aide de trois boucles for écrire une fonction réalisant la multiplication de 2 matrices de taille 5x5. Refaire le programme en utilisant la boucle while. Donner un exemple.

Exercice 04

Soit t un vecteur contenant des valeurs comprises entre -1 et 1, avec un pas de 0.01 .Soient les fonctions f ,g et h définies par :

$f(t)= (1-t)\, e^t$, $g(t)= (t^2 -t+1)/ (t^2 +1)$, $h(t)= \sqrt{|t| + 1}$

Ecrire un script Matlab représentant les fonctions f, g et h sur la même figure.

Exercice 05

Utiliser les fonctionnalités graphiques de Matlab pour Vérifier que les 3 racines de la fonction suivante : $f(x) = x^3 - 6x^2 + 11x - 6$; sont situés dans l'intervalle [0,4].

Exercice 06

En utilisant les fonctionnalités graphiques de MATLAB, localiser la racine positive de l'équation :$f(x) = 2\sin(x) - x$
Ecrire un programme qui calcul la valeur approchée de cette racine par la méthode de dichotomie, avec une précision de 10^{-6}.

Exercice 07

Considérons l'équation non linéaire: $f(x) = x^3+4x^2-10=0$
qui admet une racine **r** dans l'intervalle [1, 2].
Voici trois façons d'écrire f(x)=0 sous la forme d'un point-fixe :

$g(x)= \frac{\sqrt{10-x^3}}{2}$, $h(x)=\sqrt{\frac{10}{x+4}}$, $k(x)=x- x^3-4x^2+10$.

1- Ecrire un programme qui calcul la valeur de r par la méthode du point fixe sur g(x), h(x), k(x), en mettant : $x_0=1,5$, $tol=10^{-3}$, Nmax=50
2-Quelle est la fonction (g, h ou k) qui donne la convergence la plus rapide.

Exercice 08

Soit la fonction $f(x) = e^{-2x}- \cos(x) - 3$
1-En utilisant les fonctionnalités graphiques de MATLAB pour vérifier que le zéro de cette fonction est situé dans l'intervalle [-1; 0]
2-Ecrire un programme qui calcul la valeur de ce zéro par la méthode de Newton on prend $x_0= 0$ comme point initial.

BIBLIOGRAPHIE

1-Freddy Mudry, Matlab pour les ingénieurs, quelques exemples ,2010

2-Nadia Martaj & Mohand Mokhtari « MATLAB R2009, SIMULINK et STATEFLOW pour Ingenieurs, Chercheurs et Etudiants, Springer-Verlag Berlin Heidelberg 2010.

3-Stéphane Balac « Débuter avec MATLAB » INSA de Lyon 2001.

4-Abhishek K Gupta, Numerical Methods using MATLAB, 2014