1. Introduction :

Pour évaluer des expressions en les écrivant dans la ligne de commande (Après le prompt >>), par conséquent les commandes utilisées s'écrivent généralement sous forme d'une seule instruction éventuellement sur une seule ligne.

Cependant, il existe des problèmes dont la description de leurs solutions nécessite **plusieurs instructions**, ce qui réclame l'utilisation de **plusieurs lignes**.

Une collection d'instructions bien structurées visant à résoudre un problème donné s'appelle un **programme**.

2. La programmation Matlab

2.1 Les commentaires

Les commentaires sont des phrases explicatives ignorées par MATLAB et destinées pour l'utilisateur afin de l'aider à comprendre la partie du code commentée.

En MATLAB, un commentaire commence par le symbole % et occupe le reste de la ligne.

Exemple:

```
>> A=B+C ;          % Donner à A la valeur de B+C
```

2.2 Écriture des expressions longues

Si l'écriture d'une expression longue ne peut pas être enclavée dans une seule ligne, il est possible de la diviser en plusieurs lignes en mettant à la fin de chaque ligne au moins trois points.

Exemple :

```
>> (sin(pi/3)^2/cos(pi/3)^2)-(1-2*(5+sqrt(x)^5/(-2*x^3-x^2)^1+3*x)) ;
```

Cette expression peut être réécrite de la façon suivante :

```
>> (sin(pi/3)^2/cos(pi/3)^2)- ...  ↵
>> (1-2*(5+sqrt(x)^5 .....            ↵
>> /(-2*x^3-x^2)^1+3*x)) ;            ↵
```

2.3 Entrées/sorties

Pour le développement d'un programme interactif, la sortie écran et l'entrée clavier se font respectivement a l'aide des fonctions disp et input.

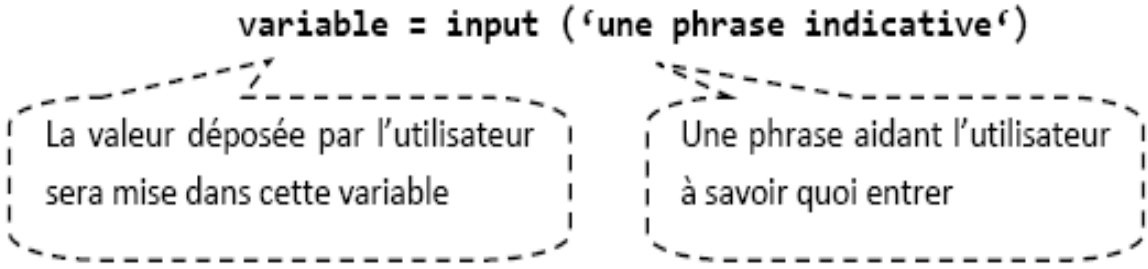
La sauvegarde et le chargement des variables se font respectivement à l'aide des fonctions save et load.

Pour une gestion plus souple de la lecture et de l'écriture d'un fichier, on peut être amène a utiliser les fonctions fopen, fclose, fprintf, fscanf, fgetl et fgets.

2.4 Lecture des données dans un programme (Les entrées)

La fonction input permet la saisie d'une valeur depuis le clavier.

- Pour les valeurs numériques, *n* = input('*message*') affiche *message* et affecte à la variable *n* la valeur numérique entrée au clavier.



2.5 Lecture des données dans un programme (Les entrées)

Quand MATLAB exécute une telle instruction, la phrase indicative sera affichée à l'utilisateur en attendant que ce dernier entre une valeur.

```
>> A = input ('Entrez un nombre entier : ')          ↵
   Entrez un nombre entier : 5                       ↵
   A =
      5
>>

>> A = input ('Entrez un nombre entier : ');         ↵
   Entrez un nombre entier : 5                       ↵
>>

>> B = input ('Entrez un vecteur ligne : ')          ↵
   Entrez un vecteur ligne : [1:2:8,3:-1:0]          ↵
   B =
      1     3     5     7     3     2     1     0
```

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

Exemple

>> nom = input('Entrez votre nom ','s')

Entrez votre nom Mohammed

nom =

Mohammed

- Il existe des cas où on désire afficher uniquement la valeur de la variable (sans le nom et sans le signe =). Pour cela, on peut utiliser la fonction **disp,**

la syntaxe : disp (objet)

La valeur de l'objet peut être un **nombre**, un **vecteur**, une **matrice**, une **chaine de caractères** ou une **expression**.

On signale qu'avec un vecteur ou une matrice vide, **disp n'affiche rien.**

```
>> disp(A)                    % Afficher la valeur de A sans 'A = '
     5
>> disp(A);                   % Le point virgule n'a pas d'effet
     5
>> B                          % Afficher le vecteur B par la méthode classique
   B =
      1     3     5     7     3     2     1     0
>> disp(B)                    % Afficher le vecteur B sans 'B = '
      1     3     5     7     3     2     1     0
>> C = 3 :1 :0                % Création d'un vecteur C vide
   C =
   Empty matrix: 1-by-0
>> disp(C)                    % disp n'affiche rien si le vecteur est vide
>>
```

2.6 Les expressions logiques

2.6.1 Les opérations logiques

L'opération de comparaison	Sa signification
==	l'égalité
~=	l'inégalité
>	supérieur à
<	inferieur à
>=	supérieur ou égale à
<=	inferieur ou égale à

L'opération logique	Sa signification
&	le **et** logique
|	le **ou** logique
~	la **négation** logique

Les opérations logiques : Une variable logique peut prendre les valeurs 1 (vrai) ou 0 (faux) avec une petite règle qui admette que :

- Toute valeur égale à 0 sera considérée comme fausse (= 0 ⇒ Faux)
- Toute valeur différente de 0 sera considérée comme vrai (≠0 ⇒ Vrai).

Le tableau suivant résume le fonctionnement des opérations logiques :

a	b	a & b	a | b	~a
1 (vrai)	1 (vrai)	1	1	0
1 (vrai)	0 (faux)	0	1	0
0 (faux)	1 (vrai)	0	1	1
0 (faux)	0 (faux)	0	0	1

Exemple :

```
>> x=10;
>> y=20;
>> x < y                          % affiche 1 (vrai)
      ans =
            1
>> x <= 10                        % affiche 1 (vrai)
      ans =
            1
>> x == y                         % affiche 0 (faux)
      ans =
            0
>> (0 < x) & (y < 30)             % affiche 1 (vrai)
      ans =
            1
>> (x > 10) | (y > 100)           % affiche 0 (faux)
      ans =
            0
>> ~(x > 10)                      % affiche 1 (vrai)
      ans =
            1
```

```
>> 10 & 1                    % 10 est considéré comme vrai donc 1 & 1 = 1
      ans =
            1
>> 10 & 0                    % 1 & 0 = 1
      ans =
            0
```

3. Programme Matlab

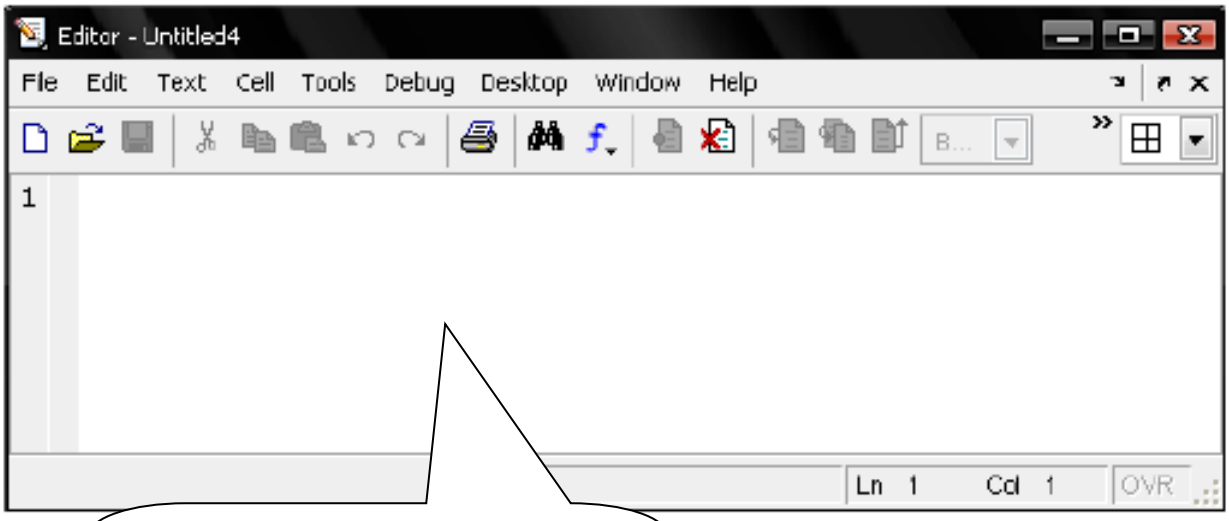
Ecrire le programme dans un **fichier séparé**, et d'appeler ce programme (au besoin) en tapant le nom du fichier dans l'invité de commande.

Cette approche est définie en MATLAB par les **M-Files,** qui sont des fichiers pouvant contenir les données, les programmes (scripts) ou les fonctions que nous développons.

Pour créer un M-Files il suffit de taper la commande **edit,** ou tout simplement aller dans le menu :

File → New → M-Files (ou cliquer sur l'icône).

Dans tous les cas une fenêtre d'édition comme celle-ci va apparaitre :



Ecrire votre programme dans cette fenêtre, puis l'enregistrer avec un nom (par exemple : '**tp.m').**

- Pour l'exécution du programme:

  Aller à l'invité de commande habituel (>>) puis taper le nom de **notre fichier (sans le '.m')** comme ceci :

  >> tp

  Et l'exécution du programme va démarrer immédiatement

Pour retourner à la fenêtre d'édition (après l'avoir fermer) il suffit de saisir la commande :

>> edit tp

4. Structures de contrôle de flux

Les structures de contrôle de flux sont des instructions permettant de définir et de manipuler l'ordre d'exécution des tâches dans un programme.

MATLAB compte huit structures de contrôle de flux à savoir :

- **If**
- **switch**
- **for**
- **While**
- **break**
- **try - catch**
- **return**
- **continue**

4.1 L'instruction if

```
if (condition)
        instruction_1
        instruction_2
            . . .
        Instruction_N
end
```

ou bien

```
if (condition)
        ensemble d'instructions 1
else
        ensemble d'instructions 2
end
```

```
if (expression_1)
        Ensemble d'instructions 1
elseif (expression_2)
        Ensemble d'instructions 2
              ....
elseif (expression_n)
        Ensemble d'instructions n
else
        Ensemble d'instructions si toutes les expressions étaient fausses
end
```

Exemple: programme qui trouve les racines d'une équation de second degré désigné par : $ax^2+bx+c=0$.

Voici le M-File qui contient le programme qui est enregistré avec le nom : exemple.m

```
% Programme de résolution de l'équation a*x^2+b*x+c=0
a = input ('Entrez la valeur de a : ');          % lire a
b = input ('Entrez la valeur de b : ');          % lire b
c = input ('Entrez la valeur de c : ');          % lire c
delta = b^2-4*a*c ;                              % Calculer delta
if delta<0
        disp('Pas de solution')                  % Pas de solution
elseif delta==0
        disp('Solution double : ')               % Solution double
        x=-b/(2*a)
else
        disp('Deux solutions distinctes: ')      % Deux solutions
       x1=(-b+sqrt(delta))/(2*a)
       x2=(-b-sqrt(delta))/(2*a)
end
```

Pour l'exécution du programme, il suffit de taper le nom du programme :

>> exemple

Entrez la valeur de a : -3

Entrez la valeur de b : 2

Entrez la valeur de c : 1

'Deux solutions distinctes :

x1 =-1/3

x2 =1

Remarque :

Il existe la fonction solve prédéfinie en MATLAB pour trouver les racines d'une équation.

>> solve('3*x^2+4*x+2=0','x')

ans =

-2/6

1

4.2 L'instruction switch

```
switch (expression)
      case valeur_1
            Groupe d'instructions 1
      case valeur_2
            Groupe d'instructions 2
            . . .
      case valeur_n
            Groupe d'instructions n
      otherwise
            Groupe d'instructions si tous les case ont échoué
end
```

Exemple:

```
x = input ('Entrez un nombre : ') ;
switch(x)
case 0
            disp('x = 0 ')
case 10
            disp('x = 10 ')
case 100
            disp('x = 100 ')
otherwise
            disp('x n''est pas 0 ou 10 ou 100 ')
end
```

L'exécution donne :

```
Entrez un nombre : 30
x n'est pas 0 ou 10 ou 100
```

4.3 L'instruction for

```
for variable = expression_vecteur
      Groupe d'instructions
end
```

L'expression_vecteur corresponde à la définition d'un vecteur : début : pas : fin ou début : fin

La variable va parcourir tous les éléments du vecteur et pour chacun, il va exécuter le groupe d'instructions.

Exemple :

L'instruction for	`for i = 1 : 4`
	`  j=i*2 ;`
	`  disp(j)`
	`end`
		`for i = 1 : 2 : 4`
		`  j=i*2 ;`
		`  disp(j)`
		`end`
			`for i = [1,4,7]`
			`  j=i*2 ;`
			`  disp(j)`
			`end`

Le résultat de l'exécution	2 4 6 8	2 6	2 8 14

4.4 L'instruction while

```
while (condition)
        Ensemble d'instructions
end
```

Tant que l'expression de **while** est évaluée à true, l'ensemble d'instructions s'exécutera en boucle.

Exemple :

a=1 ;

while (a~=0)

a = input ('Entrez un nombre (0 pour terminer) : ') ;

end

4.5 L'instruction continue, Break

- [break] pour sortir immédiatement de la boucle,
- [continue] pour passer directement à l'itération suivante de la boucle.

```
a = 10;
while a < 20
  if a == 15
    a = a + 1;
    continue;
  end
fprintf('value of a: %d\n', a);
a = a + 1;
end
```

Exercices:

Il existe des fonctions prédéfinis en MATLAB.

Essayons de les programmer (pour un vecteur donnée V).

- La somme des éléments d'un vecteur
- Le produit des éléments d'un vecteur
- La moyenne des éléments d'un vecteur
- Créer une matrice M ayant le vecteur V dans le diagonal, et 0 ailleurs
- Ordonne les éléments du vecteur V par ordre croissant

La fonction	Description	Le programme qui la simule

sum (V)	La somme des éléments d'un vecteur V	`n = length(V);`
		`somme = 0 ;`
		`for i = 1 : n`
		`  somme=somme+V(i) ;`
		`end`
		`disp(somme)`

prod (V)	Le produit des éléments d'un vecteur V	`n = length(V);`
		`produit = 1 ;`
		`for i = 1 : n`
		`  produit=produit*V(i) ;`
		`end`
		`disp(produit)`

mean (V)	La moyenne des éléments d'un vecteur V	`n = length(V);`
		`moyenne = 0 ;`
		`for i = 1 : n`
		`  moyenne = moyenne+V(i) ;`
		`end`
		`moyenne = moyenne / n`

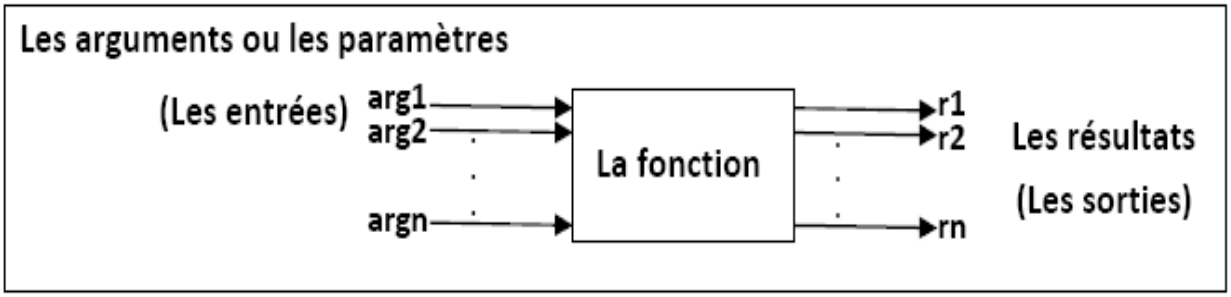
diag (V)	Créer une matrice ayant le vecteur V dans le diagonale, et 0 ailleurs	`n = length(V);`
		`A = zeros(n) ;`
		`for i = 1 : n`
		`  A(i,i)=V(i) ;`
		`end`
		`disp(A)`

sort (V)	Ordonne les éléments du vecteur V par ordre croissant	`n = length(V);`
		`for i = 1 : n-1`
		`  for j = i+1 : n`
		`  if V(i) > V(j)`
		`  tmp = V(i) ;`
		`  V(i) = V(j) ;`
		`  V(j) = tmp ;`
		`  end`
		`  end`
		`end`
		`disp(V)`

5. Les fonctions

5.1 Fonctions en informatique ou en mathématique

En informatique, une fonction est une routine (un sous programme) qui accepte des arguments (des paramètres) et qui renvoie un résultat.



En mathématique une fonction **f est une relation qui attribue à chaque valeur x au plus une seule valeur f(x).**

5.2 Création d'une fonction dans un M-Files

MATLAB contient un grand nombre de fonctions prédéfinies comme sin, cos, sqrt, sum, … etc.

Il est possible de créer nos propres fonctions en écrivant leurs codes source dans des fichiers M-Files (portant le même nom de fonction) en respectant la syntaxe suivante :

```
function [r1, r2, …, rn] = nom_fonction (arg1, arg2, …, argn)

      % le corps de la fonction
      . . .
      r1 = . . .  % la valeur retournée pour r1
      r2 = . . .  % la valeur retournée pour
      r2 . . .
      rn = . . .  % la valeur retournée pour rn
end               % le end est facultatif
```

Où: $r_1 ... r_n$ sont les valeurs retournées, et $arg_1 ... arg_n$ sont les arguments.

Exemple : écrire une fonction qui calcule la racine carrée d'un nombre par la méthode de Newton.

function r = racine(nombre)

r = nombre/2; precision = 6;

for i = 1: precision

r = (r + nombre ./ r) / 2;

end

Exécution :

>> x = racine(9)

x =

3

>> x = racine(196)

x =

14.0000

>> x = racine([16,144,9,5])

x =

4.0000 12.0000 3.0000 2.2361

Remarque :

Contrairement à un programme (un script), une fonction peut être utilisée dans une expression par exemple : **2*racine(9)-1.**

Un programme	Une fonction

`a = input('Entrez un nombre positif: ');`
`x = a/2;`
`precision = 6;`
`for i = 1:precision`
`  x = (x + a ./ x) / 2;`
`end`
`disp(x)`
	`function r = racine(nombre)`
	`  r = nombre/2;`
	`  precision = 6;`
	`  for i = 1:precision`
	`  r = (r + nombre ./ r) / 2;`
	`  end`

L'exécution :
`>> racine` ↵
`Entrez un nombre positif: 16` ↵
`4`
	L'exécution :
	`>> racine(16)` ↵
	`ans =`
	`4`

on ne peut pas écrire des expressions tel que : `>> 2 * racine + 4` ✗	on peut écrire sans problème des expressions comme : `>> 2 * racine(x) + 4` ✓

5.3 La récursivité

Dans la définition d'une fonction f, il arrive que l'on désire rappeler la même fonction f pour des valeurs différentes. L'exemple typique étant le cas de la définition de la fonction factorielle sur les entiers.

$$\text{factoriel}(n) = \begin{cases} 1 & \text{si } n = 0 \\ n * \text{factoriel}(n-1) & \text{si } n > 0 \end{cases}$$

Ceci est possible en MatLab, comme dans bien des langages de programmation, mais il faut être attentif au fait que cet outil amène souvent des boucles sans fin, si la condition d'arrêt n'est pas bien pensée. Voilà l'exemple de la factorielle programmée en MatLab.

```
function nn =factoriel(n)
%calcule le n factoriel en fonction de n.
reponse=1;
if n==0
nn=1;
else
reponse= n*factoriel(n-1);
end
nn=reponse;
```

5.4 La fonction Inline

Lorsque le corps de la fonction se résume à une expression relativement simple, on peut créer la fonction directement dans l'espace de travail courant, sans utiliser un *m-file* auxiliaire.

Syntaxe : La syntaxe des fonctions Inline est simple :

***nom-de-fonction* = inline ('*expression*', '*var1*', '*var2*', ...)**

Exemple:

```
>> f = inline('x.^2 + x.*y', 'x', 'y')

f =

Inline fonction :

f(x, y) = x.^2 + x.*y

>> f(1, 2)

ans =

3.0000
```

L'exemple suivant met en évidence le mécanisme de déclaration implicite.

Exemple

La définition

```
>> f = inline('x.^2')
```

f =

Inline fonction :

f(x) = x.^2

est équivalente à :

>> f = inline('x.^2', 'x')

Ce mécanisme est ambigu dès qu'il y a plusieurs variables

La définition

>> f = inline('x.^2 + x.*y')

est équivalente à :

>> f = inline('x.^2 + x.*y', 'x', 'y')

alors que

>> f = inline('x.^2 + x.*t')

est équivalente à :

>> f = inline('x.^2 + x.*t', 't', 'x')

5.5 Fonctions anonymes

Ce mode de définition de fonctions utilise comme pour les fonctions Inline l'espace de travail courant. La syntaxe minimale est peu explicite, mais les fonctions ainsi définies seraient **plus efficaces** que les fonctions Inline.

Syntaxe : *nom-de-fonction* = @(*var1*, *var2*, ...) *expression*

Contrairement aux fonctions Inline l'expression mathématique qui constitue le corps de la fonction ainsi que les variables **ne doivent pas** être tapées entre **apostrophes**.

• Il est préférable que les fonctions soient *vectorisées* comme le sont les fonctions prédéfinies

Exemple :

>> g = @(x, y) x.^2 + x.*y

g =

@(x, y) x.^2 + x.*y

>> g(1, 2)

ans =
3.0000

5.6 Fonctions argument d'autres fonctions

Une façon simple d'utiliser une fonction comme argument d'une autre fonction est de transmettre à la fonction utilisatrice le ***handle*** de la fonction (**le *handle* d'une fonction est la référence (l'adresse en mémoire) du code MATLAB qui définit le traitement effectué par la fonction**).

Le *handle* d'une *m-fonction* ou d'une fonction prédéfinie de MATLAB est obtenu en faisant **précéder** le **nom de la fonction** par le symbole **@**. Par exemple, le *handle* de la fonction prédéfinie exp est **@exp**.

Le *handle* d'une fonction Inline ou d'une fonction anonyme est le nom avec lequel la fonction a été définie.

Exemple:

```
function y = trapèze(f, a, b)

% Input : f : handle

% a : borne inférieure

% b : borne supérieure

% Output : aire du trapèzes

y = 0.5*(f(a) + f(b))*(b - a) ;

return
```

```
>> carre = inline('x.^2', 'x')

carre =

Inline fonction :

carre(x) = x.^2

>> trapeze(carre, 1, 2)

ans =

2.5000
```

On peut également définir la fonction trapèze comme une fonction anonyme :

Exemple :

```
>> trapeze2 = @(f, a, b) 0.5*(f(a) + f(b))*(b - a) ;
```

>> carre = @(t) t.^2 ;

>> trapeze2(carre, 1, 2)

ans =

2.5000

5.7 La fonction menu

La fonction menu génère une fenêtre contenant un menu dans lequel l'utilisateur doit choisir une option :

result = menu('*titre*', '*opt1*', '*opt2*', ..., '*optn*')

- La valeur retournée dans la variable *result* est égale au numéro de l'option choisie ;

-La fonction menu est souvent utilisée en relation avec la structure algorithmique switch-case

Exemple

result = menu('Traitement', 'Gauss', 'Gauss-Jordan', 'Quitter')

Si l'utilisateur sélectionne *Gauss*, la variable result prend la valeur 1, la valeur 2 s'il sélectionne *Gauss-Jordan* et la valeur 3 s'il sélectionne *Quitter*.

5.8 La fonction num2str()

La fonction num2str(*x*) où *x* est un nombre, retourne la valeur littérale de ce nombre.

Exemple:

>> s = ['la valeur de pi est : ' num2str(pi)] ;

>> s

s =

la valeur de pi est : 3.1416

La commande disp sera souvent utilisée avec num2str pour afficher les valeurs des expressions numériques.

La fonction num2str(*x*) où *x* est un nombre, retourne la valeur littérale de ce nombre.

Exemple:

>> s = [’la valeur de pi est : ’ num2str(pi)] ;

>> s

s =

la valeur de pi est : 3.1416

La commande disp sera souvent utilisée avec num2str pour afficher les valeurs des expressions numériques.

Exemple :

>> a = [1 2;3 4] ;

>> disp(a)

1 2

3 4

>> disp([’ordre de la matrice a : ’ num2str(size(a,1))]);

ordre de la matrice a : 2

……..Suite partie 3

Références :

[1] Cours Outils de programmation pour les mathématiques, Y.BAZIZ, Centre Universitaire RELIZANE, Institut des Sciences et Technologies, Spécialité : Mathématique et Informatique, LMD 1ére année 2013/2014.
[2] Tutoriel MATLAB, Introduction aux Méthodes Numériques 2013-2014.
[3] poly introduction Matlab, Introduction à Matlab
[4] Introduction à MATLAB, André Casadevall, Université Paris-Dauphine, Département MIDO, mars 2013
[5] Outils Mathématiques et utilisation de Matlab, Quentin Glorieux , Cours 2013-2014, Université Pierre et Marie Curie - Paris VI, Licence Professionnelle (L3) Instrumentation Optique et Visualisation,.