Présentation de Matlab

1. Introduction: MATLAB est une abréviation de **MAT**rix **LAB**oratory. Ecrit à l'origine en Fortran par Cleve Moler, MATLAB était destiné à faciliter l'accès au logiciel matriciel développé dans les projets LINPACK et EISPACK. La version actuelle est écrie en langage C par The Matworks Inc..

MATLAB est un environnement complet, puissant et facile à utiliser, destiné au calcul scientifique. Il dispose de plusieurs centaines (voire milliers) de fonctions mathématiques, scientifiques et techniques. Dans MATLAB, l'élément de base est la matrice. l'utilisateur ne s'occupe pas des allocations mémoires ou de redimensionnement comme dans les langages classiques. MATLAB comprend aussi un ensembles de boites à outils (Toolboxes) spécifiques à des domaines variés: le traitement du signal (Signal Processing Toolbox), l'automatique (Control system Toolbox), l'indentification des systèmes, les réseaux de neurones, la logique floue, les ondelettes, l'apprentissage automatique (Machine Learning),... etc.
En complément à MATLAB, l'outil additionnel "Simulink" est proposé pour la modélisation et la simulation de systèmes dynamiques utilisant une représentation de type *Schéma-blocs*.

2. Démarrage de MATLAB

Pour lancer l'exécution de MATLAB sous Windows, il faut cliquer sur *Démarrage*, ensuite *Programme*, ensuite *MATLAB*. L'invite ">>" de MATLAB doit alors apparaître à la suite duquel on entrera les commandes. La fonction "quit" ou "exit" permet de quitter MATLAB.

```
>> quit
```

3. Types de données

Dans Matlab, il y a un seul type de données: le type matrice (Matrix).

4. Notions de base de Matlab

Variables spéciales et constantes

Dans MATLAB, on trouve:
Des constantes prés-définies
pi: 3.14159265358979
eps: 2.2204e-16 ($2.2204\ 10^{-16}$)
Inf (infinite): nombre infini
NaN (Not a number): n'est pas un nombre, exprime parfois une indétermination.
Une variable pré-définie
ans: variable contenant la dernière réponse.
i et j: unités imaginaires (i^2=-1)

Exemples

```
>>pi
ans= 3.1416
>>eps
ans=2.2204e-016
>>1/0
warning: Divide by zero
ans=Inf
>>0/0
warning: Divide by zero
ans= NaN
>>i   % ou >>j
ans=0+1.0000i
```

4.1 Opérations arithmétiques (+,-,*,/)

Les opérations arithmétiques de base dans MATLAB sont: + pour l'addition, - pour la soustraction, * pour la multiplication et / ou \ pour la division.

```
>>5+7-3+2*5
ans=19
>>4/3  % c'est 4 qui est divisé par 3
>>4\3  %c'est 3 qui est divisé par 4
ans=0.7500
```

L'opérateur "^": (Touches: **Alt Gr** et **9**)
Pour l'élévation à une puissance entière ou réelle.

```
>>3^4
ans=81
```

Remarque: l'élévation à une puissance est **prioritaire par rapport aux autres opérations**; l'utilisation des parenthèses est recommandée.

```
>>9^1/2   % est équivalent à (9^1)/2
ans=4.5000
>>9^(1/2)
ans=3
```

4.2 Priorité des opérateurs

La priorité des opérateurs est come suit:
( ) ; ^ ; *,/ ; +,-
La multiplication et la division ont la même priorité. De même, pour l'addition et la soustraction. S'il y a des opérateurs avec la même priorité, les opérandes sont évalués de gauche à droite.
Les opérations peuvent être enchaînées en respectant les priorités usuelles des opérations et en utilisant des parenthèses.

```
>>4*(-5)+12
```

```
ans=-8
>>2.3*(4-6)/(3+15)   %ou (2.3*(4-6))/(3+15)
ans=-0.2556
```

Le symbole "%": tout ce qui se trouve après le symbole **%** sera considéré comme un commentaire. Il sera donc ignoré lors de l'exécution du script.
un commentaire peut servir à apporter des informations permettant une meilleure compréhension des lignes de commandes.

```
>>a=2 % la valeur initiale de a est 2
a=2
>>a=1;b=2;c=3;
>> a+b-2*c  %est équivalente à: (a+b)-(2*c)
ans= -3
>>y=2*a/(b*c) %est équivalente à: (2*a)/(b*c)
y= 0.3333
```

4.3 Les formats

Dans Matlab, il est possible d'afficher les valeurs dans différentes format: flottants courts (short), flottants longs (long), flottants courts en notation scientifique (short e), flottants longs en notation scientifique (long e), ainsi que les formats hexadécimal et rationnel (notation sous une forme d'une fraction).

Format flottant short

```
>> format short   % le format par défaut
>>22/7
ans=3.1429  (4 chiffres après la virgules)
```

Format flottant long

```
>> format long  %(15 chiffres après la virgule)
>>22/7
ans=3.14285714285714
```

Format flottant short en notation scientifique

```
>> format short e
>>22/7
ans=3.1429e+000
```

Format flottant long en notation scientifique

```
>> format long e
>>22/7
ans=3.142857142857143e+003
```

Format rationnel (fraction)

```
>>format rat
>>0.36
ans=9/25
>>format bank % 2 chiffres après la virgule
>>22/7
ans=3.14
```

Pour plus de détail, tapez: >> help format ou (>>doc format)

5. Vecteurs ou tableaux à 1 dimension

Saisie d'un tableau x, vecteur ligne

On sépare les éléments du vecteur ligne par des blancs ou par des virgules.

```
>>x=[3 2 5]  % ou x=[3,2,5]
x=3 2 5
```

***La fonction *size*:** donne la taille du vecteur.

```
>>size(x)
ans=1  3
[m, n]=size(x) % ou [m n]=size(x)
m=1
n=3
```

m et n sont respectivement le nombre de lignes et de colonnes

```
m=size(x,1)  % nombre de lignes
m=1
n=size(x,2) % nombre de colonnes
n=3
>>z1=[1;2;3] % retourne un vecteur colonne
z1=1
    2
    3
>>z2=[1;2;3]' % retourne un vecteur ligne

z2=1 2 3
```

***La fonction *length*:** détermine la longueur du tableau (ou de la matrice) qui est sa plus grande dimension.

```
longueur_x=length(x)
longueur_x=3
```

5.1 Construction d'un tableau à partir d'un autre

On veut créer v=[3 2 5 4 6 9] à partir du vecteur x précédent.

```
>>v=[x 4 6 9]
v=3 2 5 4 6 9
```

De même, pour avoir w=[1 3 2 5 4 6 9] on tape:

```
>>w=[1 v]
w=1 3 2 5 4 6 9
>>w=[1 x 4 6 9]
w=1 3 2 5 4 6 9
```

Les indices: Dans MATLAB, les indices d'un tableau **commencent à 1**. Pour récupérer une composante d'un vecteur, il faut spécifier son indice entre parenthèses.

```
>>w(3)
ans=4
```

Si les composantes d'un vecteur sont espacées d'un **pas constant** et si la première et la dernière valeur sont connues, alors ce vecteur peut être écrit de la manière suivante:

```
>>debut=0;fin=1;pas=0.25;
```

>>**t=debut:pas:fin %**ou t= **[debut:pas:fin]**

% t= **(debut:pas:fin)**

```
t=0     0.2555    0.5000     0.7500      1.0000
```

5.2 Addition et soustraction des vecteurs

Se font élément par élément.

```
>>x=[0 4 3]; y=[2 5 7]
>>x-y
ans= -2    -1    -4
>>x+y
ans= 2      9     10
```

5.3 Ajout ou retrait d'un scalaire à un vecteur revient à ajouter ou retrancher ce scalaire à toutes les composantes du vecteur.

```
>>3+x
ans=
  3  7  6
>>x-2
ans=
-2    2   1
>>2*x
ans=
0   8   6
>>x/4
ans=
  0      1.0000         0.7500
```

L'opérateur de transposition " ' ": (transposé + conjugué) permet de transformer un vecteur ligne en un vecteur colonne et inversement, avec le conjugué des nombres complexes.

L'opérateur de transposition " .' ": transposé.

A.'=A' si A est réel.

```
>>tx=x'
tx=0
     4
     3
```

Le produit d'un vecteur colonne de taille ***n*** par un vecteur ligne de taille ***m*** donne une matrice de dimension ***(n,m)***.

```
>>tx*y
ans=
0    0     0
8    20    28
6    15    21
```

Le produit d'un vecteur par sa transposé donne le carré de la norme de celui-ci.

```
>>x*tx
ans=
      25
```

L'opérateur ".": en précédant d'un point les opérateurs *, /, \ et ^, on **réalise des opérations élément par élément**.

```
>>x.*y
ans=
      0   20    21
>>x.^2
ans=
       0     16     9
>>x./y
ans=
      0    0.80000      0.4286
>>y.\x
ans=
      0      0.80000       0.4286
```

5.4 Fonctions mathémétiques dans Matlab

sin(x): sinus de x
cos(x): cosinus de x
tan(x): tangente de x
asin(x): sinus inverse (arcsinus) de x
acos(x): cosinus inverse (arccos) de x
atan(x): tangente inverse (arctan)
cosh: cosinus hyperbolique de x
sinh: sinus hyperbolique de x
abs(x): valeur absolue de x.
angle(x): valeur complexe de l'angle de phase.
sqrt(x): racine carré de x
real(x): partie réelle de la valeur complexe de x
imag(x): partie réelle de la valeur complexe de x
conj(x): complexe conjugué de x
sum(x): somme des composantes de x
prod(x): produit des composantes de x
mean(x): moyenne des composantes de x
rem(x,y): le reste de la divsion
exp(x): exponentielle
log(x): logarithme népérien
log10(x): logarithme de base 10

Exemples

```
>>x=[ 0 4 3]
sum(x)
ans=
        7
>>y=[2 5 7 ]
>>prod(y)
ans=
       70
sqrt(x)
ans=
       0     2.0000     1.7321
>>mean(x)
ans=
       2.3333
>>max(x)     % Valeur maximale du vecteur x
ans=
7
>>min(x)  % Valeur minimale du vecteur x
ans=
       0
```

6. Matrices ou tableaux à 2 dimensions

Saisie d'une matrice

```
>>x=[0 1 2;3 4 5]
x=
    0  1  2
    3  4  5
```

Les lignes de la matrice peuvent être séparées par un retour à la ligne:

```
>>x=[0  1  2
       3  4  5]
x=
    0  1  2
    3  4  5
```

Dimensions de la matrice x

```
>>[m,n]=size(x)
m=2
n=3
```

Accès à un élément de la matrice

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

6.1 Matrices particulières

Dans MATLAB, on trouve certaines matrices spéciales ou particulières, nous en donnons ci-après quelques exemples (voir **tableau 1**)

Matrice identité: **eye(m,n)** ou **eye(n)**

```
>>identite=eye(3) % ou identite=eye(3,3)
identite=
          1     0     0
          0     1     0
          0     0     1
```

Matrice nulle: **zeros(m,n)** ou **zeros(n)**

```
>>m_zero=zeros(2,3)
m_zero=
          0     0     0
          0     0     0
```

Matrice unité: **ones(m,n)** ou **ones(n)**

```
>>m_un=ones(3,2)
m_un=
          1     1
          1     1
          1     1
```

Matrice aléatoire: **rand(m,n)** ou **rand(m)**

Génère une matrice aléatoire de ligne l et de colonne c ayant des valeurs aléatoire entre 0 et 1.

```
>>m_alea=rand(3,2)
m_alea=
          0.0470     0.9347
          0.2190     0.6793
          0.6789     0.3835
```

Remarque: Pour une matrice carrée unité, nulle ou aléatoire, il suffit d'indiquer l'ordre comme seul argument des fonctions ones, zeros et rand.

6.2 Opérations élémentaires sur les matrices

Soient les matrices carrées x et y suivantes:

```
>>x=[1  2  3;0  3  1;-2  -1  4];
>>y=[2  -1  -3;1  -1  3;4  2  3];
```

La somme de 2 matrices: se note x+y

```
>>z=x+y
z=
    3     1     0
    1     2     4
    2     1     7
```

Le produit des 2 matrices: se note x*y

```
>>p=x*y
p=
    16     3    12
    7     -1    12
    11    11    15
```

Remarque: si x est une matrice**(n, m)** et y est une matrice**(p, q)**, le produit x*y ne peut se faire que si **m=p**; le résultat est une matrice **(n, q)**.

Elévation à une puissance d'une matrice

```
>> x^2
ans=
     -5     5    17
     -2     8     7
     -10  -11   9
```

L'opérateur ".": en précédant un opérateur par un point, celui-ci s'applique **à tous les éléments de la matrice**.

```
>>x.^2
```

```
ans=
     1     4     9
     0     9     1
     4     1    16
```

Remarque: Les fonctions sum, prod, mean, précédemment rencontrées s'appliquent aux colonnes de la matrice.

```
>>som_x=sum(x)
som_x=
          -1     4     8
>>moy_x=mean(x)
   moy_x=
              -1      1.3333      2.6667
```

Déterminant d'une matrice

```
>>det(x)
ans =
       27
```

Inverse d'une matrice carrée

```
>>inv(x)
ans =
       0.4815     -0.4074     -0.2593
      -0.0741      0.3704     -0.0370
       0.22222    -0.1111      0.1111
```

Extraction d'une partie d'une matrice

A partir d'une matrice, on peut extraire une autre matrice, un vecteur ou l'un de ses éléments.

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

L'élément de la 2ième ligne et de la 3ième colonne peut être récupéré en écrivant:

```
>>x(2,3)
ans=
      6
```

L'instruction suivante permet de récupérer une partie de la matrice qui est composée de toute les lignes de x (représentées par le signe :) et des colonnes 2 à 3.

```
>>x1=x(:,2:3)  % ou x1=(:,[2:3]) %ou ou x1=(:,[2,3])
X1=
     2     3
     5     6
     8     9
```

Cette partie peut être aussi obtenue en supprimant, de la matrice x, la 1ière colonne.

Remplacement de la 1ière colonne par une colonne vide

```
>>x(:,1)=[ ]
x=
   2     3
   5     6
   8     9
```

Extraction de la 2ième colonne

```
>>x=[1  2  3;4   5   6;7   8   9]
x=
    1    2    3
    4    5    6
    7    8    9
>>x(:,2) %Afficher la 2ième colonne, toutes les lignes
ans=  2
      5
      8
```

Extraction de la 2ième ligne

```
>>x(2,:) %Afficher la 2ième ligne, toutes les colonnes
ans=   4    5    6

>>a(:,end) %Afficher la dernière colonne
ans= 3
     6
     9

% Afficher juste les éléments x(1,2) et x(3,2): les lignes 1 et 3 de la colonnes 2
>>x([1,3],2)
ans=  2
      8

% Afficher juste les éléments x(2,2) et x(2,3): les colonnes 2 et 3 de la ligne 2
>>x(2,[2,3])
ans= 5    6

% Afficher  les lignes 2 et 3 (toutes les colonnes)
>>x(2:3,:)
ans= 4  5   6
     7  8   9

% Afficher  les lignes 1 et 2 et les colonnes 2 et 3
>>x(1:2,2:3)%ou x([1:2],[2:3]) %ou x([1,2],[2,3])
ans=2   3
    5   6
% Afficher  les lignes 2 et 3 et les colonnes 1 et 2

>>x(2:3,1:2) )%ou x([2:3],[1:2]) )%ou x([2,3],[1,2])
ans= 4   5
     7   8

% Afficher  les lignes 1 et 3 et les colonnes 1 et 3
>>x([1,3],[1,3])
ans= 1   3
     7   9

>>x(1:3,1:3) %ou x([1:3],[1:3]) %ou x([1,2,3],[1,2,3])
ans=
     1   2   3
     4   5   6
     7   8   9
```

Cette instruction est équivalente à: >>x

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

>>x(1:2:3,1:2:3) %ou x([1:2:3],[1:2:3])ou x([1,3],[1,3])
ans= 1   3
     7   9
```

7. Quelques instructions utiles

La commande ***who*** permet de lister les variables utilisées alors que ***whos*** donne des informations détaillées sur toutes les variables.

Exemple

```
>>t=[1 2 3];x=1;z=2-3i;

>> who
Your variables are:
t  x  z
>> whos
  Name      Size          Bytes  Class     Attributes
  t         1x3              24  double
  x         1x1               8  double
  z         1x1              16  double    complex
```

clc: efface la *fenêtre Command* (Command Window) sans affecter les variables.

clear ou **clear all:** efface toutes les variables qui existent dans le *Workspace* (espace de travail).

Exemple

```
>>clear t x %efface les variables t et x
>> who
Your variables are:
z
```

input(' '): demande à l'utilisateur de saisir une valeur en interrompant l'exécution. Cette valeur peut alors être affectée à une variable:

```
>> z=input('Donner la valeur de z: ')
Donner la valeur de z:
```

disp(x): affiche la valeur de la variable x sans faire apparaître le nom de la variable.

Exemple

```
A = [15 150];
S = 'Hello World.';% S est une chaîne de caractères
%Affichage de  la valeur de chaque variable.
>>disp(A)
   15   150
>>disp(S)
Hello World.
>>disp('A')
A
```

num2str: Convertit les chiffres en chaîne de caractères.

```
>> s = num2str(pi)
s =
'3.1416'
>> ch1='MATLAB' ; ver=9.8;
>> ch=[ch1, ' Version ', num2str(ver)]
ch =

    'MATLAB Version 9.8'
```

8. Opérations relationnelles ou logiques

Comme pour les opérations arithmétiques, les opérations logiques qui existent pour les nombres scalaires peuvent aussi être appliquées à des tableaux numériques.

Les opérateurs relationnels permettent de réaliser des comparaisons logiques entre des valeurs numériques.

8.1 Valeurs logiques

La valeur logique "VRAIE" s'appelle `true`, et "FAUX" s'appelle `false` ; ces valeurs sont de type logique. Ces valeurs logiques sont aussi représentées par les valeurs `1` et `0`.

8.2 Opérateurs relationnels

Les opérateurs relationnels permettent la comparaison de deux valeurs entre-elles. Le **tableau 2** synthétise les syntaxes des différents opérateurs relationnels disponibles.

```
>> 3==3
ans =
     1
>>5>6
ans=
     0
```

8.3 Opérateurs logiques

Les opérateurs logiques sont des opérateurs qui s'appliquent exclusivement sur des valeurs de type logique. Ils permettent la combinaison de conditions logiques.

Le **tableau 3** donne la syntaxe des opérateurs logiques disponibles en MATLAB:

tandis que le **tableau 4** rappelle le résultat de ces opérations logiques.

Tous les opérateurs relationnels ou logiques renvoient un résultat égal à la valeur logique **0** ou **1**.

9. Fichiers de programmes

MATLAB peut exécuter une séquence d'instructions stockées dans des fichiers, nommés fichiers M ou M-files. Ce nom provient du fait que l'extension est ".m". il existe deux types de fichiers M: les **fichiers scripts (**aussi appelés fichiers de commandes) et les **fichiers de fonctions.**

9.1 Éditeur/Débogueur MATLAB

MATLAB dispose d'un éditeur/débogueur pour créer et déboguer les **fichiers M**. L'éditeur/débogueur MATLAB est l'interface qui permet d'écrire les scripts et les fonctions. Il peut être lancé dans la fenêtre **Command** (Window command) en tapant "**edit"** ou en appuyant simultanément sur les touches **(Ctrl+N)** ou en cliquant sur File>>New>>M-file. Ou **"edit NomDeFichier"** si le fichier a été crée auparavant .

9.2 Fichiers de commandes ou scripts

Un fichier script est une séquence d'instructions MATLAB. Si, par exemple le fichier est nommé **ex1.m**, alors l'instruction: **>>ex1** provoque l'exécution des instructions contenues dans ce fichier. (on saisit juste **le nom**, sans extension, dans la fenêtre **Command)**. Dans ce cas, le ficher doit être sauvegardé (touches "**Ctrl**"+"**S**", ou **File--->save**) avant l'exécution pour éviter les erreurs éventuelles de modification du fichier M.

On peut aussi effectuer l'exécution par la touche "**F5**" (sauvegarde + exécution) . Un fichier M peut appeler un autre fichier M et peut aussi s'appeler lui-même de façon récursive

>>ex1 % on tape ex1 pour le fichier stocké sous le nom **ex1.m**

9.3 Fichiers de Fonctions

Les fichiers de fonctions fournissent une extensibilité Matlab. Vous pouvez créer de nouvelles fonctions spécifiques à votre domaine de travail qui auront le même statut que toutes les autres fonction MATLAB. Les variables dans les fonctions sont par défaut **locales**. Il est possible de définir des variable globales en utilisant le mot réservé global (exemple: global x y)

9.3.1 Définition et appel d'une fonction

function [y1,y2,.....,ym]=nom(x1,x2,....xn)

Corps de la fonction

où:
y1, y2, ym: sont les arguments de sortie (retour).
x1, x2, xm: sont les arguments d'entrée (d'appel).
Le fichier de la fonction doit être enregistré obligatoirement sous **nom.m**.
On peut mettre plusieurs fonctions dans le même M-file mais seule la fonction du même nom que le fichier peut être utilisée, appelée, à partir de la fenêtre de commandes ou d'une autre fonction ou d'un script.

10. Instructions et commandes structurées

Une instruction peut désigner un instruction simple (expression), conditionnelle, une boucle ou une rupture de séquence. Une commande structurée permet de réaliser des **itérations** ou des **sélections**. MATLAB dispose des instructions structurées suivantes: ***for, while, if*, et *switch***.

Instruction for

Syntaxe

for variable=expression

instructions

end

Les instructions sont répétées un nombre de fois donné. Les colonnes de "expression" (matrice) sont affectées, l'une parés l'autre, à la variable et les instructions sont exécutées. "expression" est généralement un vecteur ligne de la forme **deb:pas:fin**.

Instruction while

Syntaxe

```
while expression
      instructions
end
```

Les instructions sont répétées aussi longtemps que "expression" est **différente de zéro** (expression booléenne **vraie**).
L'expression peut être simple ou composée d'expressions reliées entre elles par des opérateurs relationnels ou logiques (= = , <, >, <=,>=, &, |,~=, etc.). Si "expression" est une variable, les instructions

sont répétées si la partie réelle de cette variable a tous ses éléments non nuls.

Instruction de sélection if

Syntaxe 1

```
if expression
    instructions I1
else
    instruction I2
end
```

Syntaxe 2

```
if expression1
   instructions I1
elseif expression2
    instruction I2
elseif expression3
    instruction I3
else
   instructions exécutées si aucune
   autre expression n'est vraie
end
```

Les instructions "I1" sont exécutées si "expression" est **différente de 0** (expression booléenne **vraie**), autrement les instructions "I2" sont exécutées. La partie ***else*** est facultative. En cas d'instructions conditionnelles imbriquées, un ***else*** est toujours associé au ***if*** le plus proche. Dans le cas de sélections multiples, on utilisera l'instruction ***elseif***.

Exemple 1: soit le fichier **script** suivant, qui permet générer un vecteur contenant les racines carrées des nombres entiers allant de 1 à *n* en utilisant la boucle ***for***. Le fichier est stocké sous le nom **exfor.m**:

```
clear;clc
n=input('Donnez une valeur de n: '),
x=[ ];
for i=1:n
x=[x, sqrt(i)];
end
x
```

Exécution: >>exfor

```
>> Donnez une valeur de n: 5

n =
    5
x =

   1.0000    1.4142    1.7321    2.0000    2.2361
```

Exemple 2: le fichier **script** suivant, permet générer un vecteur contenant les racines carrées des nombres entiers allant de 1 à *n* en utilisant la boucle ***while***. Le fichier est stocké sous le nom **exwhile.m**:

```
clear;clc
n=input('Donnez une valeur de n: '),
x=[ ];
i=1;
while i<=n
x=[x, sqrt(i)];
i=i+1;
end
x
```

Exécution: >>exwile

```
>> Donnez une valeur de n: 5
n =
    5
x =
   1.0000    1.4142    1.7321    2.0000    2.2361
```

Exemple 3: le fichier de **fonction** suivant, permet de créer une **fonction**, nommée **racines.m** pour générer un vecteur contenant les racines carrées des nombres entiers allant de 1 à n en utilisant la boucle ***for***. Le **nom du fichier** contenant la fonction **porte obligatoirement le nom de cette dernière**, donc le fichier est stocké sous le nom **racines.m** et contient les lignes suivantes:

```
function [x]=racines(n)
x=[ ];
for i=1:n
x=[x, sqrt(i)];
end
```

Puis dans la fenêtre de commandes on tape:

```
>>[x]=racines(5) % ou x=racines(5) ou x1=racines(5)
x =
   1.0000    1.4142    1.7321    2.0000    2.2361

>>racines(5)
ans =
   1.0000    1.4142    1.7321    2.0000    2.2361
```

On peut aussi appeler la fonction définie précédemment à partir d'un script, nommé par exemple **ex1.m**, comme suit:

```
ex1.m% Ne pas écrire ex1.m dans la 1ière ligne de  l'éditeur.
%ex1.m:  représente  le  nom du fichier script
clear all;clc % La saisie (le code ) commence ici
[x1]=racines(5)
```

L'exécution du **fichier script ex1.m** par la touche "**F5**", par exemple, permet d'appeler la **fonction racines.m**

```
x1=
   1.0000    1.4142    1.7321    2.0000    2.2361
```

On peut aussi appeler la fonction racines plusieurs fois pour différentes valeurs de *n*:

ex2.m % nom du fichier script

```
clear all;clc
[x1]=racines(3)
[x2]=racines(5)
[x3]=racines(7)
```

L'exécution du **fichier script ex2.m** donne:

```
x1=
   1.0000    1.4142    1.7321
x 2=
   1.0000    1.4142    1.7321    2.0000    2.2361
x 3=
  1.0000   1.4142   1.7321   2.0000   2.2361   2.4495   2.6458
```

Exemple 4: Le fichier script suivant permet d'afficher la mention obtenue en fonction de la moyenne de l'étudiant:
si la moyenne <10 , l'étudiant est Ajourné
si la moyenne >=10 , la mention est Passable
si la moyenne >=12 , la mention est Assez bien
si la moyenne >=14 , la mention est Bien
si la moyenne >=16 , la mention est Très Bien

exif.m

```
moy=input('Donnez une moyenne')
if moy >=16
mention='Très bien'
elseif moy>=14
mention=' Bien'
elseif moy>=12
mention=' Assez bien'
elseif moy>=10
mention=' Passable'
else
'Ajourné(e)'
end
```

Exécution: >>exif

```
Donnez une moyenne: 13.5
ans=
     Assez bien
Donnez une moyenne: 8
ans=
     Ajourné(e)
```

10. Instructions de rupture de séquence

break: termine l'exécution d'une boucle (*for* ou *while*). Si plusieurs boucles sont imbriquées, *break* permet de sortir de la boucle la plus proche.

continue: L'instruction *continue* est utilisée pour terminer l'itération en cours et force l'itération suivante d'une boucle *for* ou *while*.

ex_break.m

```
a = 10;
while (a < 20 )
  fprintf('valeur de a: %d\n', a);
  a = a+1;
    if( a > 15)
      break; % termine la boucle en utilisant break
    end
end
```

Exécution

```
valeur de a: 10
valeur de a: 11
valeur de a: 12
valeur de a: 13
valeur de a: 14
valeur de a: 15
```

ex_continue.m

```
a = 10;
while a < 20
  if a == 15
     a = a + 1;
    continue; % termine cette itération
  end
fprintf('valeur de a: %d\n', a);
a = a + 1;
end
```

Exécution

```
valeur de a: 10
valeur de a: 11
valeur de a: 12
valeur de a: 13
valeur de a: 14
valeur de a: 16
valeur de a: 17
valeur de a: 18
valeur de a: 19
```

Fonction	Description
ones(i,j)	crée un tableau de i lignes j colonnes contenant des 1
zeros(i,j)	crée un tableau de i lignes j colonnes contenant des 0
eye(i,j)	crée un tableau de i lignes j colonnes avec des 1 sur la diagonale principale et 0 ailleurs
diag(u)	crée une matrice carrée avec le vecteur u sur la diagonale et 0 ailleurs
diag(U)	extrait la diagonale de la matrice U
triu(A)	renvoie la partie supérieure de A
tril(A)	renvoie la partie inférieure de A
linspace(a,b,n)	crée un vecteur de n composantes uniformément réparties de a à b
A\b	résolution du système linéaire `Ax=b`
cond(A)	conditionnement d'une matrice (norme euclidienne)
det(A)	déterminant d'une matrice
rank(A)	rang d'une matrice
inv(A)	inverse d'une matrice
svd(A)	valeurs singulières d'une matrice
norm(A)	norme matricielle ou vectorielle
u'	prend le transposé de u
u*v	multiplication matricielle
u+v	addition matricielle
u-v	soustraction matricielle
u.* v	multiplication des tableaux u et v terme à terme
u./v	division du tableau u par le tableau v terme à terme

TAB. 1 – Principales opérations sur les matrices.

Opération relationnelle	Syntaxe MATLAB
	symbole
A égal à B	`A == B`
A différent de B	`A ~= B`
A supérieur à B	`A > B`
A supérieur ou égal à B	`A >= B`
A inférieur à B	`A < B`
A inférieur ou égal à B	`A <= B`

TAB. 2 - Syntaxe des opérateurs relationnels

Opération logique	Syntaxe MATLAB	
	symbole	fonction
A et B	`A & B`	`and(A,B)`
A ou B	`A | B`	`or(A,B)`
A ou exclusif B		`xor(A,B)`
Négation de A	`~A`	`not(A)`

TAB. 3 - *Syntaxe des opérateurs logiques*

A	B	and(A,B)	or(A,B)	xor(A,B)	not(A)
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

TAB. 4 - *Tableau logique*

`A` et `B` peuvent être des valeurs scalaires logiques ou des tableaux de valeurs logiques de même dimensions. Pour les tableaux, ces opérations s'appliquent terme à terme.

symbole	*fonction*
+	addition de réels et de matrices
-	soustraction de réels et de matrices
*	produit de réels et de matrices
.*	produit élément par élément de matrices
^	élévation à une puissance de réels et de matrices
.^	puissance élément par élément de matrices
\	division à gauche de réels et de matrices
/	division à droite de réels et de matrices (division classique)
./	division élément par élément de matrices
==	égalité
~=	différent
<	strictement inférieur
<=	inférieur ou égal
>	strictement supérieur
>=	supérieur ou égal
&	ET logique (AND)
|	OU logique (OR)
~	NON logique (NOT)
xor	OU exclusif (XOR)
%	commentaires
=	affectation
'	transposée de matrices et délimiteur de chaînes de caractères
!	commandes système (échappement)
,	séparation d'instructions ou de réels dans une matrice
;	séparation d'instructions (pas d'affichage de valeurs intermédiaires) ou des lignes d'une matrice
...	symbole de continuation (pour terminer une instruction ou une expression à la ligne suivante)
.	point décimal
..	répertoire parent du répertoire en cours
[ ]	définition de vecteurs ou de matrices
( )	utilisation dans des expressions ou des fonctions

TAB. 5- *Opérateurs et caractères spéciaux*