Chapter 2 : MATLAB Course

1. MATLAB is stand for **MAT**rix **LAB**oratory.

2. **MATLAB**

MATLAB is an interactive working environment in which the user can carry out quite complex computational tasks with few commands.

3. **Features of MATLAB**

Following are the basic features of MATLAB :

- **a.** It is a high-level language for numerical computation, visualization and application development.
- **b.** It also provides an interactive environment for iterative exploration, design and problem solving.
- **c.** It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
- **d.** It provides built-in graphics for visualizing data and tools for creating custom plots.
- **e.** MATLAB's programming interface gives development tools for improving code quality maintainability and maximizing performance.
- **f.** It provides tools for building applications with custom graphical interfaces.
- **g.** It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

4. **Uses of MATLAB**

MATLAB is widely used as a computational tool in science and engineering encompassing the fields of physics, chemistry, math and all engineering streams. It is used in a range of applications including:

- **a.** Signal Processing and Communications
- **b.** Image and Video Processing
- **c.** Control Systems
- **d.** Test and Measurement
- **e.** Computational Finance
- **f.** Computational Biology

5. **Mathematical Operations**

No.	Operation	Symbol	Example	Priority	Sequence
1	Parenthesis ( )	( )	(2+4)^5	Highest	1
2	power $\mathbf{a^b}$	^	2^8 =2 ^ 8	Highest	1
3	Division ÷	/ or \	$\frac{56}{8} = {}^{56}/_{8} = 8\backslash 56$	High	2
4	Multiplication ×	*	3.14 * 0.88	High	2
5	Subtraction –	–	90 – 45	Low	3
6	Addition +	+	3 + 22	Low	3

Example	
$\frac{(2^8+4)\times 5^2}{8}$ = ((**2^8**+4)* **5 ^ 2**) / 8 = (**(**256+4**)*** **5 ^ 2**) / 8 = **(**260 * 25**)** / 8 = 6500 / 8 = 812.5	$(\frac{3^2}{2}+\frac{1}{2})^{\frac{1}{2}}$ = ((**3^2**) / 2 + **1 / 2**) ^ (**1 / 2**) = (**9 / 2** + 0.5) ^ 0.5 = (**4.5 + 0.5**) ^ 0.5 = **5 ^ 0.5** = 2.2361

Home Work		
$\frac{6}{2}+2$	$\frac{6+2}{2}$	$\frac{(2+4^3)\times 3}{5/_{2^{-2}}}$

6. Variables & Constants

The ***variables*** and ***constants*** in MATLAB are both represented by letter (s) but with one difference:

- The ***variables*** are that which represents numbers.
- The ***Constants*** are that which represents alphabet(s) (i.e. string).

$$\boldsymbol{variables}\begin{cases} x = 12 & \text{(wher x takes 8 byte)} \\ y = 0.5452 & \text{(wher y takes 8 byte)} \\ z = -1 & \text{(wher z takes 8 byte)} \\ r = 0 & \text{(wher r takes 8 byte)} \end{cases}$$

$$\boldsymbol{constants}\begin{cases} x = '12' & \text{(wher x takes 4 byte)} \\ y = '0.5452' & \text{(wher y takes 12 byte)} \\ z = '-1' & \text{(wher z takes 4 byte)} \\ r = '0' & \text{(wher r takes 2 byte)} \end{cases}$$

Home Work
Determine the type and the number of bytes for the following: 'John' , 114 , 'and' , 0.007, 7736384 , 'TOMORROW' , '12.1123'

7. Variables & Constants Naming Rules

The variables and constants will obey the same rules in MATLAB, which are:

a. They are case sensitive (i.e. Cost, COST, CoSt, cost, all are differ)
b. Maximum length for the name is 63 letters, any more will be neglected (Howaboutthisvariablename = 3)
c. Any name must not start with number or contain any space (HowAboutThis = 20, how_about_this ='d', X3251 = -4.27)

8. Reserved Words in MATLAB

The following words can't be used as variables in MATLAB because MATLAB use them in its operations, these words are:

for	end	if	while	function	return	elseif	case	otherwise
switch	continuo	else	try	catch	global	persistent	break	

9. Special Variables Predefined in MATLAB

The following variables are predefined in MATLAB, which are:

No.	Variable	Notes
1	ans	the variable that store the results
2	beep	make the computer to beep
3	pi	the ratio of circle circumstance to its diameter
4	eps	the minimum number that should be added to 1 made the number greater than 1 (eps = 2.2204e-016)
5	inf	refer to infinity (i.e. 1/0)
6	NaN	refer to the input is (*not a number*) (i.e. 0/0)
7	i or j	refer to $\sqrt{-1}$

Starting with MATLAB

- **كيفية فتح برنامج ال MATLAB وغلقه :**

$MATLAB\ (shortcut\ on\ desktop)$

Or $start \rightarrow all\ programs \rightarrow MATLAB$

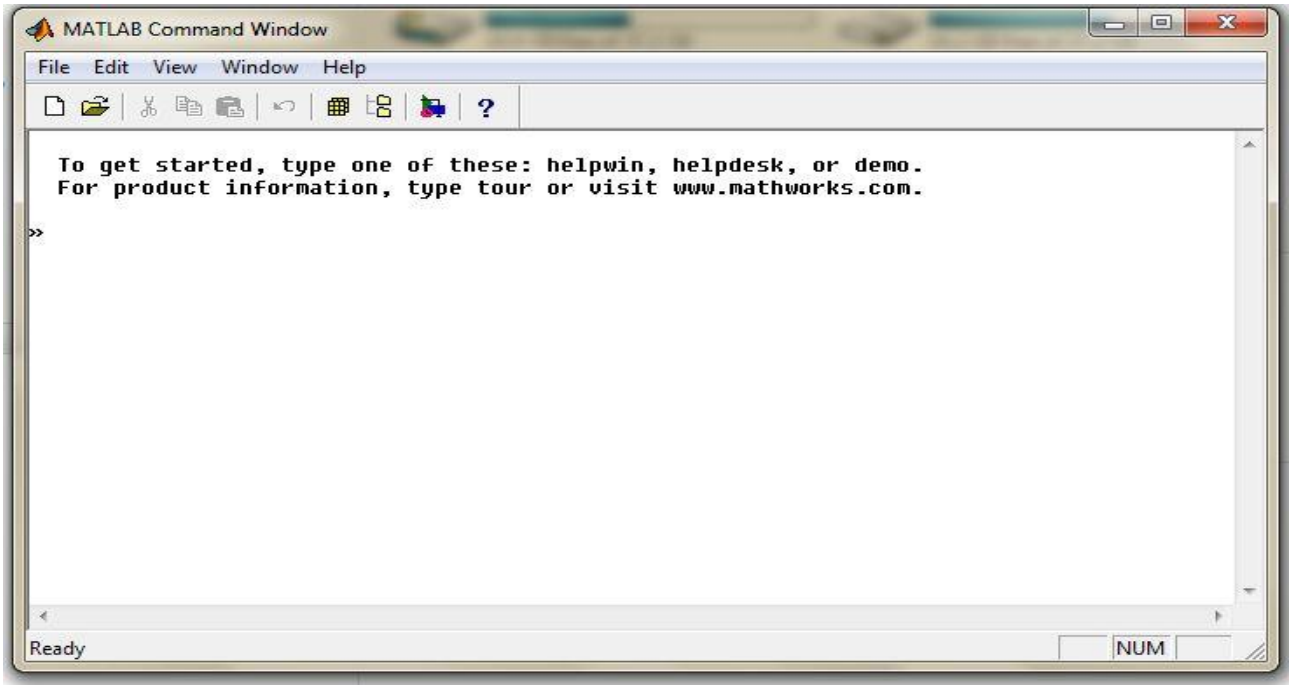
- **غلق برنامج MATLAB**

1. من زر الإغلاق في نافذة واجهة تطبيق الماتلاب .

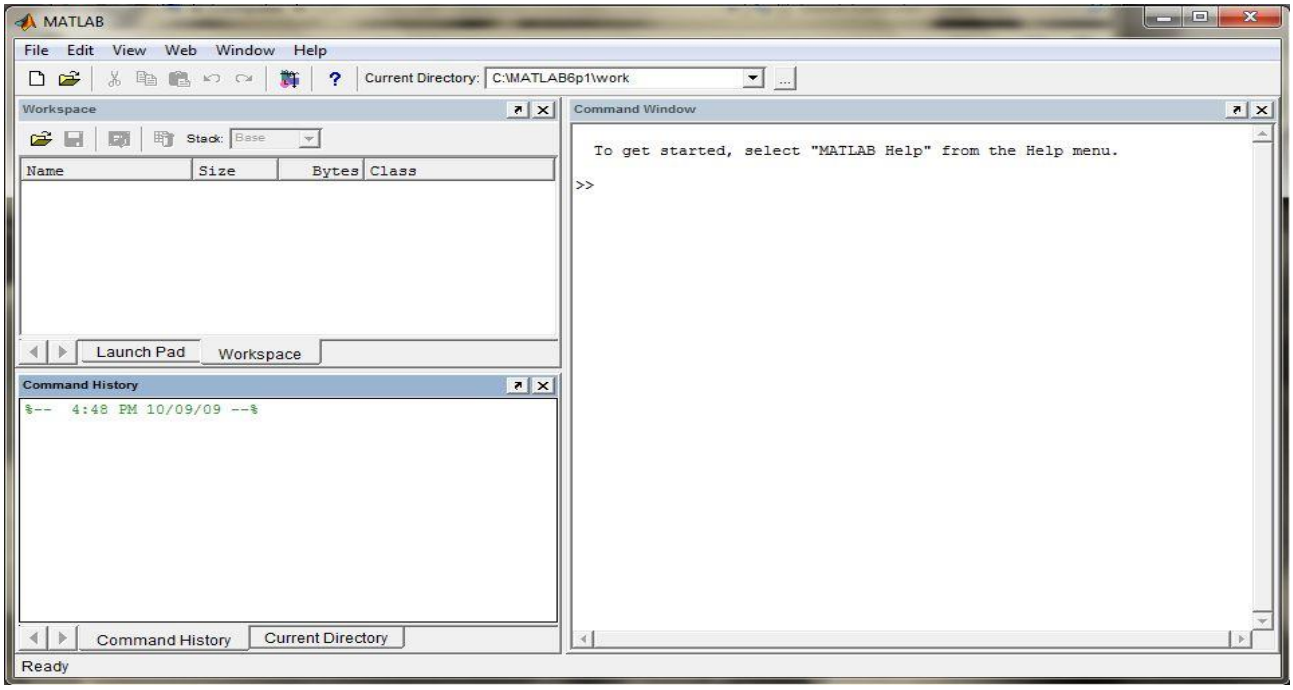
2. غلق برنامج MATLAB من خلال قائمة الملف File ← Exit Matlab

- **التعرف على واجهة التطبيق :**

1. في الإصدار (5.3) :

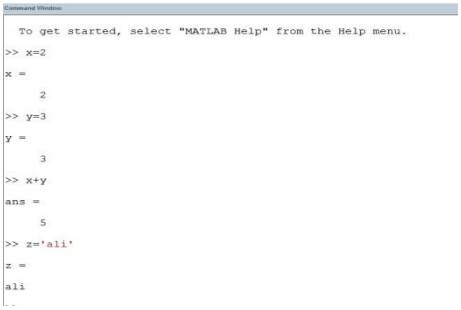


2. في الإصدار (6.1) :

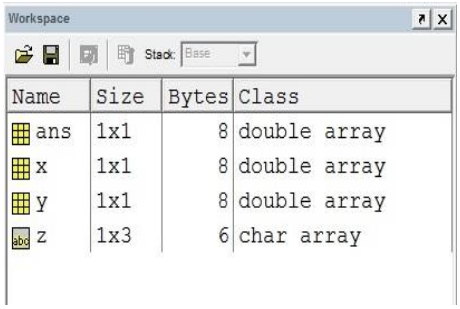


The main MATLAB Panel Consists :

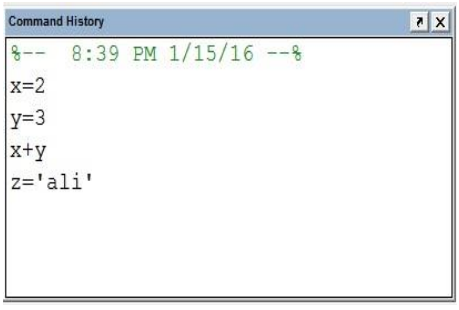
1. **Command Window:** This is the main area where commands can be entered at the command line. It is indicated by the command prompt (>>).



2. **Workspace**: The workspace shows types , sizes , Bytes to save and the class of all the variables created and/or imported from files.

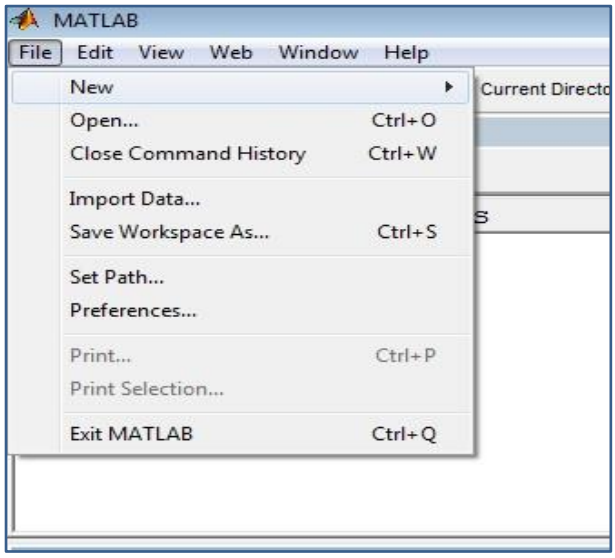


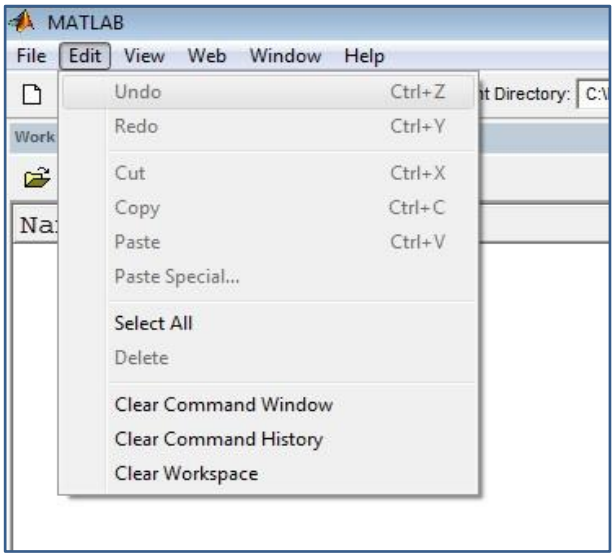
3. **Command History** : This panel shows or rerun commands that are entered at the command line.

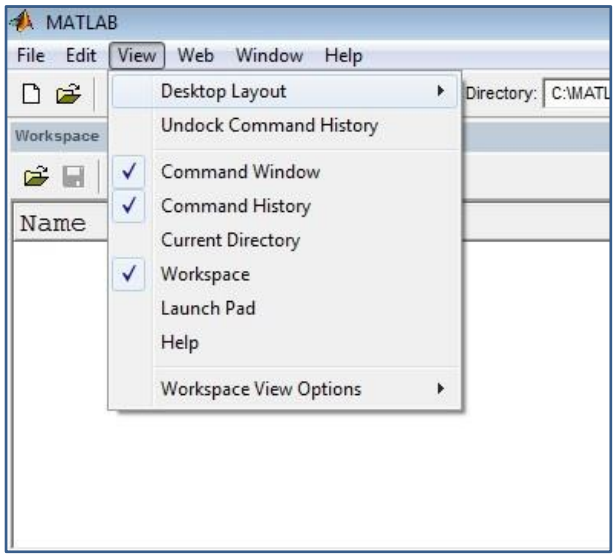


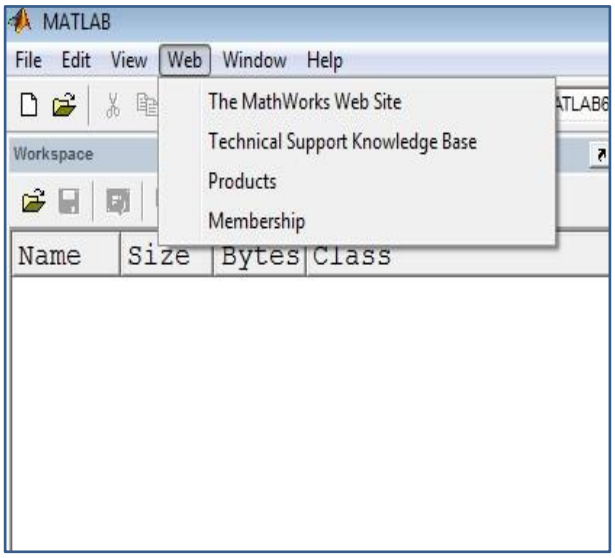
- **Menus Bar : consists of:**

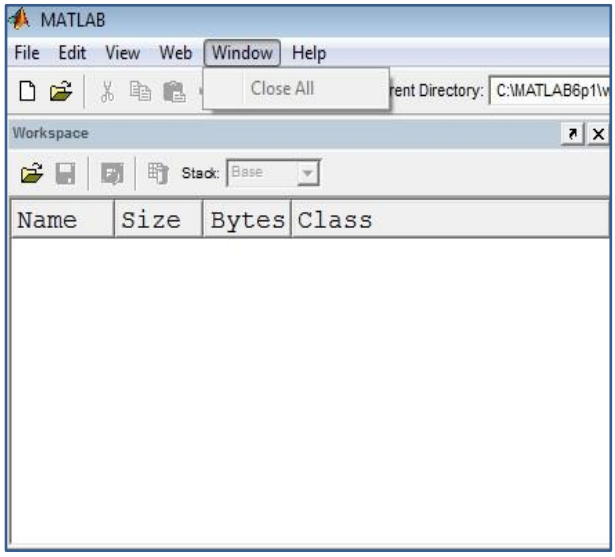
File	Edit	View	Web	Window	Help

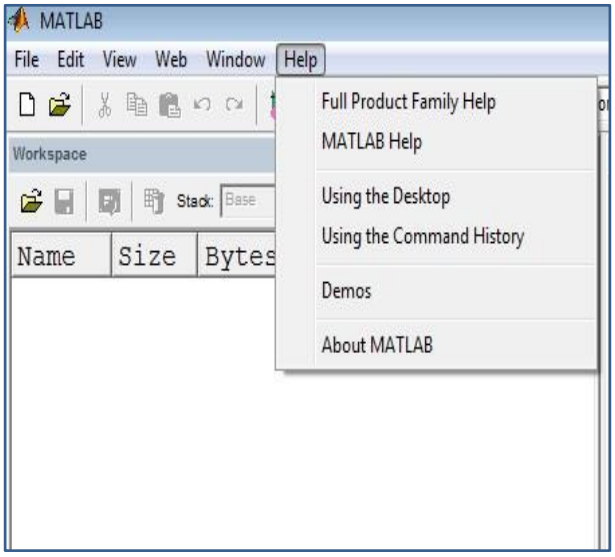












Display Formats for Numeric Values

All the variables store and process in MATLAB in double precision form but to view the number there are eleven forms, which are:

No.	Format	Pi & x = -1.916e-4	Notes
1	format short (default)	3.1416 , –1.9160 e–004	(4) digits after mantissa
2	format long	3.14159265358979 , –1.91600000000000 e–004	(14) digits after mantissa
3	format short e	3.1416e + 000 , –1.9160e–004	(4) digits after mantissa with exponential power
4	format long e	3.14159265358979 e+000 , –1.916000000000000 e–004	(14) digits after mantissa with exponential power
5	format short g	3.1416 , –0.0001916	the best between format short and format short e
6	format long g	3.14159265358979 , –0.0001916	the best between format long and format long e
7	format hex	400921fb54442d18 ,	hexadecimal form
8	format bank	3.14 , –0.00	two digit after mantissa
9	format +	+ –	positive (+) or Negative (–) or zero (0)
10	format rat	355 / 113 –29/151357	rational form
Home Work			
For the following numbers change the format of displaying number for all possible ways: -358.168e-5 , 196834.5677768129 , -85.23517291 , 1.6e-19			

Defining Variables

In MATLAB you can process the data directly in command window, or you can define theme as variables then process theme, like:

Direct Data	Using Variables	
>> 24 + 5 ↵ ans = 29 >> ans – 5 ↵ ans = 24	>> x = 2 ↵ x = 2 >> y = x^2 ↵ y = 4 >> z = x + y ↵ z = 6	>> r = z / 3 ↵ r = 2 >> r * ((z – x) ^ y) ↵ ans = 512

Defining Matrices

In order to define and deal with matrices in MATLAB, there are certain characters must be taken into account:

1. **The brackets [ ]** are used to define a matrix.
2. **Separating the elements** we either use the ***comma (,)*** or the ***space ($_{space}$)***.

Using Comma	Using Space
>> x = [1,2,3] ↵ x = 1 2 3	>> x = [1 2 3] ↵ x = 1 2 3

3. **To determine the end of the rows** we use **the *semicolon* (;)** or the ***enter*** (↵)

Define Matrix	Define Column Vector
>> x = [1 2 3; 4 5 6; 7 8 9] ↵ x = 1 2 3 4 5 6 7 8 9	>>y=[1↵ 2↵ 3] ↵ y = 1 2 2

Note that the semi-colon also used to hide the executing the result:

Without Using Semicolon	Using Semicolon	Without Using Semicolon	Using Semicolon
>> x = [1 2 3] ↵ x = 1 2 3	>> x = [1 2 3]; ↵	>>2+4↵ ans = 6	>>2+4; ↵

4. **The parentheses ( ) :**

The standard formula of parentheses is:

Operator (Target)

There are two different ways to use the parentheses:

a. To specify certain element in a matrix, the general form is:

A(row, column),

But for vectors A(index).

```
>> x = [4 5 6;3 2 1] ↵
                x =  4  5  6
                     3  2  1
>> x(1,2) ↵
>> ans =
            5
```

b. To rebuild the a new matrix from a predefined matrix depending on the position sequence of the element in the matrix:

```
>> x = [4 5 6]; ↵
>>y =x([2  1  3  2  2]) ↵
y =  5    4    6    5    5
```

```
>> z = [-1 0 2;2 4 3;1 0 4] ↵
```

$$z = \begin{matrix} -1 & 0 & 2 \\ 2 & 4 & 3 \\ 1 & 0 & 4 \end{matrix}$$

```
>> r =z([2  5  7  2;1  2  4  9]) ↵
```

$$r = \begin{matrix} 2 & 4 & 2 & 2 \\ -1 & 2 & 0 & 4 \end{matrix}$$

c. To give a detail of the matrix:

```
>> x = [4 5 6]; ↵
>>size(x) ↵
ans =     1     3
>> y = [4 5 6;3 2 1] ↵
```

$$y = \begin{pmatrix} 4 & 5 & 6 \\ 3 & 2 & 1 \end{pmatrix}$$

```
>>size(y) ↵
ans =     2     3
```

5. The colon (:) used in three different ways:

a. To create a vector with defined increment as the following syntax

[*First value: Increment : Last Value*]

Positive Increment	Negative Increment
>> x = [1:2:10] ↵ x = 1 3 5 7 9	>>x = [10:-2:1] ↵ x = 8 6 4 2

b. To create a sub matrix of a certain row or column, **or** to re-arrange the matrix in one column.

Specified Row	Specified Column	One Column Matrix
>> x = [1 2 3; 4 5 6; 7 8 9]; ↵ >> y = x(2,:) ↵ y = 4 5 6	>> x = [1 2 3; 4 5 6; 7 8 9]; ↵ >> y = x(:,2) ↵ y = 2 5 8	>> x = [1 2 3; 4 5 6; 7 8 9]; ↵ >> y = x(:)↵ y = 1 8 4 3 7 6 2 9 5 Column after column

c. To create a sub matrix of defined rang of rows or columns, usually the word ***end*** used to indicate the last index in the range.

Define Matrix	Sub Matrix	Sub Matrix Using End
>> x = [1 2 3; 4 5 6; 7 8 9] ↵ x = 1 2 3 4 5 6 7 8 9	>> y = x(2:3,1:3) ↵ y = 4 5 6 7 8 9	>> y = x(2:end,1:end) ↵ y = 4 5 6 7 8 9

or

d. to create a new matrix by merging two (or more) predefined matrices

Define Matrices	Define Matrices
>> A = [1 2 3; 4 5 6; 7 8 9]; ↵ >> B = [-1 0 -3; -5 2 11; -2 8 0]; ↵ >> C=[A B] ↵ C= 1 2 3 -1 0 -3 4 5 6 -5 2 11 7 8 9 -2 8 0	>> A = [1 2 3; 4 5 6; 7 8 9]; ↵ >> B = [-1 0 -3; -5 2 11; -2 8 0]; ↵ >> D=[A;B] ↵ D= 1 2 3 4 5 6 7 8 9 -1 0 -3 -5 2 11 -2 8 0

Note/ we can also create a vector with defined increment by using the ***linespace*** function or ***logspace*** function as the following

i)Using Colon
[First value: Increment: Last Value]
>> x = [1:0.1:100]*pi↵
x = 0 0.3142 0.6283 0.9425 1.2566 1.5708 1.8850 2.1991 2.5133 2.8274 3.1416
ii)Using Linspace
linspace(First value, Last value, Number of values)
>> x = linspace(0,pi,11) ↵
x = 0 0.3142 0.6283 0.9425 1.2566 1.5708 1.8850 2.1991 2.5133 2.8274 3.1416
iii)Using Logspace
logspace(First exponent, Last exponent, Number of values)
>> x = logspace(0,2,11) ↵
x = 1.0000 1.5849 2.5119 3.9811 6.3096 10.0000 15.8489 25.1189 39.8107 63.0957 100.00

Home Works

If $A_{3\times3} = \begin{bmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{bmatrix}$, find the following:

4. Define the matrix A in MATLAB.

5. Find the value of A(2,3), A(1,3), A(2,1), A(1,1).

6. Create the following sub matrices from A:

$i) B = \begin{bmatrix} 6 & 5 \\ 9 & 8 \end{bmatrix}$, ii) $C = \begin{bmatrix} 2 & 1 \\ 5 & 4 \end{bmatrix}$, iii) $D = \begin{bmatrix} 2 & 1 \\ 5 & 4 \\ 8 & 7 \end{bmatrix}$, iv) $E = \begin{bmatrix} 2 \\ 5 \\ 8 \end{bmatrix}$, v) $F = [3 \quad 2 \quad 1]$.

7. Create the matrix ***G*** by merging the above ***D*** & ***E*** matrices.

8. How could you merge the matrices ***D*** & ***F*** in one matrix?

9. Create and name a (4×2) matrix from the ***B*** & ***C*** matrices.

Write the MATLAB statement to:

1- Find the even numbers in the range (1-1000).

2- Find the odd numbers in the range (1-1000).

3- Divide the range (0-6) to 20 numbers using linear manner.

4- Divide the range (0-1000) to 31 numbers using logarithmic manner.

Matrix and Element - wise Operations (using .)

Operator	Description
+ **Plus**	Addition or unary plus. ***A*** + ***B*** adds the values stored in variables ***A*** and ***B***. ***A*** and ***B*** must have the same size, unless one is a scalar. a scalar can be added to a matrix of any size.
- **Minus**	Subtraction or unary minus. ***A*** **-** ***B*** subtracts the value of ***A*** and ***B***. ***A*** and ***B*** must have the same size, unless one is a scalar. a scalar can be subtracted from a matrix of any size.

Examples

```
>>G=[-1 0 1;3 4 5]; ↵
>> G - 2↵
ans =
    −3   −2   −1
     1    2    3
```

```
>>A=[1 2 3;4 5 6]; ↵
>>B=[2 1 5;0 3 4]; ↵
>>A+B↵
ans =
     3   3   8
     4   1   9
```

```
>>A=[1 2 3;4 5 6]; ↵
>>B=[2 1 5;0 3 4]; ↵
>>A+2*B-3↵
ans =
     2   3   10
     1   8   11
```

Operator	Description
***** **Cross**	Matrix multiplication. ***C = A * B*** is the linear algebraic product of the matrices ***A*** and ***B***. More precisely, $$C(i,j) = \sum_{k=1}^{n} A(i,k)B(k,j)$$ For non-scalar ***A*** and ***B***, the number of columns of ***A*** must be equal to the number of rows of ***B***. A scalar can multiply a matrix of any size.
.*	Array multiplication. ***A .* B*** is the element-by-element product of the arrays ***A and B***. ***A and B*** must have the same size, unless one of them is a scalar.

Operator	Description
/	Slash or matrix right division. ***B / A*** (i.e. ÷) is roughly the same as ***B * inv(A)***. More precisely, ***B / A = (A' \ B')'.***
./	Array right division. ***A . / B*** is the matrix with elements ***A(i , j)/ B(i , j)***. ***A and B*** must have the same size, unless one of them is a scalar.

Operator	Description
\	Backslash or matrix left division . If ***A*** is a square matrix, ***A\ B*** is roughly the same as ***inv(A)*B*** , except it is computed in a different way. If ***A*** is an **n-by-n** matrix and ***B*** is a column vector with n components, or a matrix with several such columns, then ***X = A \ B*** is the solution to the equation ***AX = B***.
.\	Array left division. ***A. \ B*** is the matrix with elements ***B(I , j) / A(I , j)***. ***A*** and ***B*** must have the same size, unless one of them is a scalar.

Operator	Description
^	Matrix power. ***X^p*** is X^p, if ***p*** is a scalar. If ***p*** is an integer, If the integer is negative, ***X*** is inverted first.
.^	Array power. ***A . ^ B*** is the matrix with elements ***A(i , j)*** to the ***B(i , j)*** power. ***A*** and ***B*** must have the same size, unless one of them is a scalar.

Operator	Description
'	Matrix transpose. A' is the linear algebraic transpose of A. For complex matrices, this is the complex conjugate transpose.

.'	Array transpose. A.' is the array transpose of A. For complex matrices, this does not involve conjugation.

Some operations are intended for matrices in particular. These include:

1. The conjugate and non-conjugate transpose (') and (. ')

Transpose Vector (.')	Transpose & Finding Complex Conjugate
>> $x = [1-i \quad 2-2i \quad 3-2i]\,.'$ ↵ x = $1-i$ $2-2i$ the transpose $3-2i$	>> $x = [1-i\,,2-2i\,,3-2i\,]'$ ↵ x = $1+i$ $2+2i$ the complex conjugate transpose $3+2i$
>> $x = [1-i\,,2-2i\,;2+i\,,3-2i].'$ ↵ x = $1-i \quad 2+i$ $2-2i \quad 3-2i$ the transpose	>> $x = [1-i\,,2-2i\,;2+i\,,3-2i]'$ ↵ x = $1+i \quad 2-i$ $2+2i \quad 3+2i$ the complex conjugate transpose

Homework

Transpose the matrices $X = \begin{bmatrix} 1+i & 2+2i \\ 1-2i & 2+i \end{bmatrix}$ & $Y = \begin{bmatrix} 4 & 2 \\ 1 & 3 \end{bmatrix}$:

1. The numbers and changing the sign of the imaginary part.
2. The numbers only.
3. Change the signs of matrix X elements without transpose theme.
4. Prove that **digits after mantissa**

2. The mathematical Operations

The mathematical operations between number and matrix have the same priority as it between numbers:

a.

Number-Matrix Operations

>>2*G - 1

ans =

$$\begin{bmatrix} 2*(-1)-1 & 2*0-1 & 2*1-1 \\ 2*3-1 & 2*4-1 & 2*5-1 \end{bmatrix} = \begin{bmatrix} -3 & -1 & 1 \\ 5 & 7 & 9 \end{bmatrix}$$

>>2*G/5+1 % It is not 2*G/(5+1)

ans =

$$\begin{bmatrix} 2*\frac{(-1)}{5}+1 & 2*\frac{0}{5}+1 & 2*\frac{1}{5}+1 \\ 2*\frac{3}{5}+1 & 2*\frac{4}{5}+1 & 2*\frac{5}{5}+1 \end{bmatrix} = \begin{bmatrix} 0.6 & 1 & 1.4 \\ 2.2 & 2.6 & 3 \end{bmatrix}$$

10.Matrix-Matrix Operations

The operation between matrices is more complex than it and numbers as follow, Let $G = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and $H = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$ then:

>>G + H (adding two matrices)

ans =

$$\begin{matrix} 2 & 3 \\ 5 & 6 \end{matrix}$$

>>2 * G – H

ans =

$$\begin{matrix} 2*1-1 & 2*2-1 \\ 2*3-2 & 2*4-2 \end{matrix} = \begin{matrix} 1 & 3 \\ 4 & 6 \end{matrix}$$

The (.) transform the operation from matrix-to-matrix to element-to-element.

>> G * H

ans =

$$\begin{matrix} 1*1+2*2 & 1*1+2*2 \\ 3*1+4*2 & 3*1+4*2 \end{matrix} = \begin{matrix} 5 & 5 \\ 11 & 11 \end{matrix}$$

>>G . * H

ans =

$$\begin{matrix} 1*1 & 2*1 \\ 3*2 & 4*2 \end{matrix} = \begin{matrix} 1 & 2 \\ 6 & 8 \end{matrix}$$

>>H / G % = G\H = H*G^{-1} where G^{-1}= $\begin{matrix} -2 & 1 \\ 1.5 & -0.5 \end{matrix}$

ans =

$$\begin{matrix} -0.5 & 0.5 \\ -1 & 1 \end{matrix}$$

>>H . / G % = G.\H

ans =

$$\begin{matrix} 1/1 & 1/2 \\ 2/3 & 2/4 \end{matrix} = \begin{matrix} 1 & 0.5 \\ 0.667 & 0.5 \end{matrix}$$

>>1 . / G % = G.^-1 but 1/G can't be performed G/3=G./3

ans =

$$\begin{matrix} 1/1 & 1/2 \\ 1/3 & 1/4 \end{matrix} = \begin{matrix} 1 & 0.5 \\ 0.33 & 0.25 \end{matrix}$$

>>2 . ^ G

ans =

$$\begin{matrix} 2\text{^}1 & 2\text{^}2 \\ 2\text{^}3 & 2\text{^}4 \end{matrix} = \begin{matrix} 2 & 4 \\ 8 & 16 \end{matrix}$$

>>H . ^ G

ans =

$$\begin{matrix} 1\text{^}1 & 1\text{^}2 \\ 2\text{^}3 & 2\text{^}4 \end{matrix} = \begin{matrix} 1 & 1 \\ 8 & 16 \end{matrix}$$

>>H ^ G %illegal at least one operand must be scalar

11. The Properties of Scalar-Matrix and Matrix-Matrix Operations

A = [a_1 a_2 . . . a_n], B = [b_1 b_2 . . . b_n], and c = scalar then:

1- A + c = [a_1+c a_2+c . . . a_n+c]
2- A – c = [a_1-c a_2-c . . . a_n-c]
3- A*c = [a_1*c a_2*c . . . a_n*c]
4- A/c = [a_1/c a_2/c . . . a_n/c]
5- A + B = [a_1+b_1 a_2+b_2 . . . a_n+b_n]
6- A.*B = [a_1*b_1 a_2*b_2 . . . a_n*b_n]
7- A./B = [a_1/b_1 a_2/b_2 . . . a_n/b_n]
8- B.\A = [a_1/b_1 a_2/b_2 . . . a_n/b_n]
9- A.^c = [a_1^c a_2^c . . . a_n^c]
10- c.^A = [c^a_1 c^a_2 . . . c^a_n]
11- A.^B = [a_1^b_1 a_2^b_2 . . . a_n^b_n]

12. Creating Standard Matrices

MATALB provide functions to create standard matrix like I, D, 0,…etc. these functions:

1- **ones():** create matrix all its elements are equal to one.

Create Square Matrix	Create mxn Matrix
>> ones(3) ans = 1 1 1 1 1 1 1 1 1	>> ones([3 3]) %ones(3,4) ans = 1 1 1 1 1 1 1 1 1

2- **zeros( ):** create matrix all its elements are zeros.

Create Square Matrix	Create mxn Matrix
>> zeros(3) ans = 0 0 0 0 0 0 0 0 0	>> zeros([3,4]) %zeros(3,4) ans = 0 0 0 0 0 0 0 0 0 0 0 0

3- **eye( ):** create unity matrix (I).

Create Square Matrix	Create mxn Matrix
>> eye(3) ans = 1 0 0 0 1 0 0 0 1	>>eye([3,4]) eye(3,4) ans = 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0

4- **rand():** create random matrix its elements are in the range (0-1).

Create Square Matrix	Create mxn Matrix
>> rand(3) ans = 0.8147 0.9134 0.2785 0.9058 0.6324 0.5469 0.1270 0.0975 0.9575	>> rand([3,1]) %rand(3,1) ans = 0.9649 0.1576 0.9706

5- **randn():** Generate values from a normal distribution with mean 0 and standard deviation 1.

Create Square Matrix	Create mxn Matrix
>>randn(3) ans = 0.7254 −0.2050 1.4090 −0.0631 −0.1241 1.4172 0.7147 1.4897 0.6715	>>randn([3,1]) %randn(3,1) ans = 0.7269 −0.3034 0.2939

6- **diag():**is used:

a- to create diagonal matrix (D), using the vector V, its position is determine by the value of k.

$$k = \begin{cases} = 0 & on\ the\ main\ diagonal \\ > 0 & upper\ the\ main\ diagonal \\ < 0 & under\ the\ main\ diagonal \end{cases}$$

b- to determine the main diagonal of a matrix.

Create Diagonal Matrix	Create Diagonal Matrix away from main diagonal	Determine the Matrix Diagonal
>> V = [1 2 3]; >>diag(V,0) % b same as diag(V) ans = 1 0 0 0 2 0 0 0 3	>>diag(V,1) ans = 0 1 0 0 0 0 2 0 0 0 0 3 0 0 0 0 >>diag(V,-2) ans = 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 2 0 0 0 0 0 3 0 0	>>x = floor(rand(3)*10) x = 0 7 4 4 7 4 3 1 6 >>diag(x) ans = 0 7 6

7- to create matrix of equal elements there are four ways to do so:

Via Multiplication (Very Slow)	Via Addition (Slow)	Via Direct Substitution (Fast)	Via repmat (Very Fast)
>>ones([2,3])*2 ans = 2 2 2	>>2+zeros([2,3]) ans = 2 2 2	>>d=2; >>d(ones([2,3])) ans =	>>repmat(2,[2,3]) ans = 2 2 2

2 2 2	2 2 2	2 2 2 2 2 2	2 2 2

13. Functions Frequently Used With Matrices

There are few function that are frequently used with matrices, these functions are:

1- size(): return the dimensions of a matrix:

All Dimensions	Two Dimensions	Specified Dimension
>> x = [1 2 3;4 5 6]; >> size(x) ans = 2 3	>> x = [1 2 3;4 5 6]; >> [rows columns] = size(x) rows = 2 columns = 3	>> x = [1 2 3;4 5 6]; >> rows = size(x,1) rows = 2 >> columns = size(x,2) columns = 3

2- Length: return the Length of vector or largest matrix dimension:

Length of Vector	Largest Matrix Dimensions
>> x = [1 3 5 7]; >> length(x) ans = 4	>> x = [1 2 3;4 5 6]; >> length(x) ans = 3

3- numel: return the number of elements in array:

Number of Elements in a Vector	Number of Elements in a Matrix
>> x = [1 3 5 7]; >>numel(x) ans = 4	>> x = [1 2 3;4 5 6]; >>numel(x) ans = 6

4- max: return the maximum number in a vector or in the each column in a matrix:

Maximum of a Vector	Maximums in the Columns	Maximum in a Matrix
>> x = [1 3 5 7]; >>max(x) ans = 7	>> x = [1 2 3;4 5 6]; >> max(x) ans = 5 6	>> x = [1 2 3;4 5 6]; >> max(x(:)) ans = 6

5- min: return the minimum number in a vector or in the each column in a matrix:

Minimum of a Vector	Minimums in the Columns	Minimum in a Matrix
>> x = [1 3 5 7]; >>min(x) ans = 1	>> x = [1 2 3;4 5 6]; >> min(x) ans = 1 23	>> x = [1 2 3;4 5 6]; >> min(x(:)) ans = 1

6- mean(): return the average value of vector or the averages of matrix's columns.

Average of a Vector	Average of the Columns	Average of a Matrix
>> x = [1 3 5 7]; >>mean(x) ans = 4	>> x = [1 2 3;4 5 6]; >> mean(x) ans = 2.5000 3.5000 4.5000	>> x = [1 2 3;4 5 6]; >> mean(x(:)) ans = 3.5000

7- sum(): return the sum of the entries of a vector or the columns of a matrix.

Sum of a Vector	Sum of the Columns	Sum of a Matrix
>> x = [1 3 5 7]; >>sum(x) ans = 16	>> x = [1 2 3;4 5 6]; >> sum(x) ans = 5 79	>> x = [1 2 3;4 5 6]; >> sum(x(:)) ans = 21

8- clear: delete vector or matrix from memory.
Clear x; %clear x only from the memory
Clear (or clear all) clear all matrices and variable from the memory
9- whos: return the names and size and dimensions of the vectors and matrices in memory.
10- who: return the names of the vectors and matrix in memory.

14.Drawing Using MATLAB

There are standard function that used to draw in MATLAB:
1- plot(): this function plot linear line between two vectors:
plot(y) %plot vector y versus its index
plot(x_1,y_1,x_2,y_2,…,x_n,y_n) %plot y_n versus $x_n$vector
plot(x_1,y_1,linespec$_1$,x_2,y_2,LineSpec$_2$,…,x_n,y_n,LineSpec$_n$) %LineSpec(Line Specifier) is a character string made from one elementfrom any or all the following 3 columns:

Line Color	Meaning	Point Shape	Meaning	Line Shape	Meaning
B	Blue	.	Point	-	solid
G	Green	o	Circle	:	dotted
R	Red	x	x-mark	-.	dashdot
C	Cyan	+	Plus	--	dashed
M	Magenta	*	Star	(none)	no line
Y	Yellow	s	Square		
K	Black	d	Diamond		
W	White	v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	Pentagram		
		h	Hexagram		

For example, PLOT(X,Y,'c+:') plots a cyan dotted line with a plus at each data point;
PLOT(X,Y,'bd') plots blue diamond at each data point but does not draw any line.
ylabel('label'), xlabel('label'), title('title')

2- plot3(): displays a three-dimensional plot of a set of data points.
plot3(x,y,z) %plot vector y versus its index
plot3(x,y,z,LineSpec) %plot vector y versus its index
where the LineSpec is the same as in plot().

3- polar(): it use the polar coordinates to plot.
polar(theta,rho)
polar(theta,rho,LineSpec)
where the LineSpec is the same as in plot().

4- semilogx():creates a plot using a base 10 logarithmic scale for the *x*-axis and a linear scale for the *y*-axis.
semilogx (y) %plot vector y versus its index
semilogx (x_1,y_1,x_2,y_2,…,x_n,y_n) %plot y_n versus x_n vector
semilogx (x_1,y_1,*linespec*$_1$,x_2,y_2,*LineSpec*$_2$,…,x_n,y_n,*LineSpec*$_n$)
where the LineSpec is the same as in plot().

5- semilogy(): creates a plot using a base 10 logarithmic scale for the *y*-axis and a linear scale for the *x*-axis.
semilogy (y) %plot vector y versus its index
semilogy (x_1,y_1,x_2,y_2,…,x_n,y_n) %plot y_n versus x_n vector
semilogy (x_1,y_1,*linespec*$_1$,x_2,y_2,*LineSpec*$_2$,…,x_n,y_n,*LineSpec*$_n$)
where the LineSpec is the same as in plot().

6- loglog():creates a plot using a base 10 logarithmic scale for the*x*- and*y*-axis
loglog (y) %plot vector y versus its index
loglog (x_1,y_1,x_2,y_2,…,x_n,y_n) %plot y_n versus x_n vector
loglog (x_1,y_1,*linespec*$_1$,x_2,y_2,*LineSpec*$_2$,…,x_n,y_n,*LineSpec*$_n$)
where the LineSpec is the same as in plot().

7- bar() &barh(): plot a bar chart between two vectors, bar() plot it vertically and barh() plot it horizontally.
bar(y) %plot vector y versus its index
bar(x_1,y_1,x_2,y_2,…,x_n,y_n) %plot y_n versus x_n vector
barh(y) %plot vector y versus its index
barh(x_1,y_1,x_2,y_2,…,x_n,y_n) %plot y_n versus x_n vector

8- bar3() & bar3h(): plot a 3-dimentional bar chart between two vectors, bar3() plot it vertically and bar3h() plot it horizontally.
bar3(y) %plot vector y versus its index
bar3(x_1,y_1,x_2,y_2,…,x_n,y_n) %plot y_n versus x_n vector
bar3h(y) %plot vector y versus its index
bar3h(x_1,y_1,x_2,y_2,…,x_n,y_n) %plot y_n versus x_n vector

9- mesh() : create wireframe parametric surfaces specified by the matrix Z.
mesh(Z) %create 3D wireframe surface

10- surf() :create colored parametric surfaces specified by the matrix Z.
surf(Z) %create 3D shaded surface

15.Forming a Plot in MATLAB

The following function form the final view the plot

1- figure(): create figure graphics object.

2- xlabel('string'), ylabel('string'), zlabel('string'): label x-, y-, and z-axes.
xlabel('temperature'), ylabel('volume')

3- title('string'): add a title for a plot.
title('the relationship between temperature and volume')

4- grid(): to view the grid lines (on) or hide the grid lines.
grid on % appear the plot grids
grid off % hide the plot grids

5- box(); to close the 2D- or 3D-dimensions plots axes (on) or open it (off).
box on % close the plot boundaries
box off % open the plot boundaries

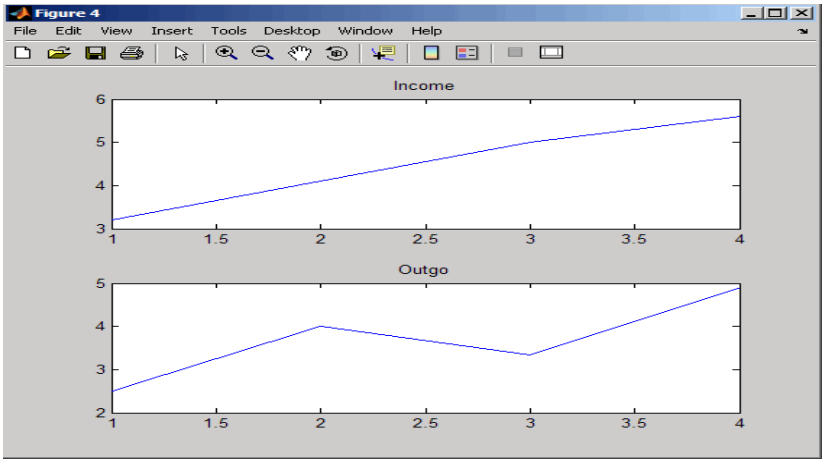
6- axis(): Axis scaling and appearance using one of the forms
axis([xmin,xmax,ymin,ymax,zmin,zmax]) %set new scales for the axis
v = axis %return the current axis
axis auto %set the axis to the default values
axis off %hide the axis
axis on %return back the axis

7- xlim(), ylim(), zlim(): set or query x-, y-, z-axis limits
xlim([xmin, xmax]) %set new scale for the axis
v = xlim() %return the current axis
xlim('auto') %set the axis to the default values

8- subplot(r,c,s): divides the current figure into rectangular panes that are numbered row wise.
where r number of rows and c number of columns and s the sequence of the subplot.
subplot(2,1,1), plot(income)
subplot(2,1,2), plot(outgo)

9- close(): close certain figure or all figures
close(2) %close the second figure according to their creation sequence

close or *close all* %close all figures



16. Logical Comparison and Combining Operators

The logical comparison operator returns false (0) if the logical comparison does not match or true if it exists (1 or nonzero number):

Logical Operator	Meaning
<	less than
< =	less or equal than
>	greater than
> =	greater or equal than
= = (differ than = assignment)	equal to
~ =	not equal
||	or operator
&&	and operator
~	not operator

Let x=2 and y=3 then

= =			
>>x = = y ans = 0	>>x ~ = y&& y > x ans = 1	>>x < = y || 7 ans = 1	>>x > = y ans = 0

17. Control Flow Statements

There are several statements that control the flow of executing program, these statements are:

1- For Loop Statement

For statement is used to repeat executing certain statement(s) for define number of times.

for x=initval:stepval:endval, statements, end

Positive Increment	Without Increment	Negative Increment	Double For Loop

>>x = 2; >>for lcv=1:2:10, y = x^lcv, end y=1 y=8 y=32 y=128 y=512	>>x = 2; >>for lcv=1:4 y = x/lcv end y=2 y=1 y=0.667 y=0.5	>>x = 0; >>for lcv=10:-1:6 y = x +lcv end y=10 y=19 y=27 y=34 y=40	>>for r=1:6 for c=1:4 H(r,c) = (r+c-1) end end >>H

2- While Loop statement

While statement is used to repeat executing certain statement(s) for undefined number of times until the condition becomes false.

while condition, statements, end

>>n = 0; >>a = 1e9; >>while 2^n<a n = n + 1; end, n n = 30	>>x = 10; >>while x>=0.5 x= x/2 end y=5 y=2.5 y=1.25 y=0.625	>>tank = 0; >>gallon = 4.5; >>times = 0; >>while tank <= 200 tank = tank + gallon; times = times + 1; end *tank =202.5 , times = 45*

3- If condition statement

Execute statement(s) if condition is true, otherwise it executes the else statement(s).

if condition, statements, end

OR

if condition, statements
else, statements, end

OR

if condition, statements
elseif condition, statements
elseif condition, statements
:
else, statements, end

>>X = input('input the value of X = '); >>if X>50 'Success' end >>if X<50 'Failed' end	>>X = input('input the value of X = '); >>if X>49 'Success' else 'Failed' end
>>Passed = 0; >>PassedScore = 0; >>Score = round(rand([1 50])*99); >>for lcv=1:50 if Score(lcv)>=50 Passed = Passed + 1; PassedScore = PassedScore + Score(lcv);	>>Passed = 0; Failed = 0; Fifty = 0; >>PassedScore = 0;FailedScore = 0; >>Score = round(rand([1 50])*99); >>for lcv=1:50 if Score(lcv)>50 Passed = Passed + 1; PassedScore = PassedScore+

```
        end
end
>>Passed
>>MeanPassedScore =…
PassedScore/Passed
```

```
Score(lcv);
        elseif Score(lcv) == 50
        Fifty = Fifty + 1;
        else
        Failed = Failed + 1;
        FailedScore = FailedScore+ Score(lcv);
        end
end
>>Passed, Failed, Fifty
>>MeanPassedScore =PassedScore/Passed
>>MeanFailedScore = FailedScore/Failed
```

4- Break Statement:

Break terminates the execution of a *for* or *while* loop. Statements in the loop that appear after the break statement are not executed.In nested loops, break exits only from the loop in which it occurs. Control passes to the statement that follows the end of that loop.

```
for u=1:100
   x=input('the value of x:');
   if x==0, break end
   y = 3*x^5 –10* x^3 +107* x
end
```

```
for u=1:10
 for v=1:10
  if u+v>10
  y = (20*u)/(v^2)
disp(v);
    else
  break
    end
  end
end
```

```
x = 5;
while x>1
 x = x – 0.5;
y = sin(x);
 if x==3, break end
end
```

5- Continue statement: Pass control to next iteration of for or while loop in which it appears, skipping any remaining statements in the body of the loop.

```
for u=1:10
   x=input('the value of x:');
   if (x/2)~=fix(x/2) , continue end
   y = 3*x^5 –10* x^3 +107* x
end
```

```
for u=1:10
  for v=1:10
     if u+v<10
        y = (20*u)/(v^2)
disp(v);
     else
continue
     end
  end
end
```

```
x = 5;
```

while x<500 if x/5==fix(x/5), continue end y = sin(x);x = x + 1;` end	

6- Switch-End:

The switch selection structure provides an alternative to using the if, elseif, and else commands. The advantage of the switch structure is that in some situations, it yields code that is morereadable.

switch expression
case test expression 1
commands
case{test expression 2, test expression 3}
commands
.
.
.
otherwise
commands
end

if choice == 1 x = -pi:0.01:pi; elseif choice == 2 x = -pi:0.01:pi; plot(x, sin(x)); end	switch choice case 1 x = -pi:0.01:pi; case 2 x = -pi:0.01:pi; plot(x, sin(x)); end
Interval = input('Input the interval value:'); switch interval < 1 case 1 Result = interval/10; case 0 Result = 0.1; end	x = input('input the distance in meter:'); units = input('input the unit to convert to: '); switch units case {'inch','in'} y = x*0.0254; case{'feet','ft'} y = x*0.3048; case{'meter','m'} y = x; case{'centimeter','cm'} y = x/100; case{'millimeter','mm'} y = x/1000; otherwise

	disp([’Unknown units: ’ units]) y = NaN; end