Lab 1
Introduction to MATLAB

1.1 Motivation:

MATLAB is a tool developed by ***MathWorks*** in order to solve mathematical problems handled by matrices, plot 2D and 3D graphs , create programs as scripting systems allowing users to develop/modify them.

1.2 Introduction and background theory:

1.2.1 User Interface:

The user interface is divided into four parts:

1- Command window
   - It is the main window used to communicate with MATLAB, where a user writes commands and wishes to execute them instantly.
   - We can use special characters with MATLAB:
     - ❖ To write comments about certain script or code: use ''%''.
     - ❖ To suppress printing results the command is followed by semicolon '';'.
     - ❖ To clear the command window ''clear screen'', type ''clc''.

2- M-file editor:
   - To write commands you wish to execute after you finish the whole program use M-files. They are scripts ''series of commands'' saved in mfile named ''.m''
   - To run the file: click the run button or press ''f5''.

   Notes :
     - ❖ Execution is done according to the orders of commands.
     - ❖ To stop a m-file in the middle of the execution: Press ''control +c'' to break execution.

3- The current directory:
   - Shows the path to the directory in which MATLAB is working. Ex: c:\mMATLAB7\work
   - It can be changed or browsed using the Current Directory window.

4- MATLAB workspace:
   - When defining a new variable or any data type in MATLAB, it is stored temporarily in the computer memory as long as MATLAB is opened.
   - It lists all the built variables during execution
   - To access the work space, use workspace browser in the command window:
     - ❖ To see list of current variables, type ***who***
     - ❖ To see list of current variables with information about them, type ***whos***

- ❖ To view the contents of a variable name,simply type its name in the command window. (*Note*: MATLAB is case-sensitive)
- ❖ To delete variables stored in the workspace in order to save memory, type ***clear.***

1.2.2 Variables representation in MATLAB:

Types of variables:

- Matrices
- Vectors (arrays): single row or single column matrix
- Scalars: single row and single column matrix
- Symbolic variable

In this lab, we will consider the first three types.

<u>1.3 Lab procedure:</u>

1.3.1 To create a matrix:

- Write a variable name
- Follow it by ''=''
- Fill in the values between square brackets []
- To separate rows: use spaces or semicolon

1.3.2 Matrix and array operations:

- Matrix operations deal with matrices as they are treated in math.
- Array operations are performed on element by element basis.
- (+ ,-): same for matrix and array operations
- (* , /, ^ , '): (multiplication, division, power , transpose),different for matrix &
- array operations
- (*,/): (multiplication, division) are the same in matrix and array operations if the
- multiplication & division is by a scalar value.
- Use ''.'' (dot operator) before operators for array operations
- Inv( ): Inversion is only available for matrix operations
- Operators priority when executing:
  - ❖ Transpose
  - ❖ Power
  - ❖ Multiplication
  - ❖ Division
  - ❖ Addition
  - ❖ Subtraction

1.3.3 Complex numbers:

- Complex numbers can be written as a=x+yi or a= x+yj
- To find the magnitude of a complex number : use abs(a)
- To find the phase of a complex number :use angle(a)
- To find the transpose of a complex number: use (.')

- To find the transpose conjugate of a complex number: use (')

Here are some examples:
1. Define a vector of four elements and perform the following operations:
-addition
-Subtraction
-multiplication (elements wise, matrix wise)
-Division

2. Define two 2*2 matrix to perform the following operations:
-addition
-subtraction
-division
-Multiplication
-Transpose
-Inversion
-Power

3. Define two complex numbers
-addition
-subtraction
-Magnitude
-Angle (phase)

<u>1.4 Conclusion & Home Work:</u>

<u>You will be given a number # corresponding to your lab section. The number is posted in lab.</u>

Evaluate the following matrices operations using mathematical calculations.
Compare your results with those obtained using MATLAB.

(1) A=[1 2 0;2 5 -1;4 10 0] ,B=#*[1 2 4; 2 5 10;0 -1 -1], Perform:
-addition, subtraction, division, transpose

(2) Given two matrices A=[1 2 3;4 5 6;7 8 9] and B=#*[9 8 7;6 5 4;3 2 1]
Perform:
-multiplication
-inversion

Turn in only the answer sheet, and only use the answer sheet.

Answer Sheet #=___ Name:_______________Section:____

LAB 1 (A) BELOW

Problem 1: A + #*B = [ ]

[ ]

[ ]

A - #*B = [ ]

[ ]

[ ]

A / #*B = [ ]

[ ]

[ ]

A transpose = [ ]

[ ]

[ ]

Problem 2: A*#*B = [ ]

[ ]

[ ]

Inverse of A = [ ]

[ ]

[ ]

LAB 1 (B) BELOW

x vs. y1,y2 in a separate window

all on same window, separate cells

all on same figure

Plot of R*#*D

Inverse of B =	[ ]	<u>Bonus assignment</u>
	[ ]	
	[ ]	

Lab 1 (Part B)
2D Graphics

1.1 Motivation:

MATLAB has the ability to draw 2-D and 3-D graphics. It is capable of drawing line plots, polar plots, contour plots, mesh plots and surface plots. In this section different plotting functions will be illustrated. Then, signal representation will be discussed.

1.2 Introduction and background theory:

1.2.1 Line plots:

Line plots are of great importance to our lab so it will be explained in fine details. The function `plot(x,y)` is used to draw the variable `y` versus the variable `x`.

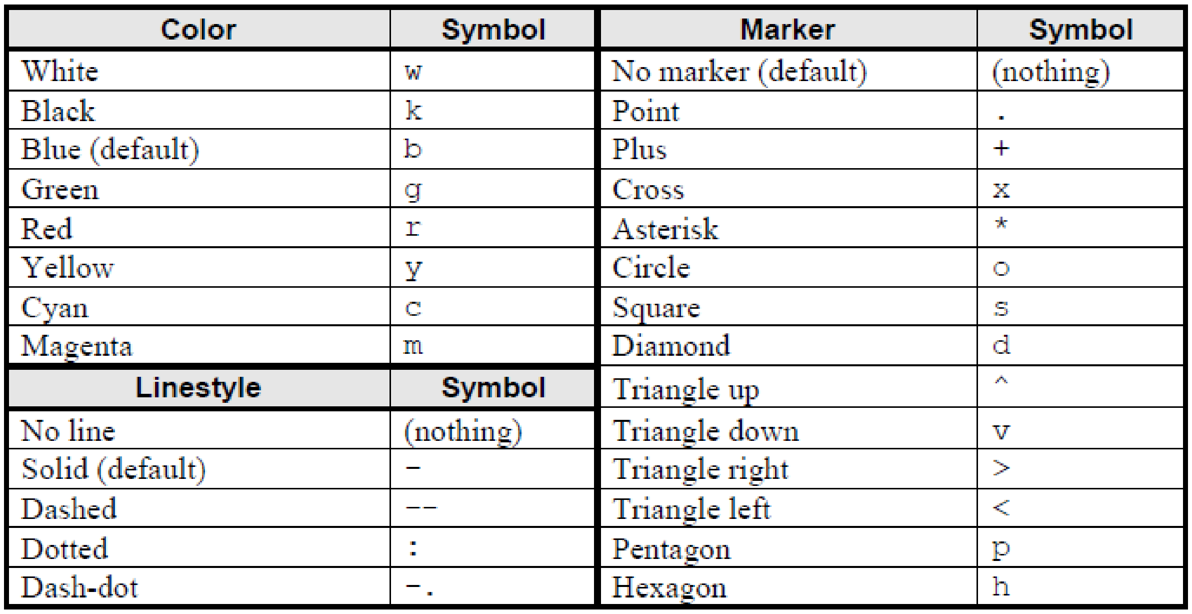
1.2.2 Line parameters:

Line parameters include the color, linestyle, and marker. They can be set for each plot using a string with single quotes following the variables x and y according to the coming tables.

To write the plot command including the line parameters:

- To specify all the parameters together, type plot(x,y,'color linestyle marker').

Here is a table for Line Parameters (Specifiers)



Color	Symbol	Marker	Symbol
White	w	No marker (default)	(nothing)
Black	k	Point	.
Blue (default)	b	Plus	+
Green	g	Cross	x
Red	r	Asterisk	*
Yellow	y	Circle	o
Cyan	c	Square	s
Magenta	m	Diamond	d
Linestyle	**Symbol**	Triangle up	^
No line	(nothing)	Triangle down	v
Solid (default)	-	Triangle right	>
Dashed	--	Triangle left	<
Dotted	:	Pentagon	p
Dash-dot	-.	Hexagon	h

1.2.3 Discrete Sequence Plots:

Discrete graphs are ones including a sequence of Dirac-Delta functions having magnitude shaping envelope of any continuous function. Such schemes are plotted using **stem**. Stem have several syntax forms in which it can be used:
To plot the data sequence in vector y versus the one in x, type **stem(x,y)**, this will produce plots with data sequences which are terminated with circle markers for the data value.
The syntax **stem(...,'filled')** produces a stem plot with filled markers.
The syntax **stem(...,'LineSpec')** uses linetypes specified for the stems and markers.

1.2.4 Graphics control functions:

1.2.4.1 Figures:

Figures are spaces that we can plot within. To create a figure:
Use the command **figure** to create a new figure with an index (figure number) that follows the largest index of an open figure.
The command **figure(n)** creates a new figure with the index of "**n**" (i.e. Figure No. N). But if this figure window exists (old) MATLAB uses it (i.e. makes it the active one).

1.2.4.2 Subplots:

The command subplot(m,n,i) or subplot(mni) divides the current figure window into mxn rectangular panes (subplot areas) that are numbered row-wise using the index i. Subsequent plots are plotted at the current pane (the active subplot) which has the index i. You can combine different ways for dividing or splitting subplot structures on the same plot according the representative manner that suits your requirements.

1.2.4.3 Axes scaling and appearance:

The command axis is used to define specific axis properties in the following syntax:
axis([x-min x-max y-min y-max]) sets the x and y axes scaling**.**

1.2.4.4 Grid Lines for 2-D & 3-D Plots:

Grid lines on the current graph can be turned on and off using grid on / grid off commands, or by using the grid command which inverts (flips) the grid lines status.

1.2.4.5 Adding Text:

MATLAB provides the following commands to add text to plots and subplots:

title('text')	adds title to the current plot / subplot
xlabel('text')	adds title to the current plot / subplot x-axis
ylabel('text')	adds title to the current plot / subplot y-axis
legend('txt1',txt2',..)	adds legend to the current plot / subplot
text(x,y,'text')	adds text at (x,y) coordinates in the current plot / subplot
gtext('text')	adds text to the current plot / subplot using the mouse

1.2.4.6 Holding Current Graphs:

When you plot a new graph on an existing one in MATLAB, the new graph replaces the old one. If you wish to plot the new graph while keeping the old one, in other words, plotting the new graph on the old, you can use the hold command before plotting the new graph.
The hold status can be turned on and off using the hold on / hold off commands.

1.3 Lab procedure:

1.3.1 Plotting a figure:

Example 1: given f=1, t= -1:0.00005:1, plot the following functions:

1- Sine wave.
2- Square wave.
3- Sawtooth wave.
4- Sinc wave.

1.3.2 Plotting multiple figures:

Example 2:

Given x=0:0.05:2, y1=exp(x), y2=exp(-x), y3=exp(-abs(x))
Plot x versus y1, y2, y3, y4 for the following cases:

1. Each in separate window:
2. All on the same window, but each on a separate cell:
3. All on the same figure:

1.3.3 Special functions:

We can plot different figures using special functions as well as shifting these figures along the x-axis.

1-Unit step function:

In MATLAB u(t) the unit step is called the Heaviside function, it was populatized by Oliver Heaviside a pioneering Electricial who among other things increased our understanding of the ionosphere (the Heaviside Layer) and developed Maxwell’s equations for electromagnetism into the form we know today.

Example 3:

Given t=-10:0.02:10

Plot: u(t), -u(t), u(-t), (2-u(t)), u(t-2), u(t+2).

2-Rectangular pulse:
It is a function used to generate a rectangular pulse whose dimensions are variable according to the user.
Syntax: y=rectpuls(t)
y=rectpuls(t,w)

3-Triangular pulse:
It is a function used to generate a triangular shape whose dimensions are variable according to the user.

Syntax: y=tripuls(t)
y=tripuls(t,w) generates a triangular pulse of width w.
Note: w is the full width.

Example 4:

Given t= -2:0.002:2,
1-draw a rectangle whose amplitude is 2 and its period is 5,
2-draw a triangle whose amplitude is 3 and its period is 7

1.4 conclusion and homework:

In this lab we learned how to represent any 2 dimension signal by using MATLAB. We learned also that decreasing the step size (the increment) defined in the range given for plotting is important to have accurate graphs.

- Given 0:0.001: 5, y1=cos(2x) , y2= # * sin(x)
  Plot x versus y1, y2 for the following cases:
  1. Each in a separate window.
  2. All on the same window, but each on a separate cell.
  3. All on the same figure.
- Given R=5 cos(x) , D=2 * # *sinc^2(x)…. notice this is sinc squared!
  Plot: R*D

Sinc is the special function (1/t) sin(t).

❖ Bonus assignment:

Plot a triangular pulse that is centered at 2 with amplitude # (without using a tripuls function) by using logical expressions.