Lab 1: Introduction to MATLAB

Name: Basil khowaja	Student ID: bk08432

1.1 Objective

Familiarization with MATLAB environment, its applications and basic commands.

In-Lab:

Task 1: Performing array operation.
Task 2: Generating matrices and performing mathematical operations on the matrices and use of transpose and matrix concatenation.
Task 3: Plotting 2D graphs using multi-color markers, subplot, hold on hold off, figure.

Post-Lab:

Task 4: Practicing array initialization.
Task 5: Generating matrices using Special matrices.

1.2 Background

MATLAB is a high-level technical computing language equipped with a user-friendly interface. Its name stems from the words **MATrix** and **LABoratory** as it is based on the use of matrices. It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

MATLAB is an extremely powerful tool useful for scientists and engineers from various disciplines. For example, MATLAB can be used in a wide range of applications, such as telecommunications, signal and image processing, control, mathematics, financial modelling, bioengineering, aeronautics, and many more.

1.3 Introduction

This lab introduces us to MATLAB environment. We will learn how to use some of the MATLAB commands to perform mathematical computations, and MATRIX operations, methods of initializing arrays, and plotting signals.

A guide to Introduction to MATLAB-(Chapter 1) is shared with you that document will be helpful in getting hands on basic MATLAB Programming skills.

1.4 Starting MATLAB

To access the ONLINE Version of MATLAB

- Please use the following website to access MATLAB Online on your browser https://matlab.mathworks.com/
- It is mandatory to have a math work account (based on your official Habib university email address)
- Personal math work accounts will **not** work for MATLAB Online

How to create Mathworks Account

- Go to **https://www.mathworks.com/login**
- Click on the link "Create Account."
- On the create account page, complete all required fields (Note: Make sure you enter your Habib University email address.) Click "Create" when you're done.
- A page will be displayed with steps you'll need to perform (check for an email from MATLAB, click on the link in the email) so that your email address can be verified. Important: Do not close this window until you've completed these steps.
- Click "Continue" after you have completed the steps to verify your email address. You will be returned to the MathWorks home page, and will be logged in to your new MathWorks account.

Dedicated Portal for download and online training

MATLAB has created a dedicated portal for our university. From there, you will be able to download the latest software, and a license will be automatically assigned to your math work account based on the HU email address.
https://www.mathworks.com/academia/tah-portal/habib-university-31522703.html

Toolboxes to be installed are:

- Simulink
- Signal Processing Toolbox
- Symbolic Math Toolbox
- Image Processing Toolbox
- Image Acquisition Toolbox

1.4.1 The Environment

The major tools accessible are:

The Command Window: The Command window is the main window of the program. The commands are typed at the command prompt indicated by the prompt (>>) and the obtained results also appear in the command window.
The Command History: In the Command History window one can see all the previously executed commands.
The Workspace: Explores data that you create or import from files.

The Current Directory: Moreover, at the lower left part of the figure we notice the Current Directory. The current directory specifies the active folder of MATLAB. By default, the current Directory is the folder work.

The Help Browser: Displays the help text for the functionality specified by name.

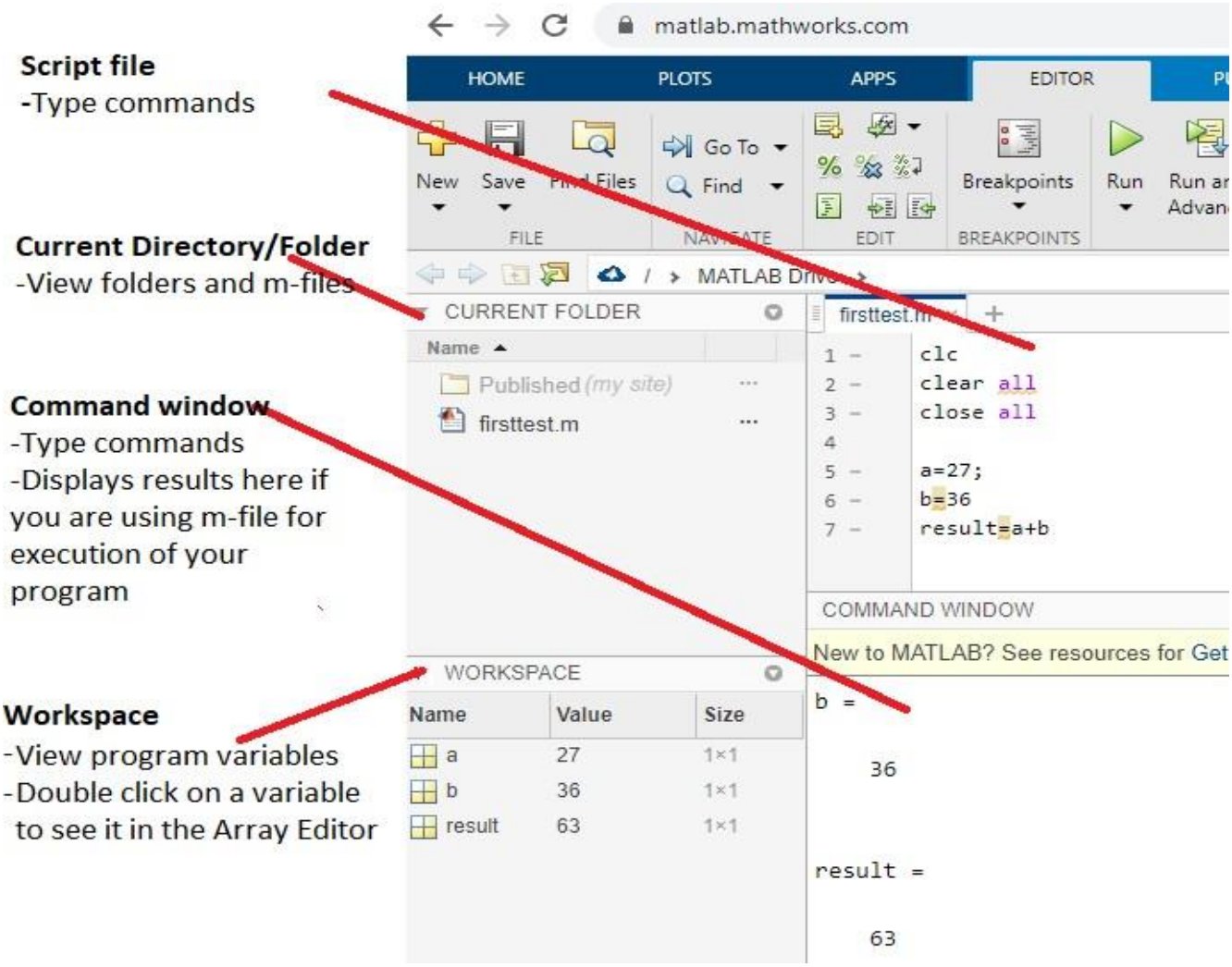


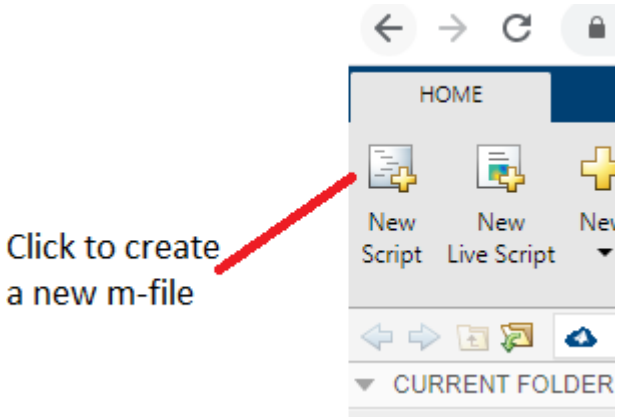
Figure 1: MATLAB IDLE.

There are two possible ways to work with MATLAB.

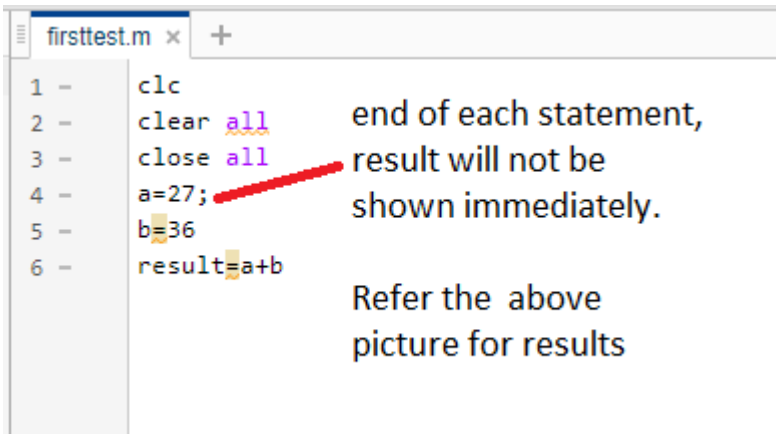
1. **Command Window:**
   The first is through the command window where the user types and executes the commands one by one. *Statement once executed cannot be edited you need to retype it if want to make any edits.*
2. **M-file:**
   In order to write many commands that are executed all together, the program must be written in a text editor. In this editor, one can type all the needed commands to form a program, save the program, and execute it any time he or she wants. The text files are called M-files due to their extension which is ".m" M-files can be edited.
   There are two categories of M-files: the Scripts and the Functions (We will talk about the functions later).

A script file is executed either directly from the editor window from the menu Debug/Run or by typing the script's name at the command prompt. In any case, the script file must be saved in the Current Directory.

For the **online version**, every time you edit the script file you need to save it before you execute/run your program.



An example code written in an .m file:



1.4.2 Comments

The symbol **%** is used to insert comments. Anything written to the right of **%** is considered comment and is not taken into account from MATLAB.

For multi-line comments

%{This is a multiline comment-**%}**

Try 1: Create your Mathworks account, create a script file and execute

```
% Program to understand the use of script file
clc
close all
clear all
% Everything written at the right of percent symbol is a comment and is not
```

```
taken into account
a=212;
b=255;
c=a+b
```

Click on save, give it a name and now run the file. Comment over your observations in the box below:

The first command clc which actually clears the command window then when the second command close all executes it closes all open figure windows. After which clear all command clears all variables and their values from workspace window. Then each variable is assigned the value which appears at the right hand side of the window (a,b and c) when the whole code is executed.

Variables in MATLAB

All the variables are created with double precision unless specified and they are matrices. After these statements, the variables are 1x1 matrices with double precision.

1.4.3 Pre-defined constant values in MATLAB

There are many predefined values in MATLAB, go to Section 1.3.5 of Chapter 1-Introduction to MATLAB for the complete list. Some of them are presented below:

Constant	Value
pi	π
i, j	$i, \sqrt{-1}$
Inf	∞
NaN	Not a number

1.4.4 Relational and Logical Operators

Relational operators perform element-by-element comparisons between two arrays.
They return a logical array of the same size, with elements set to true (1) where the relation is true, and elements set to false (0) where it is not.

SYMBOL	INTERPRETATION
==	Equal to
<	Smaller than
>	Greater than
<=	Smaller than or equal to
>=	Greater than or equal to
|	OR operator
&	AND operator

1.5 Vectors and Matrices

MATLAB (named after MATrix LABoratory) is a programming tool specialized for working with matrices (or arrays). Even the scalars are considered as matrices of size 1 x 1. In this section, we introduce a special case of matrices, the vectors.

1.5.1 Row Vector

To define a row vector, the elements of the vector are given into square brackets [. . .]. Spaces or commas are inserted between the elements. A vector element is specified by its index. The index numbering starts from 1, namely, the index of the first element of a vector is 1. In order to refer to a position in a vector, parentheses are used.

1.5.2 Vector/Matrix Index

The matrix indices begin from 1 (not 0(as in C)). The matrix indices must be positive integer.

Try 2: Type the following commands one by one in the ***command window*** and see the results give reason in case of error.

a) a=[1 2 3 4 5] % row vector a of 5 elements
b) a(3) % The third element of vector b, where 3 is the index number.

State your results and observations:
When we execute the first command the array 'a' gets displayed and its order is also shown in the result which is 1x5.
When we execute both commands together then the third element of array 'a' is displayed which is 3.

1.5.3 Commands length/size

In order to compute the number of elements of a vector, the appropriate command is the command length. A similar command is the command size. Command size is usually employed to compute the dimensions of a matrix. However, vectors are special case of matrices; thus, using the command size to compute the length of a vector is completely valid.

Try 3: Type the following commands one by one in the command window

(a) length(a) % Returns number of elements of vector a.
(b) size(a) %Vector a consists of 1 row and 5 columns

State your results and observations:

When we execute the first command length(a) it shows us the number of digits in array 'a' or number of elements inside the array.
When we execute the second command second command it shows us the dimension of the array 'a'

It is also possible to edit only one specific element of a vector by referring to the position (index) of the vector e.g a(2)=3, this will replace the 2nd element of vector a by 3.
Try 4: For a given vector $f = [1\ 2\ 3\ 4\ 5\ 6]$ replace the 4th element by digit 60. Vector f should now have values **f= 1 2 3 60 5 6.**

Code and results:
Code:

```
%try 4
f=[1 2 3 4 5 6]
f(4)=60
```

results:

```
f = 1×6
     1     2     3     4     5     6

f = 1×6
     1     2     3    60     5     6
```

Column Vectors

The way to define a column vector is quite similar. In this case, the elements are also inserted between square brackets. However, the elements are not separated by comma but by a semicolon (**;**). A second way to define a column vector is given now, after opening the square bracket and inserting the first element pressing the Enter key moves us to the next line where the second element is inserted. This process continues for all vector elements. Closing the square brackets and pressing Enter provides us a column vector.

1.6 Mathematical Operators

The most common arithmetic operations in MATLAB are:

OPERATION	SYMBOL
Addition	+
Subtraction	-
Multiplication	*
Division	/
Power	^
Complex conjugate transpose	‘

1.6.1 Element by element Operation

OPERATION	SYMBOL
Element by element multiplication	.*
Element by element division	./

Element by element power	.^

1.6.2 Array Operation

MATLAB has two different types of arithmetic operations: array operations and matrix operation

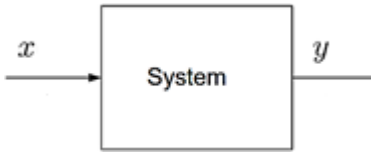
Matrix operations follow the rules of linear algebra. By contrast, array operations execute element by element operations and support multidimensional arrays. The period character (.) distinguishes the array operations from the matrix operations. However, since the matrix and array operations are the same for addition and subtraction, the character pairs . + and . − are unnecessary. Array operations execute element by element operations on corresponding elements of vectors, matrices, and multidimensional arrays. If the operands have the same size, then each element in the first operand gets matched up with the element in the same location in the second operand. If the operands have compatible sizes, then each input is implicitly expanded as needed to match the size of the other.

1.6.3 Array Multiplication

A.*B multiplies arrays A and B by multiplying corresponding elements. The sizes of A and B must be the same or be compatible.
If the sizes of A and B are compatible, then the two arrays implicitly expand to match each other. For example, if one of A or B is a scalar, then the scalar is combined with each element of the other array.

Task 1 Array Operation

Before starting this task open the MATLAB Chapter 1 guide document*Refer section 1.6.
There is a system "S" which gives takes an input signal & processes it in such a way that it gives the square of the incoming signal:



Let x=[2,4,5,9] be the input, using MATLAB the desired output 'y' for system "S" can be obtained by following two methods:

i. Element by element multiplication operation –
```
>> y=x.*x
```

ii. Element by element Power operation
```
>> y=x.^2
```

State your observations: When we execute both I and ii commands then in return we get the matrix in which each entry is the square of the each entry of matrix 'x'

1.7 Matrix

In this section, we introduce how a matrix (or array) is defined in MATLAB. The use of square brackets is necessary. The first way is to type the elements of the first row separated by comas or spaces and then insert a semicolon (which corresponds in changing a row), type the elements of the second row, and so on.

A second way is to type the elements of the first row, then press Enter to move to the second row, etc. Matrix will also be written within the square brackets with space in between each element of a row (or you can say a new column) and a semicolon is used for a new row.

In MATLAB, the following command:

$$x = [1\ 2\ 3;\ 5\ 1\ 4;\ 3\ 2\ -1]$$

gives a three-by-three matrix as follows:

```
x = 1 2 3
    5 1 4
    3 2 -1
```

1.7.1 Matrix Multiplication

Note: You can only multiply matrices if their dimensions are compatible, which means the number of columns in the first matrix is same as the number of rows in the second matrix. If A is an $m\ x\ n$ matrix and B is an $n\ x\ p$ matrix, they could be multiplied together to produce an $m\ x\ n$ matrix C.

Task 2(a) Matrix Multiplication

Now onwards use a script file to write your code. Before starting this task open the MATLAB Chapter 1 guide document*Refer section 1.7 specifically 1.7.4 and 1.7.9.

For the given matrices:

$$a = \begin{bmatrix} 2 & 5 & 3 \\ 1 & 1 & 0 \\ 9 & 6 & -1 \end{bmatrix}, b = \begin{bmatrix} -1 \\ 6 \\ 9 \end{bmatrix} \text{ and } c = \begin{bmatrix} 3 & -2 & 4 \end{bmatrix}$$

Perform the following arithmetic operations:

i) $x = a \times b$
ii) $y = a + 4$
iii) $z = b \times a$
iv) Take square of every element of array c.

Note: part iii) exhibits an error. Discuss the reason behind it and try to resolve the issue.

part iv) requires you to use the element-by-element operation as discussed in Task 1.

```
x = 3×1
     55
      5
     18

y = 3×3
      6     9     7
      5     5     4
     13    10     3

m = 1×3
      9     4    16
```

In task 3 there was error, because the dimension of matrix b and a is not matching the matrix multiplication condition. Which means that b*a can't be carried out. For resolving this we need to correct the dimensions

Transpose

The transpose of a matrix A (usually denoted by A^T) is a new matrix whose rows are the columns of the original. (This makes the columns of the new matrix the rows of the original).

The transpose matrix can be used to convert a row vector into a column vector and vice-versa. **Note:** In case of complex numbers transpose, function does not affect the sign of the imaginary parts.

Task 2(b)

Click on **help** icon **(?)** and type transpose in search bar **or** type transpose in search documentation.

i) Find the transpose of matrix $\boldsymbol{a} = [\mathbf{3} \quad \mathbf{1} \quad \mathbf{4}]$ $\boldsymbol{and}$ $\boldsymbol{b} = [\mathbf{4} \quad \mathbf{2} \quad \mathbf{3}]$

ii) Find the **inner product (dot product)** of a and b. This can be done by taking product of a and b^T or b and a^T. Perform the following operation and comment on it.
   a. ab^T
   b. ba^T

iii) Find the **outer product** of a and b. This can be done by taking product of b^T and a or a^T. and b. Perform the following operation and comment on it.
   a. b^Ta
   b. a^Tb

I)

```
a = 1×3
     3     1     4

at = 3×1
     3
     1
     4

b = 1×3
     4     2     3

bt = 3×1
     4
     2
     3
```

ii and iii)

```
c = 26
d = 26
z = 3x3
    12     4    16
     6     2     8
     9     3    12
u = 3x3
    12     6     9
     4     2     3
    16     8    12
```

Matrix Concatenation

Matrix concatenation is the process of joining one or more matrices to make a new matrix. To concatenate two matrices, they must have compatible sizes. In other words, when you concatenate matrices horizontally, they must have the same number of rows. When you concatenate them vertically, they must have the same number of columns.

Try 5: Type the following in the script file and see the results:

```
A= [1 2 3]
B= [2 3 6]
C= [A B]
```

State observations: when these commands are executed it concatenates the matrix A and B into a single matrix C.

1.7.2 More methods for initializing vectors and matrices

Consider more generally the array constructor pattern:

a. array = start : step : stop

```
k = 2 : -0.5 : -1
k = 2   1.5    1    0.5    0    -0.5    -1
```

b. array = start : stop (when the step is not defined, then MATLAB by default uses 1.)

```
t = 1:10
t = 1   2    3    4    5    6    7    8    9    10
```

c. B = [1:4; 5:8] gives

```
B =
     1 2 3 4
     5 6 7 8
```

1.7.3 Special Matrices

Function	Description
zeros(m,n)	Returns m-by-n matrix with all the elements having value zero
ones(m,n)	Returns m-by-n matrix with all the elements having value one
rand(n) or rand(m,n)	Uniformly distributed random elements
eye(n)	Returns n-by-n identity matrix

1.8 Plotting Signals in MATLAB

MATLAB is a very reliable and power full tool for plotting. A graph is constructed as a set of points in two or three dimensions. These points are typically connected with a solid line. Commonly a graph of a function is required. However, in MATLAB a plot is done using the vectors and matrices not functions.

Commands:

plot(X,Y) %This MATLAB function creates a 2-D line plot of the data in Y versus the corresponding values in X, where length of both the vectors should be same

title % adds the specified title

xlabel % adds label to the x-axis in a plot

ylabel % adds label to the y-axis in a plot

subplot(m,n,p) % This MATLAB function divides the current figure into an m-by-n grid and creates axes in the position specified by p. where m=row, c=columns and p=plot number

suptitle[1] % Super title, when we make more than one plots in a figure and want to add a main or super title then this command is used

plot(t, x, 'LineWidth',2) %linewidth defines the thickness of the signal plotted

grid on % adds grid lines to the current axes or chart.

grid off % grid off removes all grid lines from the current axes or chart.

[1] In some versions this command does not work

hold on % retains the current plot and certain axes properties so that subsequent graphing commands add to the existing graph.

hold off % resets axes properties to their defaults, hold off is the default.

legend % to label data series plotted on a graph

axis(limits) % specifies the limits for the current axes

figure % creates a new figure window using default property values. If there are more than one figures and we want to see the results at once of all then, after 1st figure before plotting the upcoming figures, we use this command (i.e., before the **plot** command).

It is possible to create a graph using colors, symbols used to draw the points, and type of line that connects the points of your choice. This is achieved by applying one more input argument to the command Plot The new argument is a series of special character given in single quotes. The available special characters are presented in the table below.

Data markers†		Line types		Colors	
Dot (·)	·	Solid line	-	Black	k
Asterisk (*)	*	Dashed line	- -	Blue	b
Cross (×)	×	Dash-dotted line	-.	Cyan	c
Circle (o)	o	Dotted line	:	Green	g
Plus sign (+)	+			Magenta	m
Square (□)	s			Red	r
Diamond (◇)	d			White	w
Five-pointed star (★)	p			Yellow	y

†Other data markers are available. Search for "markers" in MATLAB Help.

1.8.1 Plotting in Two Dimensions

Suppose that we want to plot a function (x), where 'x' is an independent variable. The procedure to plot (x) is as follows:
1. Vector x is created. Such as $a \leq x \leq b$, where a and b are scalars.
2. The function will be plotted over the interval $[a, b]$
3. Create the vector y, which is of the same length as x. **The two vectors must have an equal number of elements.**
4. The value of each element of y is the value (x) of calculated for each value of x.
5. Finally, the function $y(x)$ is plotted by typing **plot(x,y)**

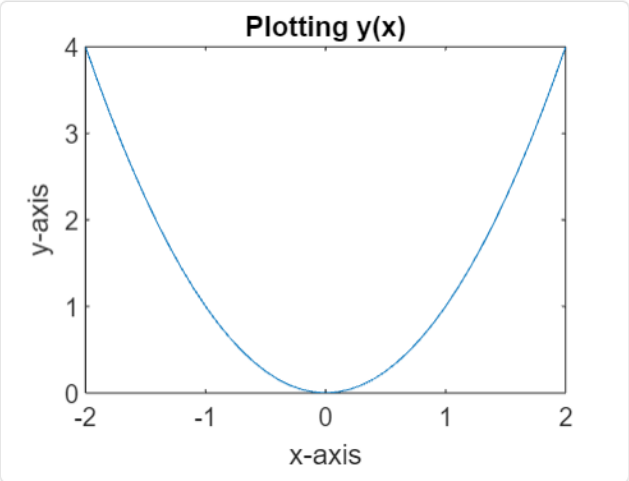
Try 6: Plotting the function

$$(x) = x^2, -2 \leq x \leq 2$$

Following is the MATLAB code for plotting the above function. Run the code, observe and attach the results. **Note:** Copy paste the code on notepad first and then copy from notepad and paste on

the script file, if still some error comes up then look for the apostrophe error ‘. You can remove this error by rewriting the command where ever the error appears.

Code:	Attach Graph/Plot here:

x=-2:0.1:2;
%start_step:step_size:end_step
y = x.^2; %{ Element wise multiplication operation %}
plot(x, y)
title('Plotting y(x)');
xlabel('x-axis');
ylabel(‘y-axis’);
	

1.8.2 Selection of time vector for plotting functions

Proper definition of time vector plays an important part in a good visualization of functions. When plotting sinusoidal and exponential signals, we can follow some rules for a proper graphical representation of these functions.

Before going into that let’s learn few terms,

Time vector is defined as: $t = t_0 : dt : t_{max}$
$where\ t_0 =$ start time,
dt = time step size,
$t_{max} =$ end time

Start time: This is the time where you want your signal to start from, if a signal starts from 0 seconds, then the start time is $t_0 = 0$.

Signal Duration: The duration of the signal can be decided by the fundamental frequency f_0 of the wave, if four cycles are required then t_{max}may be selected as follows:

$$t_{max} = \frac{4}{f_0}$$

.

Step size (dt): The step size can be selected by following some rule of thumb

Rule of Thumb for Sinusoidal function:

Suppose a sinusoidal function is represented as

$$A\sin(2\pi f_0 t + \phi)$$

here A is the amplitude, ϕ is the phase angle in radians, and f_0 is fundamental frequency. A reasonable time step dt may be obtained as

where the parameter k is some integer number between 10 and 50.

Note: You will learn the reasoning behind this thumb rule when you will be taught the concept of sampling in the course

Rule of Thumb for Exponential function:

Let's suppose we have a function $y = e^{-at}$, what should be the step size to visualize the exponential function properly? Rule of thumb is $dt = 1/3a$ and $t_{max} = 8/a$

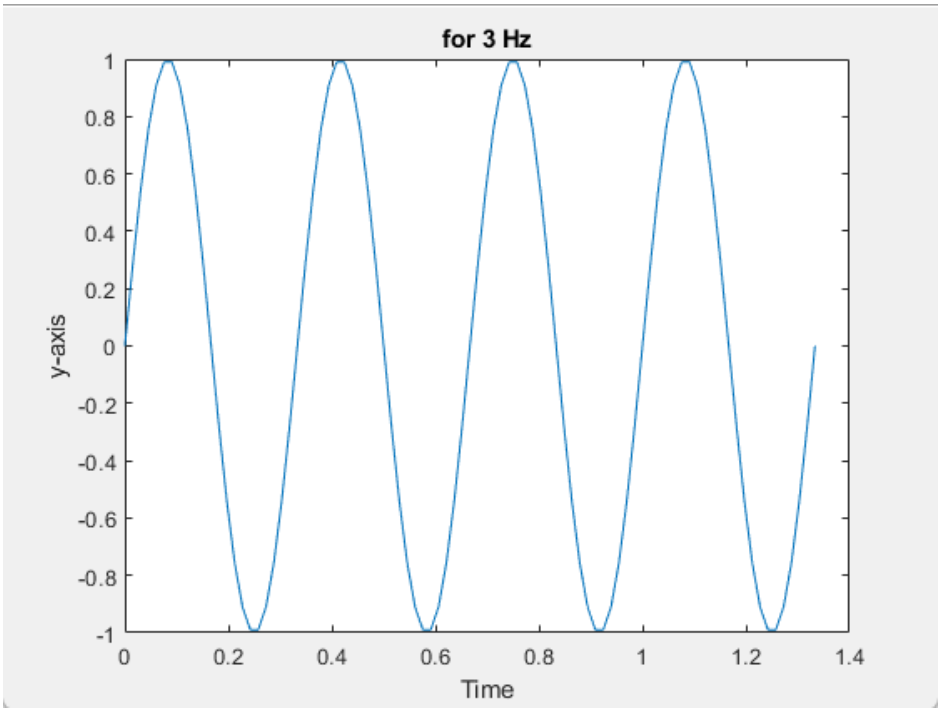
Task 3:

(i) Plot a sinusoidal function $x = \sin(2\pi f_0 t)$, for two different values of f_0, once for $f_0 = 3$ Hz, and the other for $f_0 = 300$ Hz, such that three cycles of the sinusoidal wave are plotted for both cases. You define time as **t = t0 : dt : tmax**. You need to obtain the value of **dt** and **tmax** for both cases.
Obtain separate plots for both cases with proper x, y label and plot title of the figure.

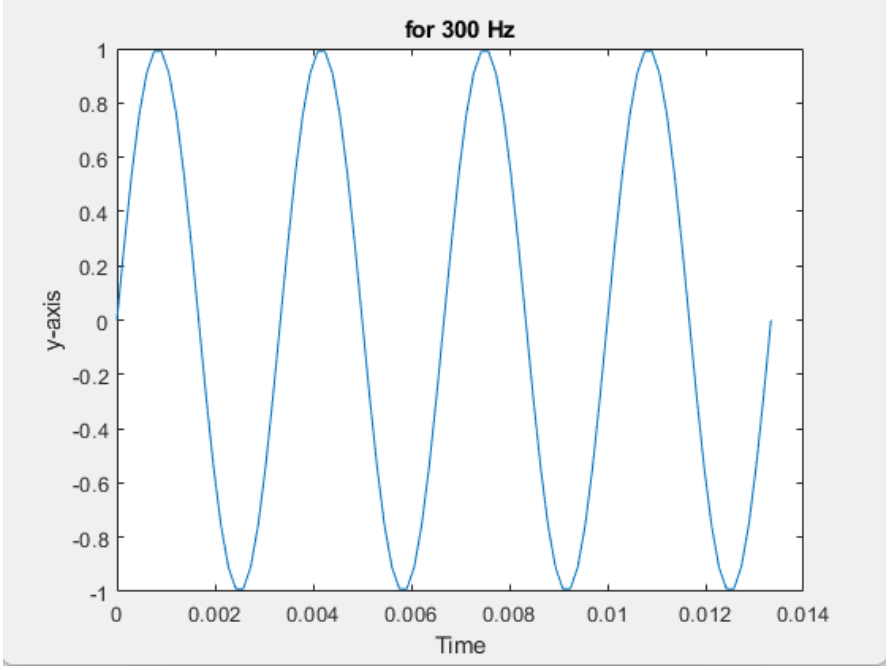
(ii) Plot a rising exponential function $y = 1 - e^{-at}$ for two different values of a, once for a = 0.01 and the other for a = 100. You define time as **t = t0 : dt : tmax**. You need to obtain the value of **dt** and **tmax** for both cases.
Obtain separate plots for both cases with proper x and y labels and title of the figure.

i)

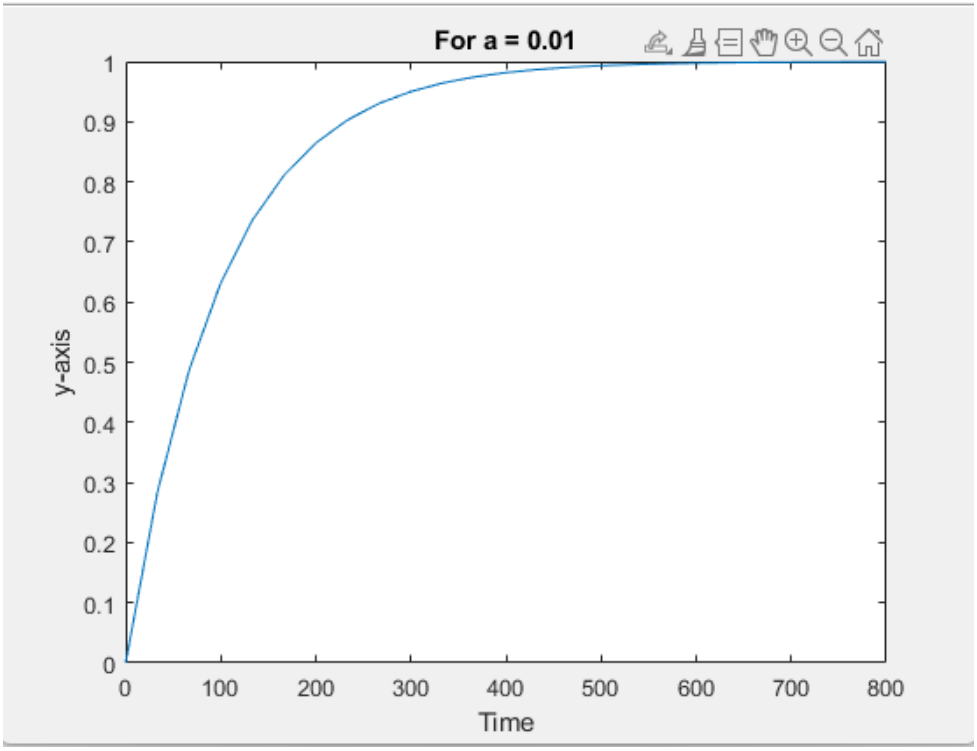
for fo=3hz:



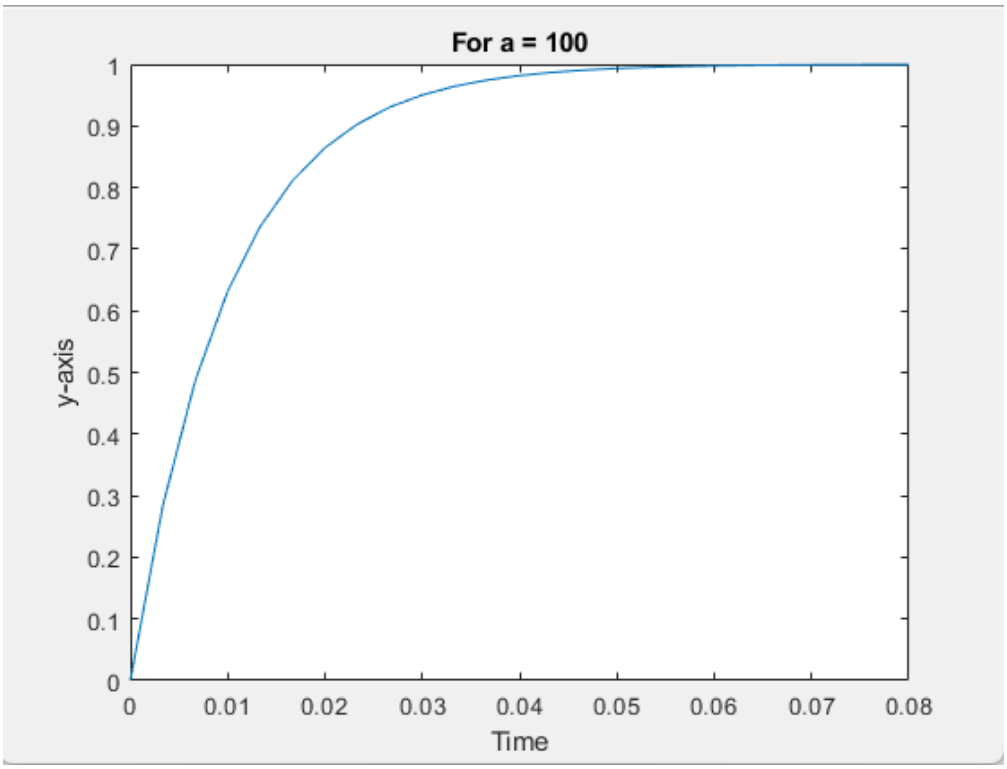
For fo=300hz



ii) For a= 0.01



For a =100:


For a = 100
y-axis
Time
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08

Lab 1: Introduction to MATLAB

Habib University
EE/CE-252L/251L Signal and Systems

Name:_____ Student ID.: _____

Marks distribution:

		LR2	**LR5**	**LR9**	**AR4**
In-Lab	**Task 1**		8	12	10
	Task 2	8	8		
	Task 3	4	8		
Post Lab	**Task 4**		12		
	Task 5	6	4		
Max Marks = 80		18	40	12	10

Marks obtained:

		LR2	**LR5**	**LR9**	**AR4**
In-Lab	**Task 1**				
	Task 2				
	Task 3				
Post Lab	**Task 4**				
	Task 5				
Max Obtained =					