

Name: AHSAN tariq

Roll No: 20-CSE-26

Session: 20-2024

(LAB MANUAL Signals And systems)

Submitted To:

Engr. Jawad masood

Computer System Engineering

ISLAMIA UNIVERSITY OF BAHAWALPUR

The Islamia University of Bahawalpur

**U**niversity **C**ollege of **E**ngineering **& T**echnology **D**epartment of **C**omputer **S**ystem **E**ngineering

|  |  |  |
| --- | --- | --- |
| **LAB MANUAL** | **SIGNALS AND SYSTEMS EE-311** | **4th Semester** |

**LAB EXPERIMENT # 01**

**Introduction to Matlab Programming**

|  |  |
| --- | --- |
| **Student Name: Ahsan Tariq** | **Roll No: 20-CSE-26** |
| **Lab Instructor Signatures:** | **Date:** |

## OBJECTIVES:

The purpose of this lab is to familiarize with basic MATLAB functionality. The students will also learn how vectors and matrices are defined and manipulated in MATLAB.

# The MATLAB Working Environment:

## The MATLAB Desktop:

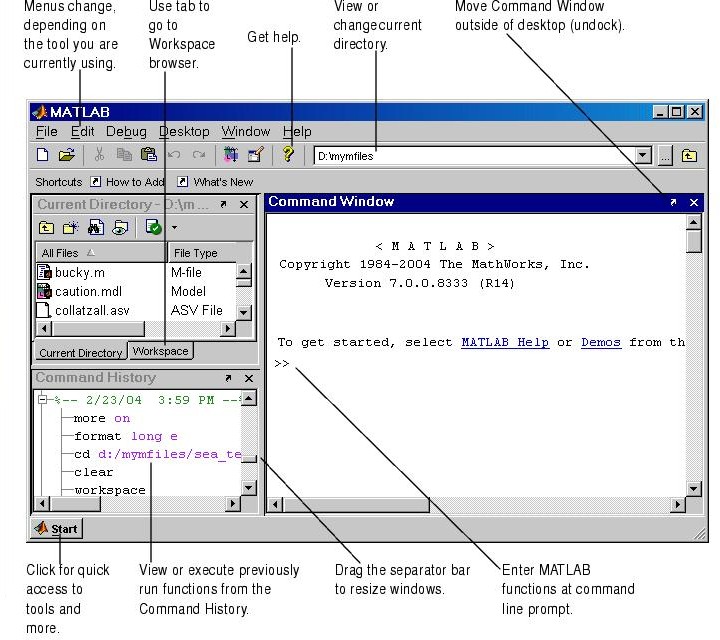
The MATLAB desktop is the main MATLAB application window. In Figure 1, the desktop contains five sub-windows: the Command Window, the Workspace Browser, the Current Directory Window, the Command History Window, and one or more Figure Windows, which are shown only when the user displays graphics or plots.

The ***Command Window*** is, where the user types MATLAB commands and expressions at the prompt (>>) and where the outputs of those commands are displayed.

MATLAB defines the workspace as the set of variables that the user creates in work session. The ***Workspace Browser*** shows those variables and some information about them.

The ***Current Directory*** tab above the Workspace tab shows the contents of the current directory, whose path is shown in the Current Directory Window. By Clicking on the button to the right of the window allows the user to change the current directory. MATLAB uses a search path to find M-files and other MATLAB. Any file run in MATLAB must reside in the current directory or in the directory that is on the search path. The easiest way to see which directories are on the search path, or to add or modify a search path is to select **Set Path** from the **File** menu on the desktop, and then use the **Set Path** dialog box.

The ***Command History Window*** contains a record of the commands a user has entered in the Command Window, including both current and previous MATLAB sessions. Previously entered MATLAB commands can be selected and re-executed from the Command History Window by right clicking on a command. This action launches a menu from which to select various options in addition to executing the commands. This is a useful feature when experimenting with various commands in a work session.



## Getting Started:

To run MATLAB simply double-click the MATLAB icon on the desktop or find the MATLAB command in the start menu. This will open a MATLAB command window, which displays a prompt “>>”. The commands are typed at this prompt. The “>>” is called the MATLAB prompt. There is no need to type that part. Explore the built-in demos by typing demo.

## MATLAB BASICS:

**1.31 Getting Help:**

MATLAB provides an online help system accessible by using the **help** command. For example, to get information about the function **size**, enter the following:

>> help size

There also is a help desk (formatted in HTML for viewing from a Web browser) with useful introductory material. It is accessible from the Help menu. If you have no prior experience with MATLAB, see the topic “Getting Started” in the help desk. Spend some time with this. You can find in the help desk all the information you need to carry out the exercises.

## Important Commands :

1. **Variables:**

Pwd Cd

Lookfor help

MATLAB does not require the variables to be declared before their use. To assign value to a variable equal “=” sign is used. To set *x* equal to 2 write at command prompt.

>> x = 2

Matlab replies with x =

2

Type 2^4 at the command prompt. Output

ans =16

where *ans* is also a MATLAB variable.

* 1. **MATLAB as Calculator** P=3\*(sqrt(5)-1)/(sqrt(5)+1)^2-1 Q=log(exp(3))

R=log10(exp(3)) S=(sin(pi/6))^2+(cos(pi/3))^2 T=(1+3i)/(1-3i)

## Techniques for rounding off variables:

A=[-2.33 2.66] ;

fix (A) ; floor (A) ; ceil (A) ; round(A) B = [1.52 1.67]

C= rem (A,B)

## Suppressing output:

When we type a command, function, or variable name, its value appears on the screen. Sometimes it is inconvenient if you do not want to display the output at every step of your program. The output display of the value can be suppressed by putting the semicolon at the end of command or variable name. If the semicolon is not present at the end of a command line, MATLAB outputs the contents of the operation on the screen.

## Clearing the variables:

The command

**>> clear x**

The *clc* command clears the variable x, from the workspace. This is useful when we want to reclaim system memory. Similarly, “clear all” and “close all” commands are used to clear all previous variables in memory and close all figure windows opened in MATLAB desktop.

## Vectors:

MATLAB simplest mathematical objects are scalars, vectors, and matrices. There are two types of vectors supported in MATLAB i.e., row vectors and column vectors.

The simplest way to create a row vector is by using concatenation brackets or square brackets. Various values in a row vector are separated by space or comma. Whereas in column vector various values are separated by semicolon.

**>> x = [1 2 3 4]**

**>> y = [1; 2; 3; 4]**

This will create a row vector x and column vector y.

**Question 1**: Initialize a row vector x1 and column vector y1 with 3 elements each.

>>

>>

To convert a row vector into column vector and vice versa transpose operator **.’** is used.

**>> x1 = x.’**

**>> y1 = y.’**

This will create a column vector x and row vector y. To see the dimension of x and y variables, type

>>whos

The *whos* command provides information on existing variables. Try who command.

MATLAB does not allow arguments of vectors or matrices to be zero or negative. For instance, if we want the first entry of the vector y we need to type:

>>y(1)

It will return the first element of vector y.

What will happen if the following command is entered? Write the output message.

>>y(0)

## ANS:

To create a vector corresponding to a sequence of numbers (in this case integers) there are different approaches, as follows:

## Method 1:

>>n = 0:10

It will create a row vector with entries 0 to 10 increased by 1. Write down the output.

n =

Columns 1 through 10

0 1 2 3 4 5 6 7 8 9

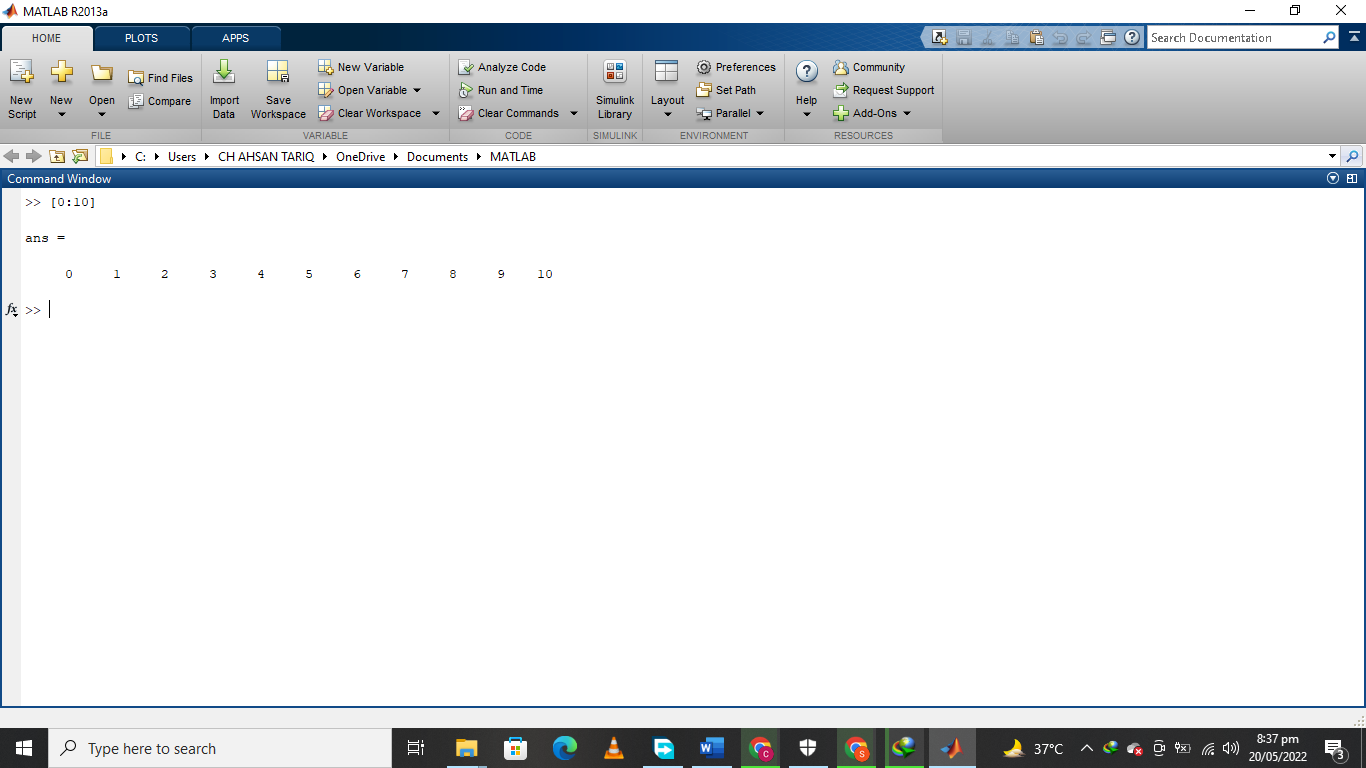
Column 11

10

## Method 2:

>>[0:10]

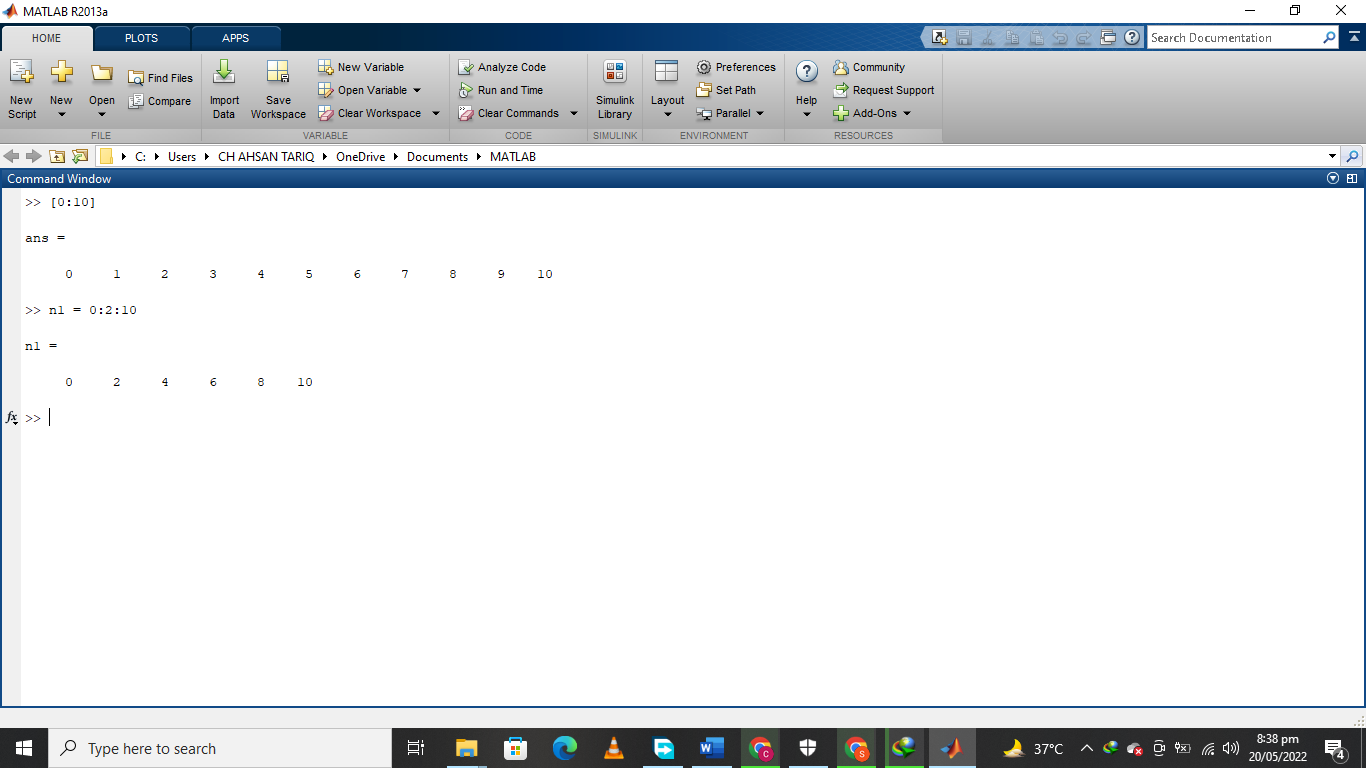
What will be the output of the above command?



If we wish the increment different from 1 (default value), then we indicate it as in the following:

>>n1 = 0:2:10

What will be the output after executing the above command?



An array has dimensions M × N, where M and N are positive integers. M is the number of rows and N is the number of columns.

If M= N = 1, the variable is a scalar.

If M = 1and N > 1, then the variable is a row vector.

If M > 1 and N = 1, then the variable is a column vector.

If both M and N are greater than one, then the variable is a matrix. If M = N, then the variable is a square matrix.

The coefficients of an array are real or complex numbers.

# Matrices:

## Defining Matrices

Matrices are represented in MATLAB by writing individual rows, separated by semicolons For example, typing

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

It will create a 3 x 3 matrix in MATLAB in which the first the workspace row has elements 1, 2, 3, the second row has elements 4, 5, 6, and so on (note the use of the brackets [ ]).

>>B = X(2,3)

>>C = [1:10 ; 11:20]

>>D = 1 : 2 : 20 start , step , stop

>>E = X(:13) ;

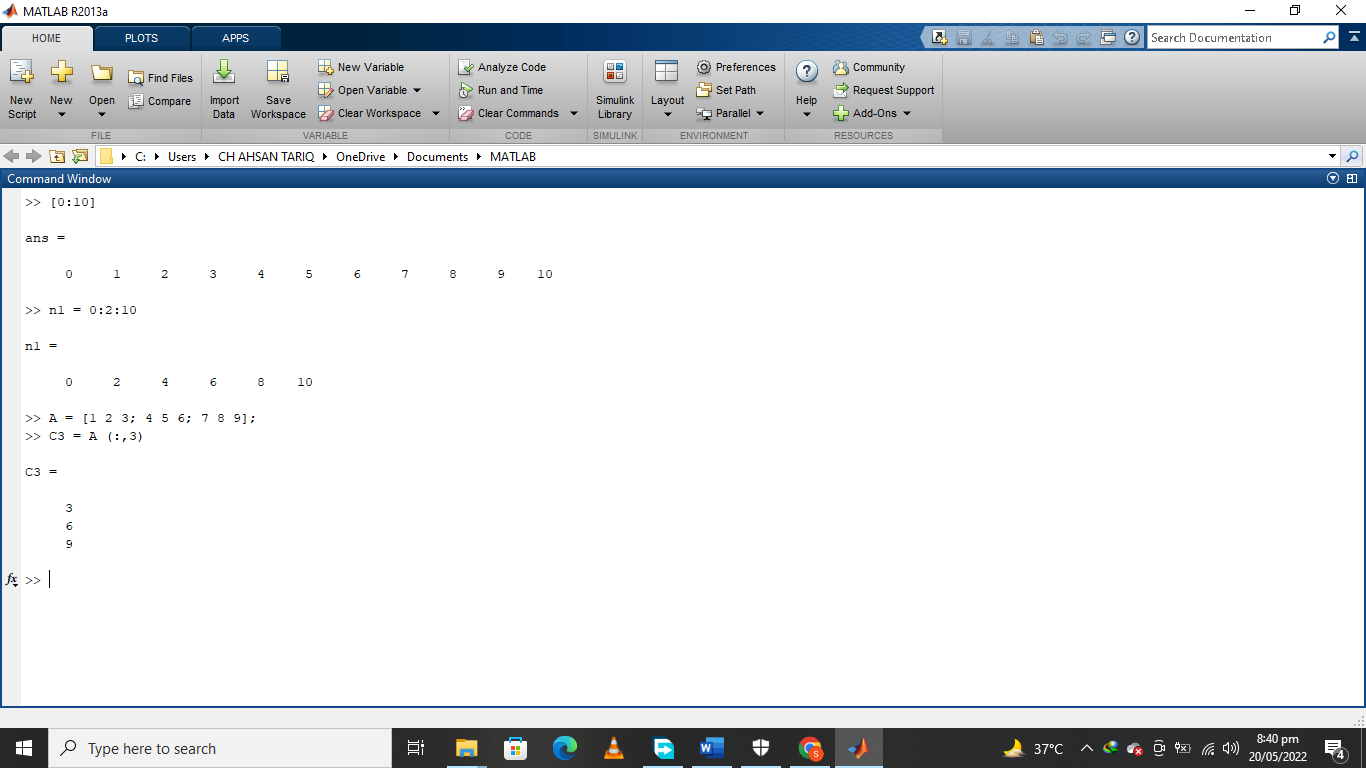
>>F = X(1:3 | 2 :4);

>>G= X(2, :) =[]

One of MATLAB most powerful index operators is the colon operator ”**:**”. For example, when used in matrix indexing, the colon operator selects parts of a matrix. For example, the following statement extracts the third column of A.

What will be the output?

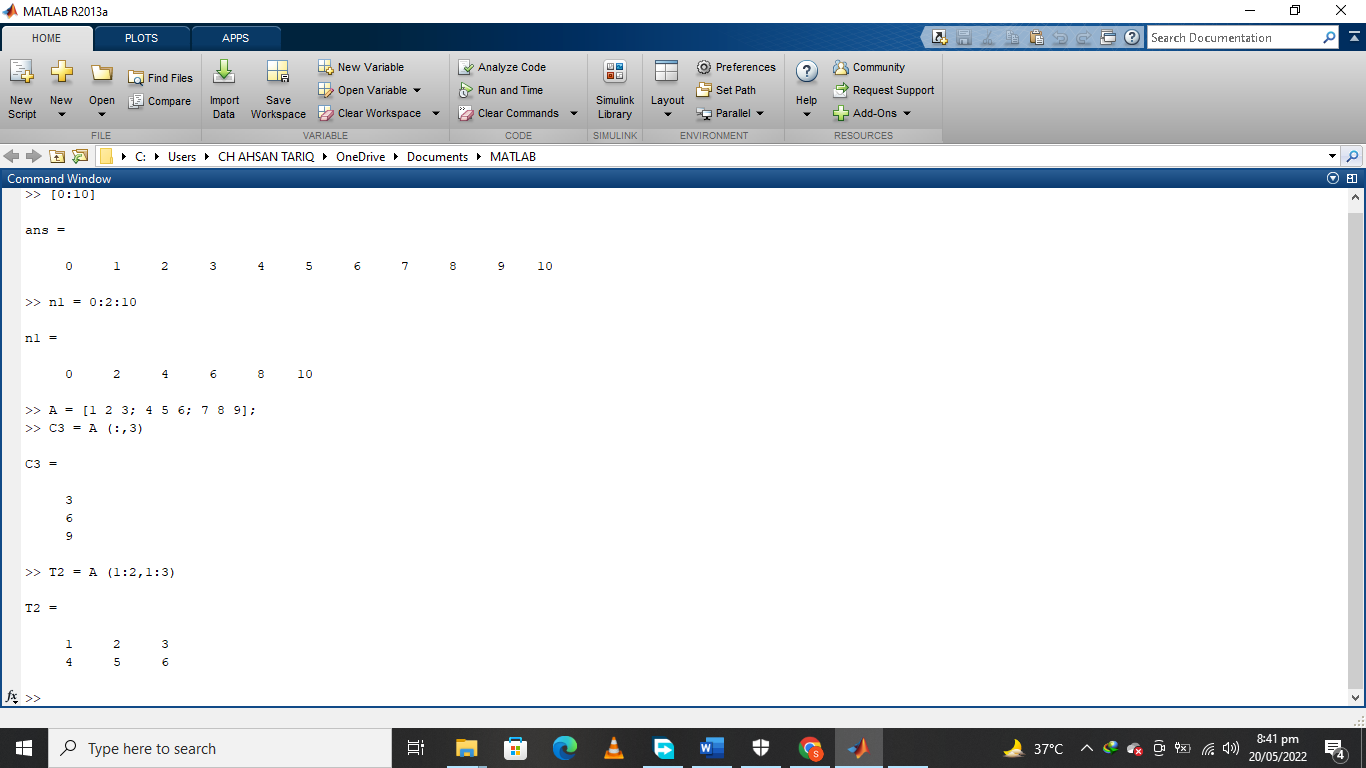
>> C3 = A (:,3)



Finally, the following statement extracts the top two rows:

>> T2 = A (1:2,1:3)

The output will be.



## Built-in Matrices Functions

MATLAB provides four functions that generate the following basic matrices:

>> E= *zeros(M,N)* generates an M x N matrix of zeros.

>>F= *ones(M,N)* generates a M x N matrix of ones.

>>G= *eye(M) generates a diagonal matrix*

>>H= *zeros* (M,N)+5 can generate M x N matrix with all elements are 5.

>> *I=ones* (M,N)\*5 can generate M x N matrix with all elements are 5.

>> *rand(M,N)* generates a M x N matrix whose entries are uniformly distributed random numbers in the interval [0 to 1].

>> *randn(M,N)* generates a M x N matrix whose numbers are normally distributed (i.e Gaussian) numbers with mean 0 and variance 1.

>>L = *diag(D) Can extract the diagonal of Matrix M*

>>M= *diag(D,2) Can extract the 2nd bi-diagonal of Matrix*

>>N= *diag(k) if k is vector then can generate matrix with elements of k as diagonal of new matrix*

>>o*= [A, B+30 ; C-20 , D\*2 ] can generate a very large matrix by using already defined matrices.*

>>P= repmat*(F,M,N) Can repeat already defined matrix by provided number of row and column repetition numbers M&N.*

>>Q= *sum(J) can sum the columns of any already defined matrix*

>>R= *transpose(Q)* or Q’ this command can generate transpose matrix

## Matrices Concatenation:

Matrix concatenation is the process of joining small matrices to create larger matrices. The concatenation operator is the pair of square brackets, [ ]. Earlier in this section, when we wrote

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

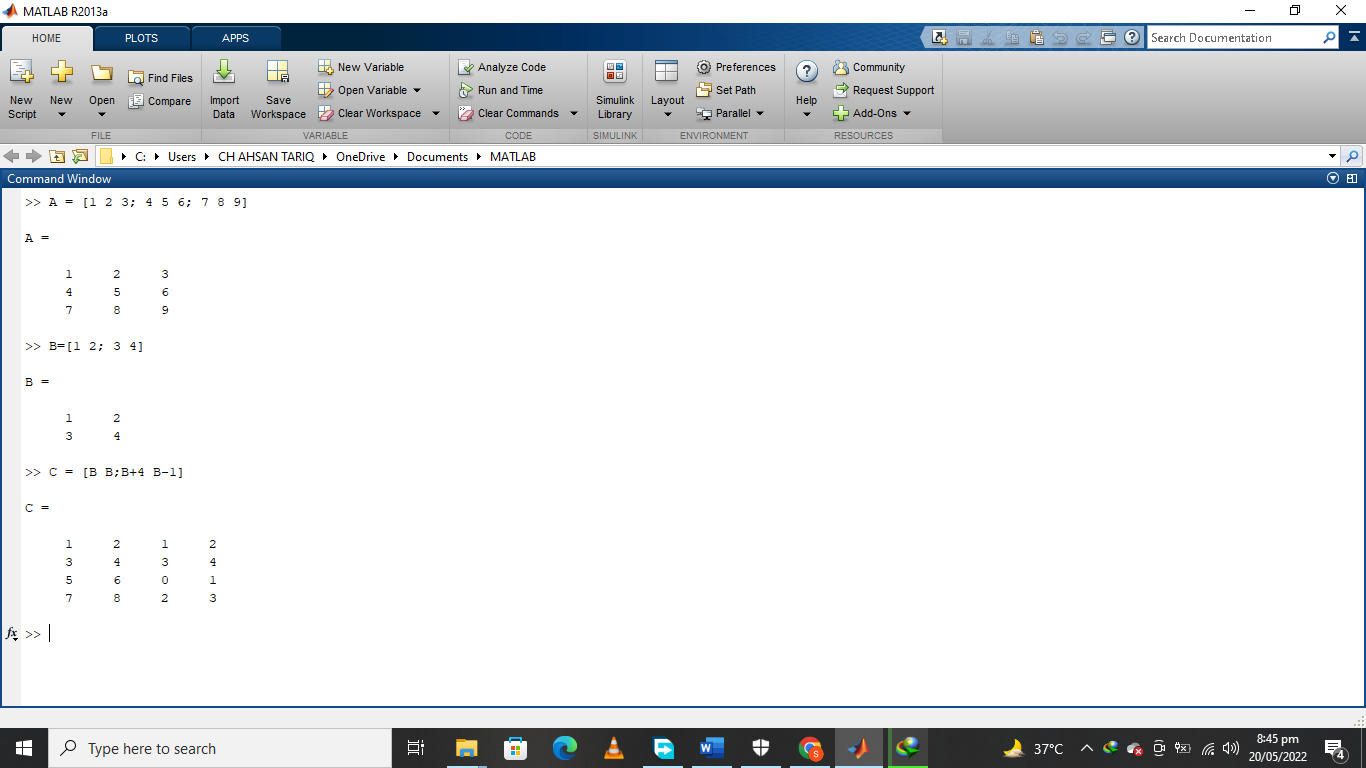
This command will create a matrix of size 3 x 3. Note that individual elements are separated by a space. Separating the elements by commas would yield the same result. Note also the order in which the elements were concatenated. The group of three elements before the first semicolon formed the first row, the next group formed the second row, and so on. Clearly, the number of elements between semicolons has to be equal. This concept is also applicable when the elements themselves are matrices. For example, consider the 2 x 2 matrix

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

The statement

>> C = [B B;B+4 B-1]

in which B + 4 and B - 1 indicate adding 4 and subtracting 1 from all elements of B, respectively, yields the following result.



# MATLAB Script files:

Using command prompt is recommended for temporary calculations. However, it is mostly desirable to save code so that we can run it again in future Matlab sessions. The save Matlab files are called m-files and must have an extension .m.

There are two types of m-files. Scripts and functions.

* + Scripts do not accept input arguments or return output arguments. They operate on data in the workspace.
  + Functions accept input arguments and return output arguments. They operate on data in the workspace.

# Logical Operator :

y=[1,3,5,7]

z=[3,2,5,9]

We can perform logical operations on variables, elements of arrays by logical operators

x>=y x<=y

x==y x-=y

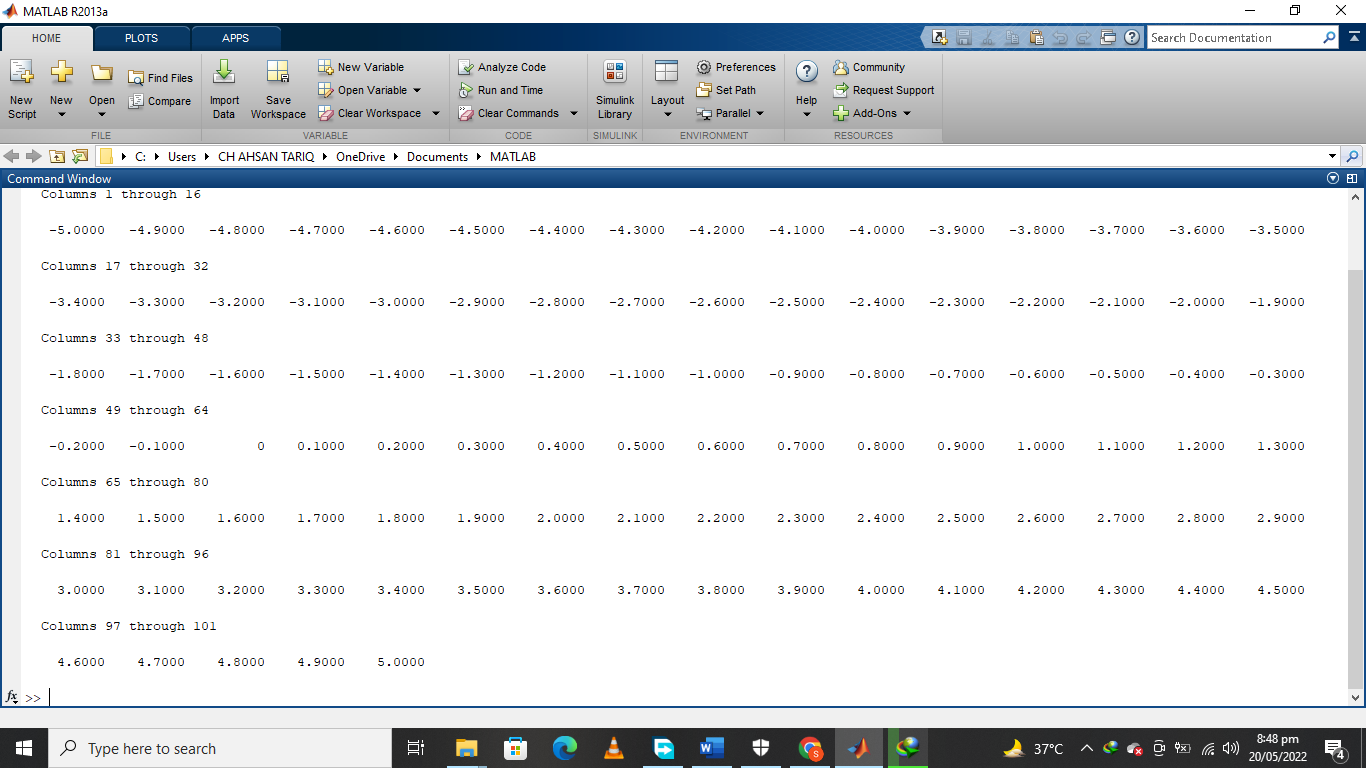
## Comments:

Starting a line with percent “%” sign tells MATLAB that line is a comment. If comments are placed in very beginning of m-file, then by typing help filename, MATLAB prints the comments that appear at the beginning of file. This will allow the user to expand MATLAB help facility for user-defined functions.

## Homework Problems:

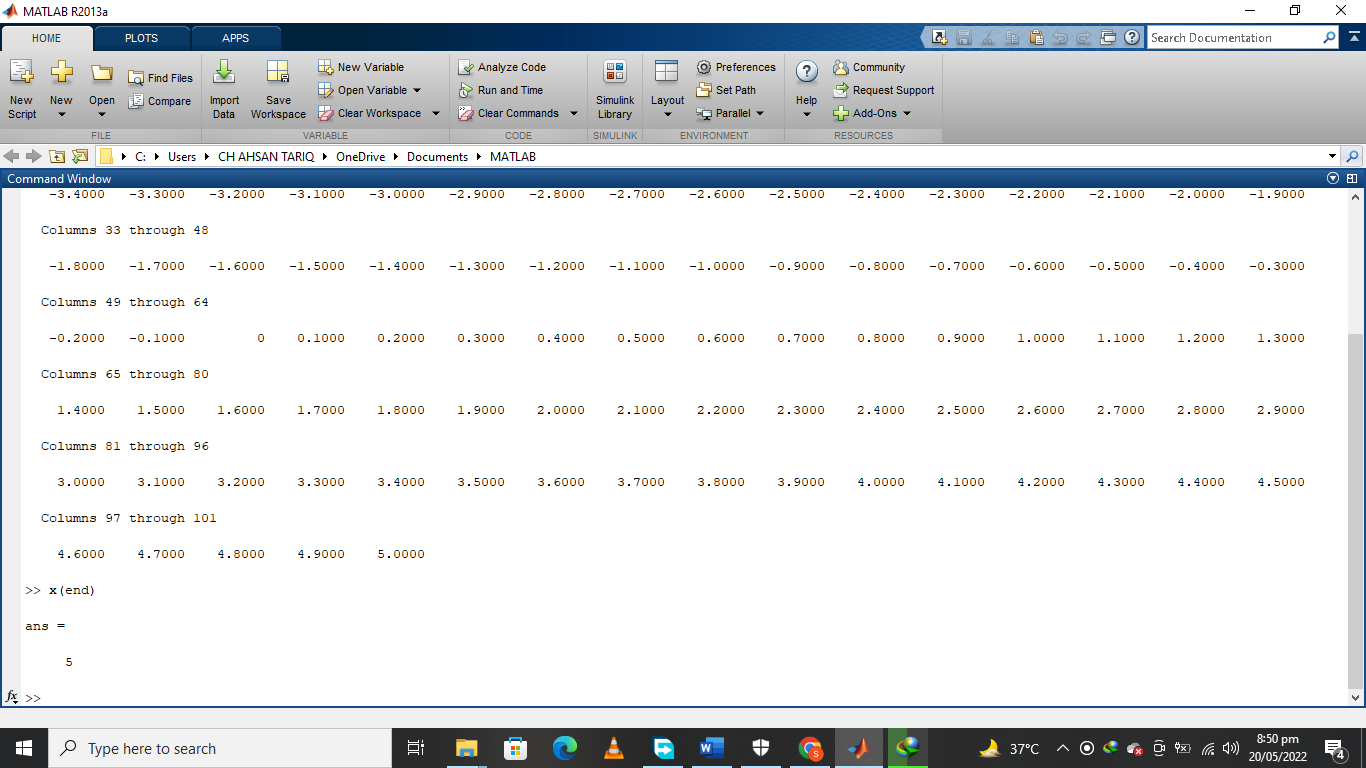
**Question 1:** Write MATLAB commands to initialize a column vector whose entries will start from 5 and end with +5 with an increment of 2.

**Ans:**



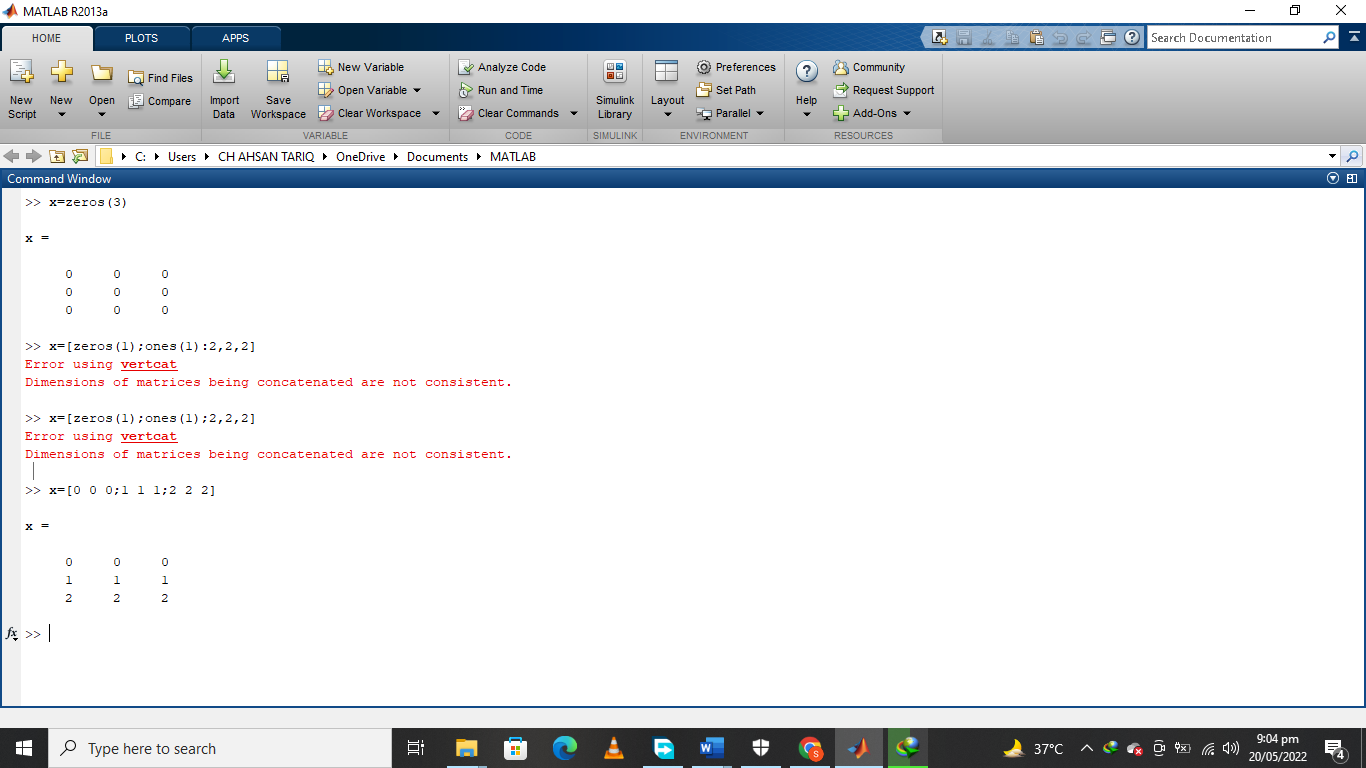
**Question 2:** Write MATLAB commands to access all entries except the last one, of a vector initialized in Question 1. Repeat the same problem to access only the last entry in the already stored vector.

**Ans:**



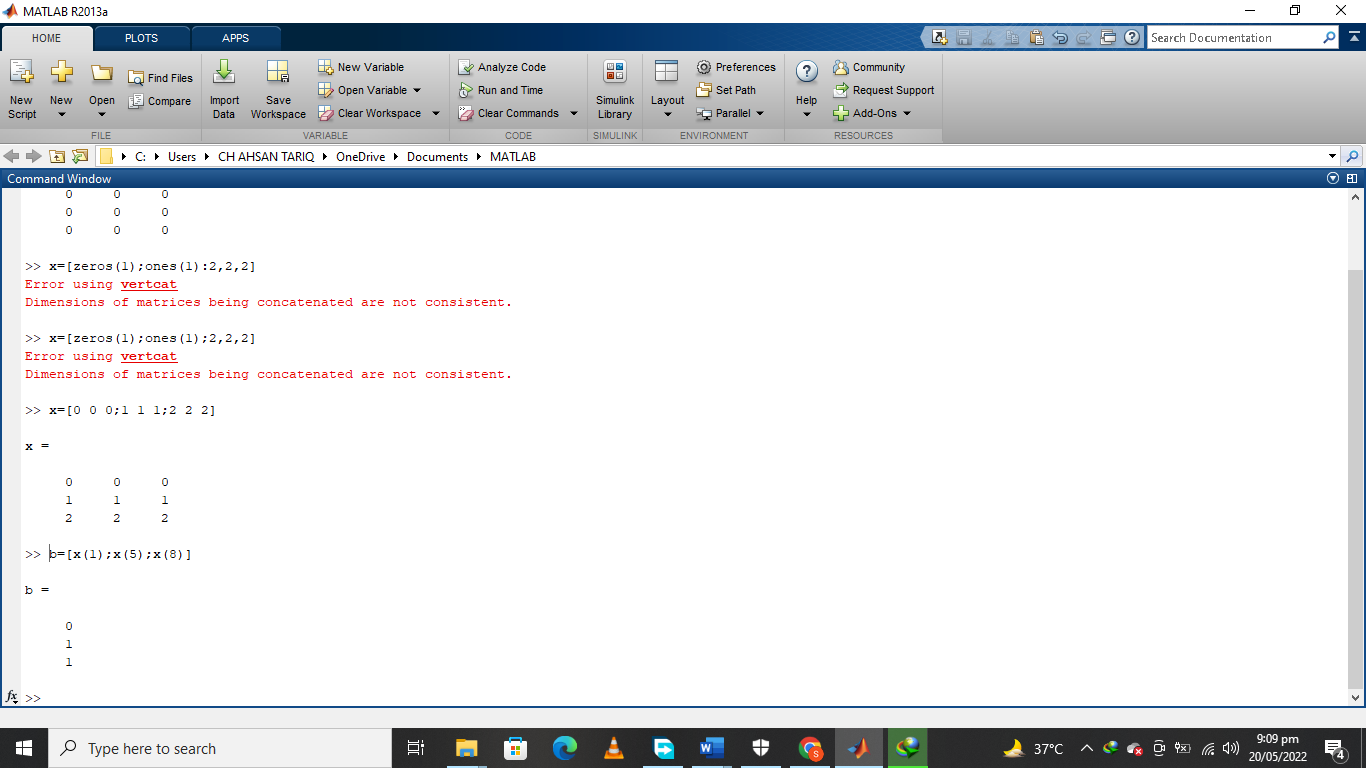
**Question 3:** Write down a single MATLAB command to initialize a 3 x 3 matrix, whose first rows contains zeros, second row contains ones and third row contains 2. The matrix should be:

**Ans:**



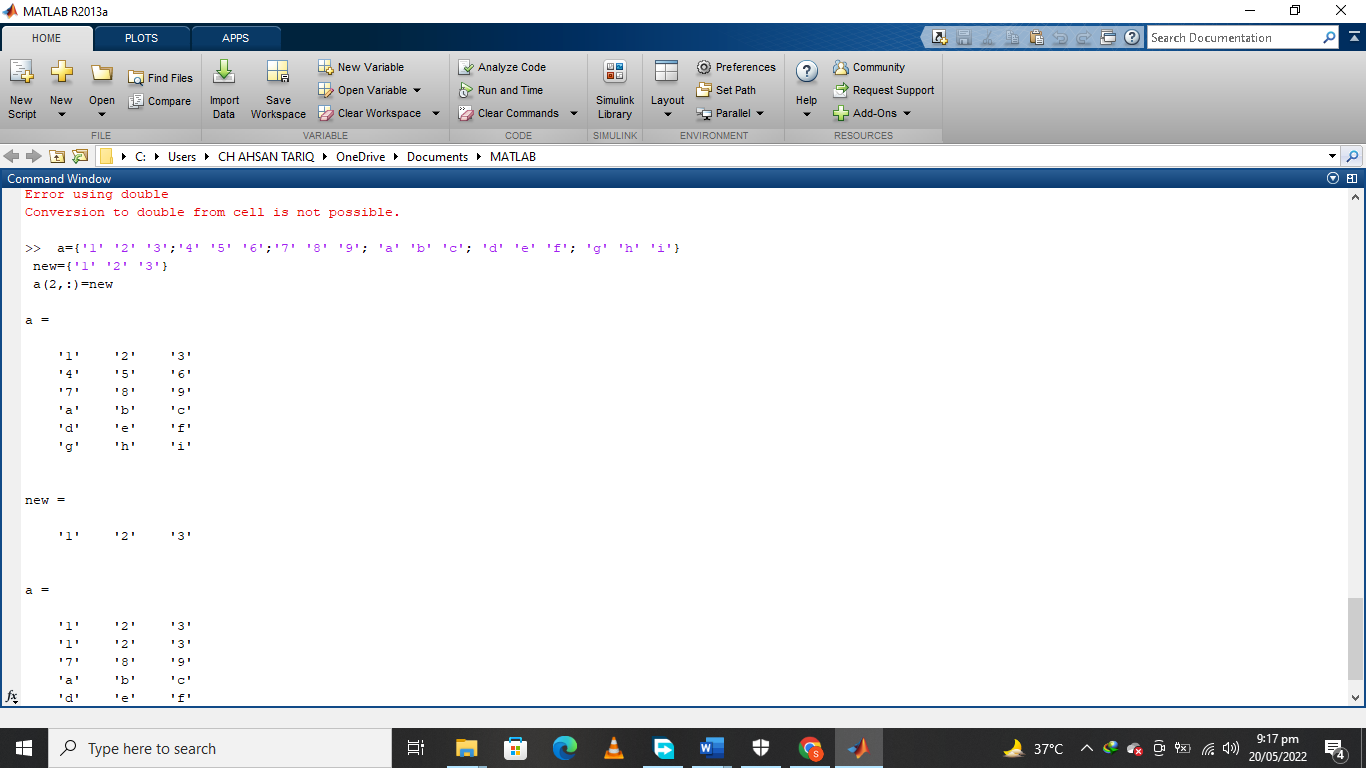
**Question 4:** Write MATLAB commands to initialize a matrix and access the elements from first and third rows and columns of B.

Ans:



**Question 5:** Write MATLAB command to replace the second row with elements 1, 2, and 3 for the matrix initialized in Question?

Ans:



The Islamia University of Bahawalpur

**U**niversity **C**ollege of **E**ngineering **&T**echnology **D**epartment of **C**omputer **S**ystem **E**ngineering

|  |  |  |
| --- | --- | --- |
| **LAB MANUAL** | **SIGNALS AND SYSTEMS EE-311** | **4thSemester** |

**LAB EXPERIMENT # 02**

**Generation of continuous and discrete time signals in MATLAB**

|  |  |
| --- | --- |
| **Student Name: Ahsan Tariq** | **Roll No: 20-CSE-26** |
| **Lab Instructor Signatures:** | **Date:** |

## OBJECTIVE:

To generate and plot the sinusoidal, exponential, unit-step function, impulse, and ramp type continuous-time signals and discrete-time sequences in MATLAB.

# Continuous-time Signals:

## SINUSOIDAL SIGNALS:

The first basic class of continuous-time signals presented is that of sinusoidal signals. This type of the signal is the mathematical function that describes the smooth repetitive oscillations. Its most basic form as function of time (t) is:

𝑦(𝑡) = 𝐴𝑠i𝑛(𝜔𝑡 + 𝜃) (1)

where𝐴 is the amplitude of the sinusoidal signal, 𝜔 is the angular frequency in rad/sec (𝜔=2f), and θ is the phase shift (in radians). The sinusoidal signals are periodic signals with fundamental period T given by T = 2/𝜔. Finally, a useful quantity is the frequency *f* given in Hertz. Frequency f is defined by f = 1/T or f =𝜔/2.

**Example 1:** Plot the signal 𝑥(𝑡) = 3cos(3𝜋𝑡 + 𝜋) in four periods.

3

First, the period T is calculated as T = 2/𝜔= 2=3= 2/3. Hence, the MATLAB implementation is as follows.

## MATLAB IMPLEMENTATION:

% STUDENT NAME:

% Roll #:

% Generation and plotting of continuous-time sin wave.

A = 3;

Omega = 3\*pi; theta=/3;

T = 2\*pi/omega;

t = 0:0.01:4\*T;

% The amplitude of thesignal is 3.

% The angular frequency is 3.

% The phase is /3.

% The period is 2/3.

% The time is defined from 0 to 4T.

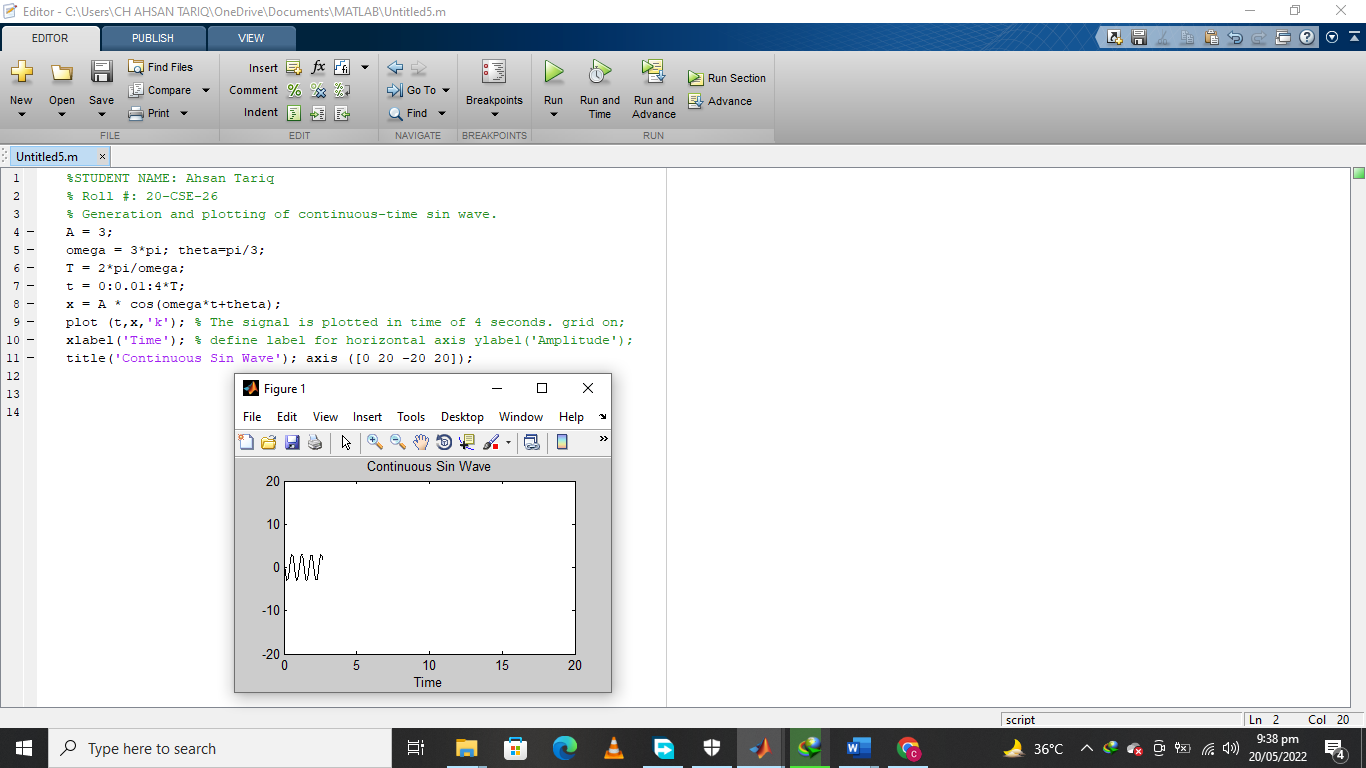
x = A \* cos(omega\*t+theta);% Signal definition

plot (t,x,'k'); % The signal is plotted in time of 4 seconds. grid on;

xlabel('Time'); % define label for horizontal axis ylabel('Amplitude');

title('Continuous Sin Wave'); axis ([0 20 -20 20]);

**OUTPUT**:



The continuous-time signal isdrawn by using the “*plot*” command. Whereas, “*stem*” command is more appropriatewhen dealing withdiscrete-time signals.

**Task 1:** Modify the above code to create a sin wave of 5-second period and produces 3 cycles of the waveform.

## MATLAB CODE:

|  |
| --- |
| %STUDENT NAME: Ahsan Tariq  % Roll #: 20-CSE-26  % Generation and plotting of continuous-time sin wave.  A = 3;  omega = 3\*pi; theta=pi/3;  T = 5\*pi/omega;  t = 0:0.01:3\*T;  x = A \* sin(omega\*t+theta);  plot (t,x,'k'); % The signal is plotted in time of 4 seconds. grid on;  xlabel('Time'); % define label for horizontal axis ylabel('Amplitude');  title('Continuous Sin Wave'); axis ([0 20 -20 20]); |
|  |

**OUTPUT:**

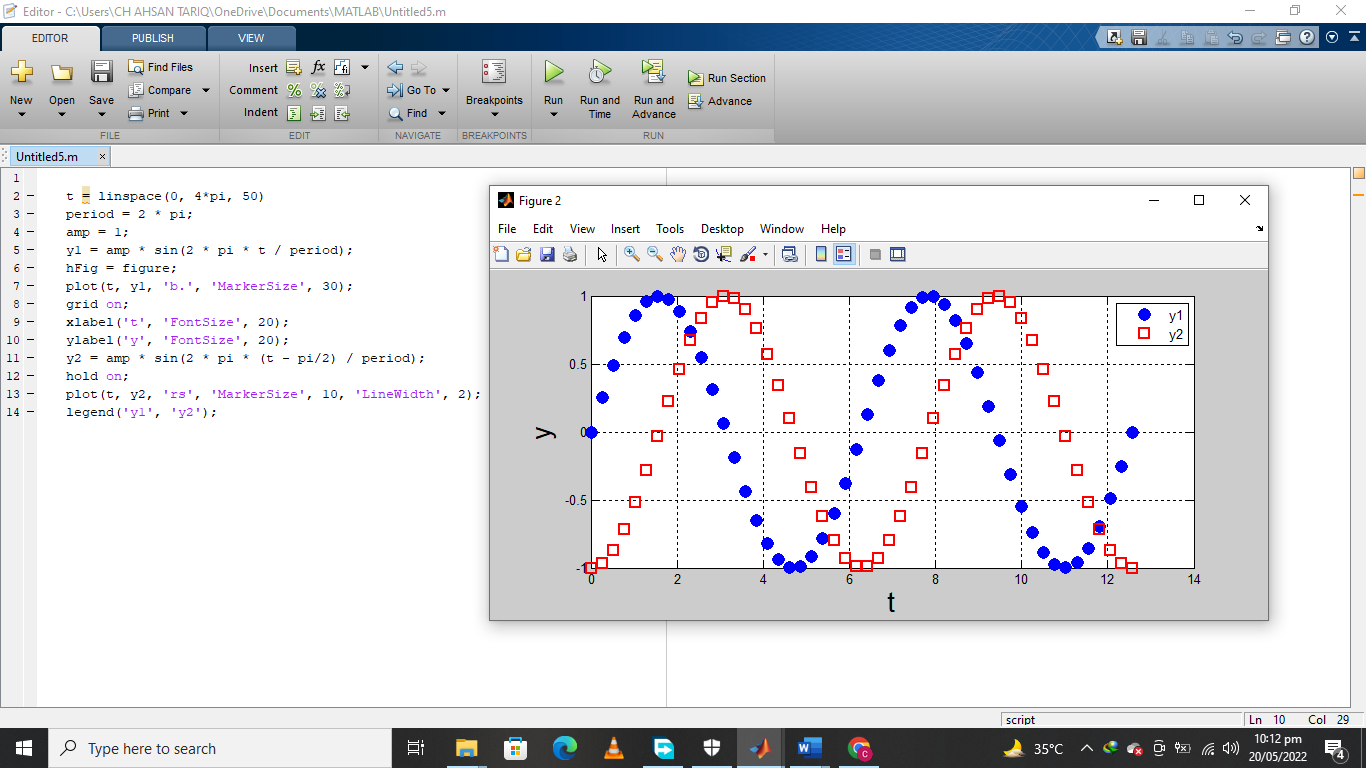
|  |
| --- |
|  |

When referring to sinusoidal signals, we refer both to sines and cosines signals, as a cosine and sine are in fact the same signal except with a phase difference of /2, i.e., each point on the cosine wave occurs exactly 1/4 cycles earlier than the corresponding point on the sine wave.

**Task 2:** Write MATLAB code to plot in the same figure both cos(𝑡) and sin(𝑡) for time of one period. Plot the sine signal with solid line, while the cosine signal with the dotted line. Also try to use different colors for the two curves.

## MATLAB CODE:

**OUTPUT:**



## EXPONENTIAL SIGNALS:

Exponential signals are signals of the form𝑥(𝑡) = 𝐴𝑒(𝑎𝑡), where *A* is the amplitude at *t*=0, and *a* is the rate of decay or increase. If *a*> 0, 𝑥(𝑡)is an increasing functionwhile if *a*< 0, 𝑥(𝑡)is a decreasing function. At *t*= 0, the signal takes the value 𝑥(0) = 𝐴.

**Example 2:**Plot the signals𝑥(𝑡) = 3𝑒0.4𝑡and 𝑦(𝑡) = 2𝑒−0.9𝑡in time interval−2 ≤ 𝑡 ≤ 5.

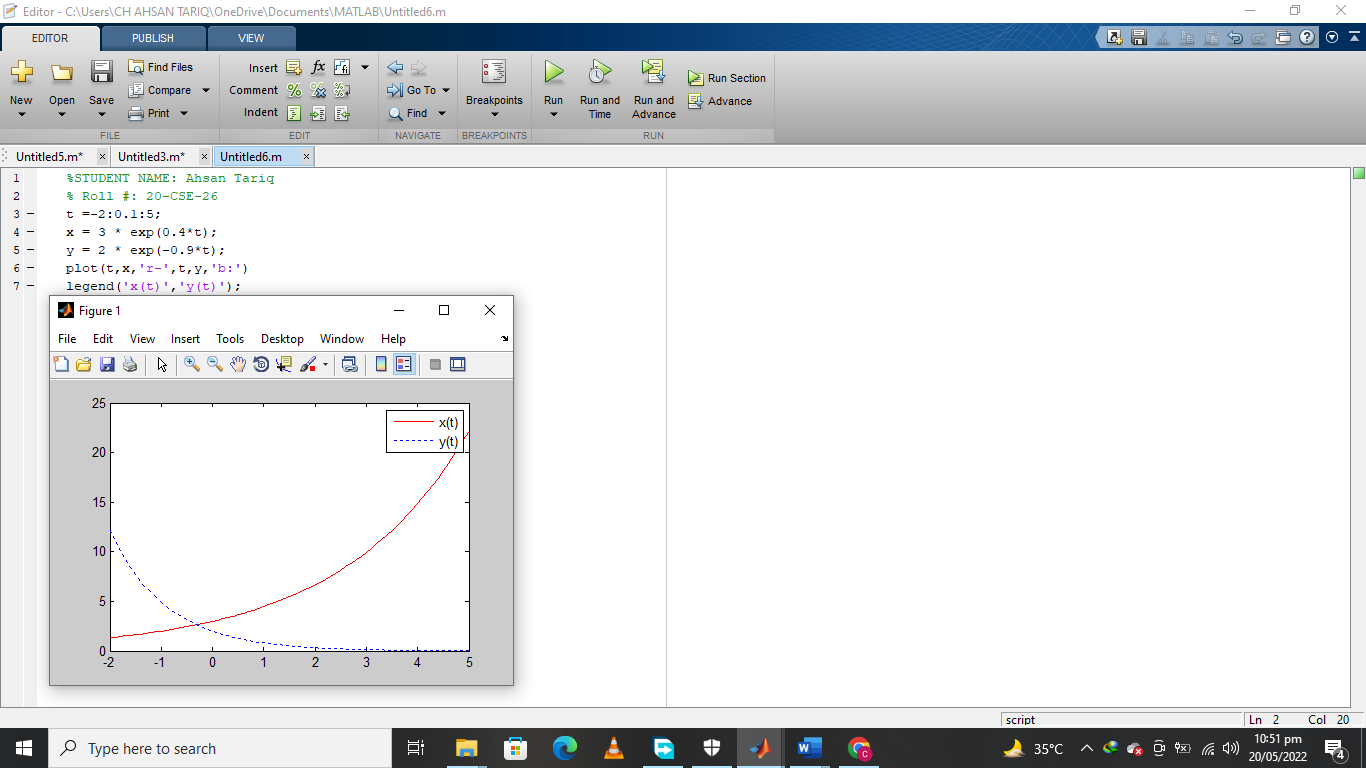
t =-2:0.1:5;% The time is defined from -2 to 5.

x = 3 \* exp(0.4\*t); % Define first exponential signal. y = 2 \* exp(-0.9\*t); % Define second exponential signal.

% Plot the two signals on the same figure, the x(t) in solid line and red color, signal y(t) in dashed line in blue color. plot(t,x,’r-‘,t,y,’b:’

legend(‘x(t)’,’y(t)’);

**OUTPUT:**



## Unit StepFunction:

Another basic signal is the unit step function u(t). The function is used in the mathematics of control theory and signal processing to represent a signal that switches on at a specified time and stays switched on indefinitely. It is also used in structural mechanics together with the Dirac delta function to describe different types of structural loads.

The unit step function can be defined as:

𝑢(𝑡) = {1 𝑡 ≥ 0}

0 𝑡 < 0

The MATLAB command that generates the unit step function is the command

*heaviside(t).*According to MATLAB programmers, unit step function is given by:

𝑢(𝑡) = {1 𝑡 > 0}

0 𝑡 < 0

i.e., it is not defined at t = 0. There are different ways to present and plot the unit step function in MATLAB.

**Method 1:** Using the *heaviside* command

t =-5:0.1:10;

u = heaviside(t) plot(t,u) ylim([0.3 1.3])

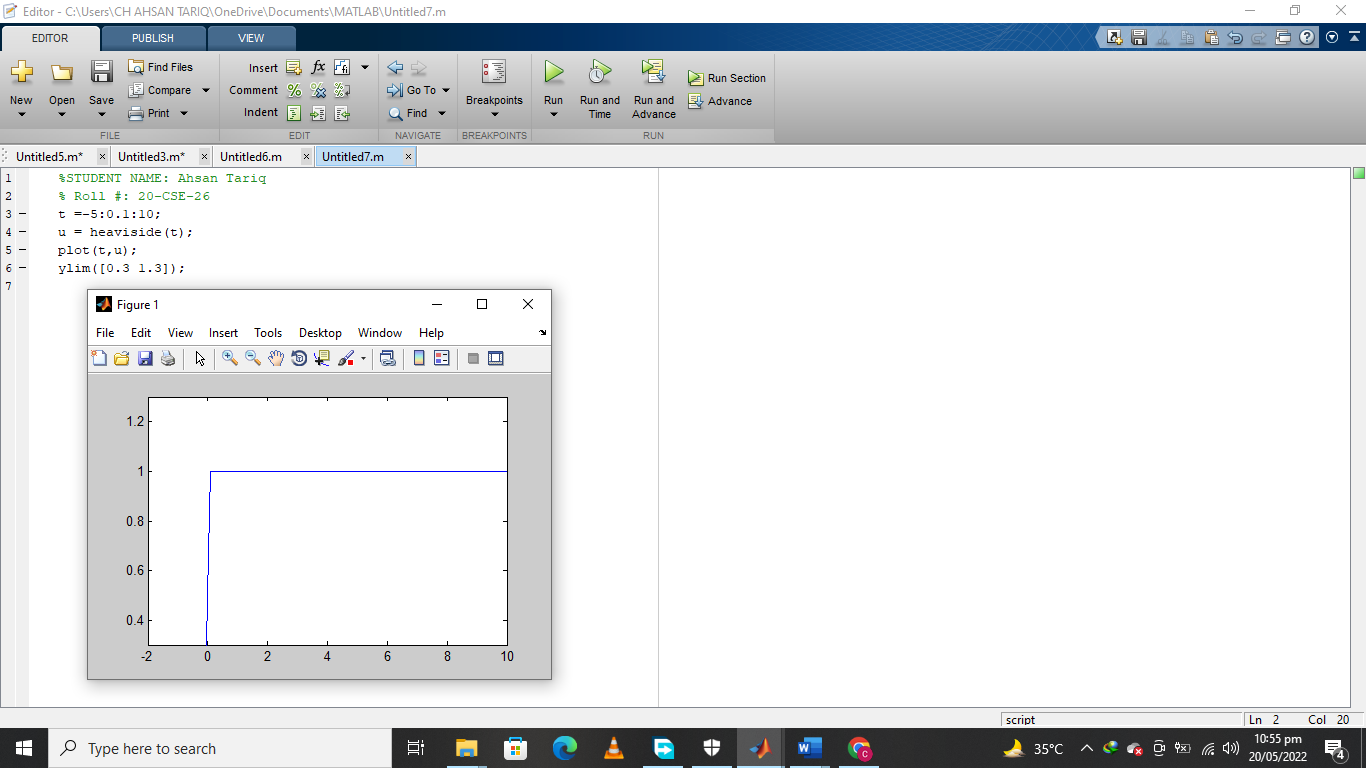
% Definition of the timeinterval−5 ≤ 𝑡 ≤ 10.

% Definition of u(t).

% Graph of u(t).

%Change of axis forbetter appearance ofthe graph.

## OUTPUT:



**Method 2: Implementation with specific number of zeros and ones**

t=-5:0.1:5;

u=[zeros(1,50) ones(1,101)];

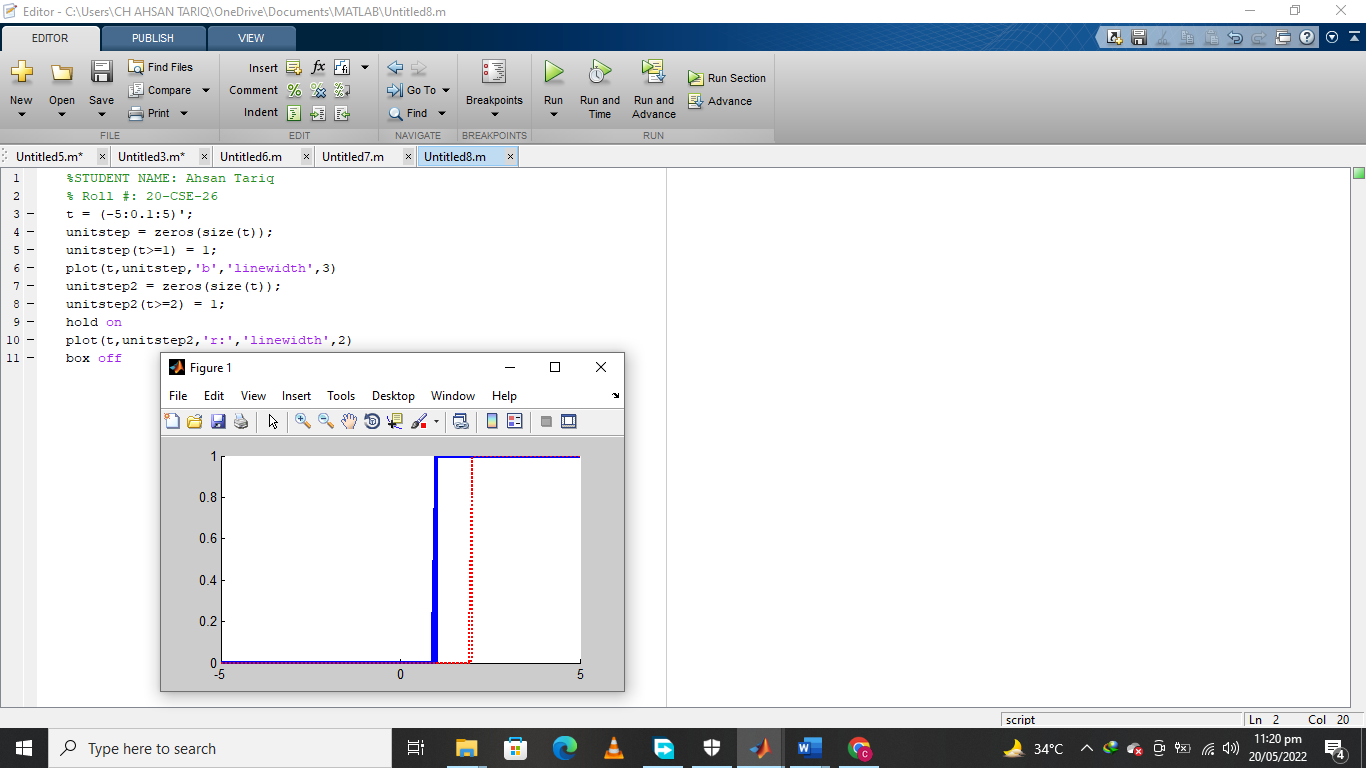
plot(t,u,'k');

xlabel('Time'); ylabel('Amplitude'); ylim([-0.3 1.3]);

title('Unit step function');

In the above method, the time interval *t* contains two different values. The values of first 50

elements of *t* are zero, while the next 101 elements including zero have unity values. Finally, two subparts of *t* are concatenated.



The general form of the unit step function is:

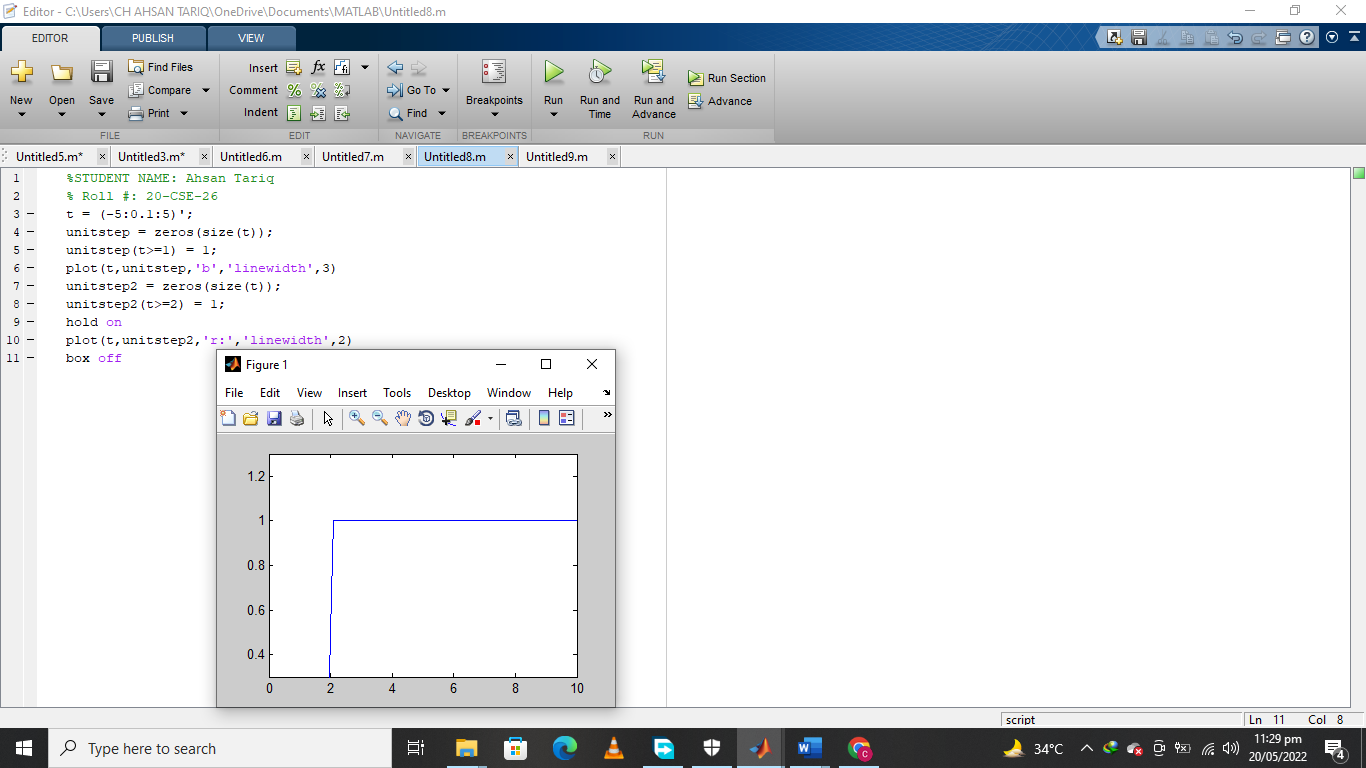
( ) 1 𝑡 − 𝑡0 ≥ 0 → 𝑡 ≥ 𝑡0

𝑢 𝑡 − 𝑡0 = { }

0 𝑡 − 𝑡0 < 0 → 𝑡 < 𝑡0

Suppose that we want to define and plot the unit step function for t0 = 2, i.e., we want to define and plot the function u(t - 2).

## MATLAB CODE:



t = -5:0.1:10;%Definition of the timeinterval for t = –5 to 10.

u = heaviside(t-2) % Definition of u(t - 2). plot(t,u)

ylim([0.3 1.3])

Noticethat function 𝑢(𝑡 − 2)becomes 1 from thetime instance t = 2 andafterward. Thus,𝑢(𝑡 −

2)is a shifted by2 units to the rightversion of u(t).

## UNIT IMPULSE SIGNALS:

The **Dirac delta function**, or ***δ* function**, is (informally) a generalized function on the real number line that is zero everywhere except at zero, with an integral of one over the entire real line. The delta function is sometimes thought of as an infinitely high, infinitely thin spike at the origin, with total area one under the spike, and physically represents an idealized point

mass or point charge. In the context of signal processing it is often referred to as the **unit impulse**. Its most basic discrete form as function of time is

**MATLAB CODE FOR IMPLEMENTATION:**

# Continuous-time unit impulse signal

Amplitude

|  |  |  |  |
| --- | --- | --- | --- |
| **MATLAB CODE** | **OUTPUT** | | |
| t = -20:20; | 3  2.5  2  1.5  1  0.5  0  -20 | Continuous Unit Impulse |  |
| u =[zeros(1,20) 1 zeros(1,20)]; |  |  |
| plot(t,u,'k'); |  |  |
| grid on; |  |  |
| xlabel('Time'); |  |  |
| ylabel('Amplitude'); |  |  |
| axis([-20 20 0 3]); |  |  |
| title('Continuous-time unit |  |  |
| impulse'); |  |  |
|  | -10 0 10 | 20 |
|  | Time |  |

1. **Discrete-time unit impulse sequence**

Discrete Unit Impulse

3

2.5

2

1.5

1

0.5

0

-20

-10

0

Time

10

20

n = -20:20;

u =[zeros(1,20) 1 zeros(1,20)];

stem(n,u,'k'); grid on; xlabel('Time');

ylabel('Amplitude'); axis([-20 20 0 3]);

title('Discrete-time unit impulse sequence');

**OUTPUT**

**MATLAB CODE**

Amplitude

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## UnitRampFunction:

The unit-ramp function r(t) is defined in terms of the unit step function u(t) as:

𝑟(𝑡) = 𝑡. 𝑢(𝑡) = {𝑡 𝑡 ≥ 0}

0 𝑡 < 0

Therefore, inorderto define the ramp function, first we have to construct unit stepfunction.

## MATLAB CODE:

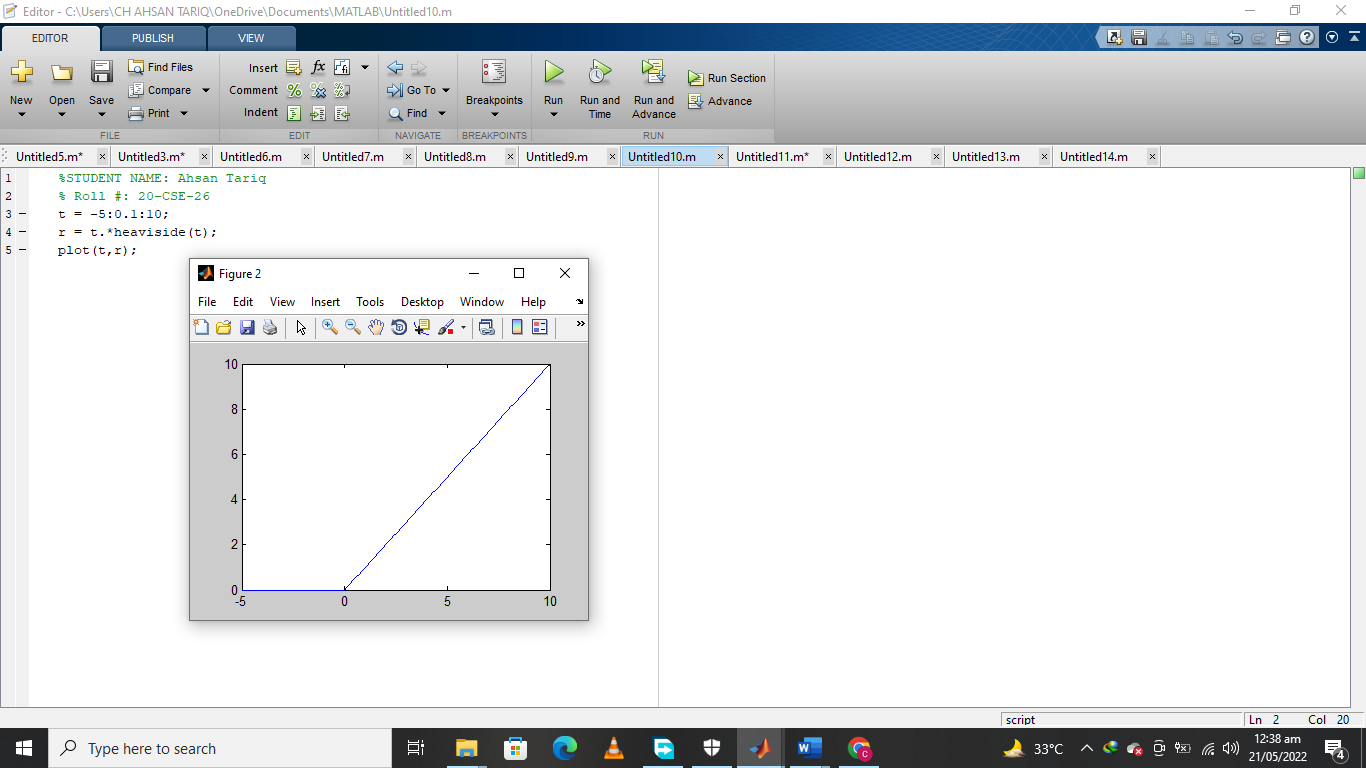
t = -5:0.1:10;

% Definition of the timeinterval [-5 to 10].

r = t.\*heaviside(t); %Definition of the ramp function r(t)=tu(t).

plot(t,r) % Graph of r(t).

**OUTPUT:**

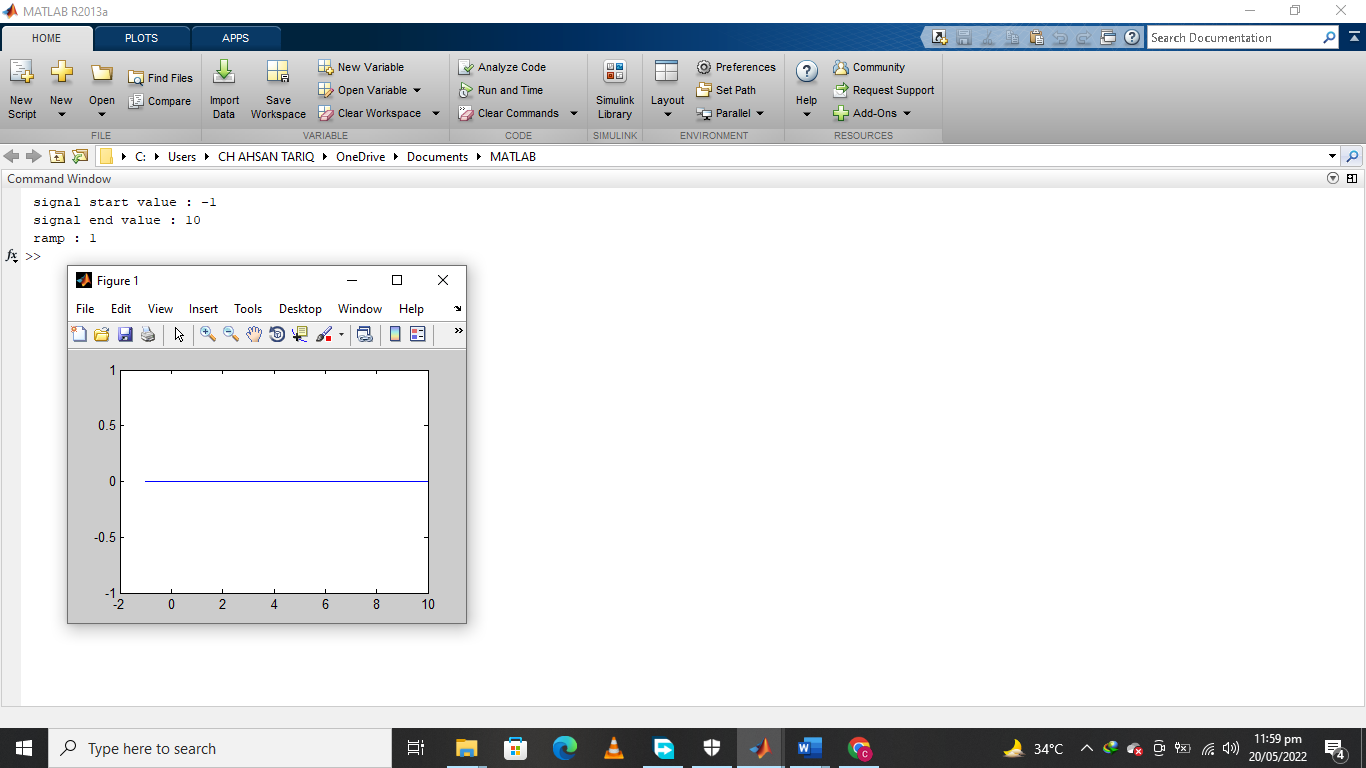


**TASK 3:** Define and plot the ramp function 𝑟(𝑡 − 1).

## MATLAB CODE:

|  |
| --- |
| %STUDENT NAME: Ahsan Tariq  % Roll #: 20-CSE-26  signal\_start = input(' signal start value : ');  signal\_end = input(' signal end value : ');  ramp\_value = input(' ramp : ');  a = [ signal\_start:signal\_end];  b =mod(a,ramp\_value);  plot(a,b) |

**OUTPUT:**

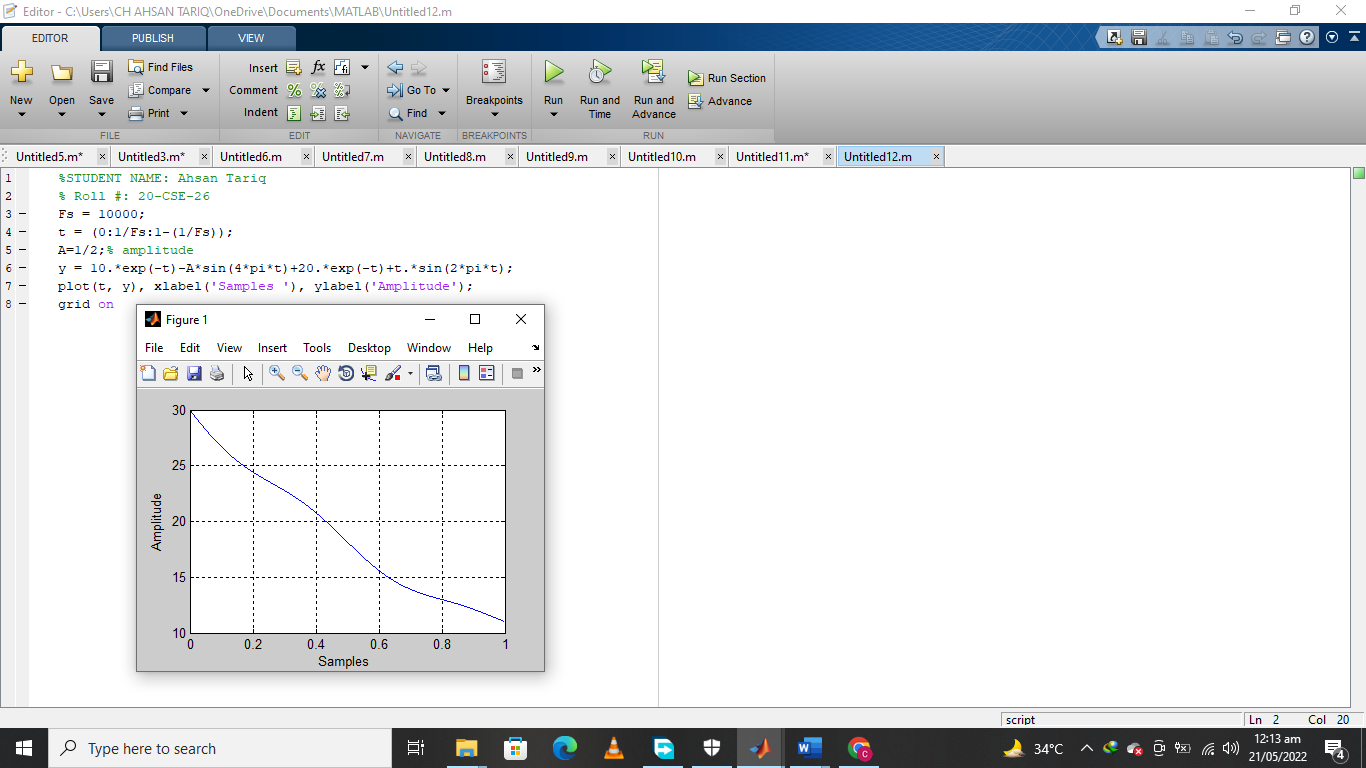


## HOMEWORK PROBLEMS:

**Question 1:** Generate and plot the following signal in MATLAB

*x*(*t*) = 10*e*−0.5*t*sin(4π*t*) + 20*e*−*t*sin(2π*t*)

Ans:

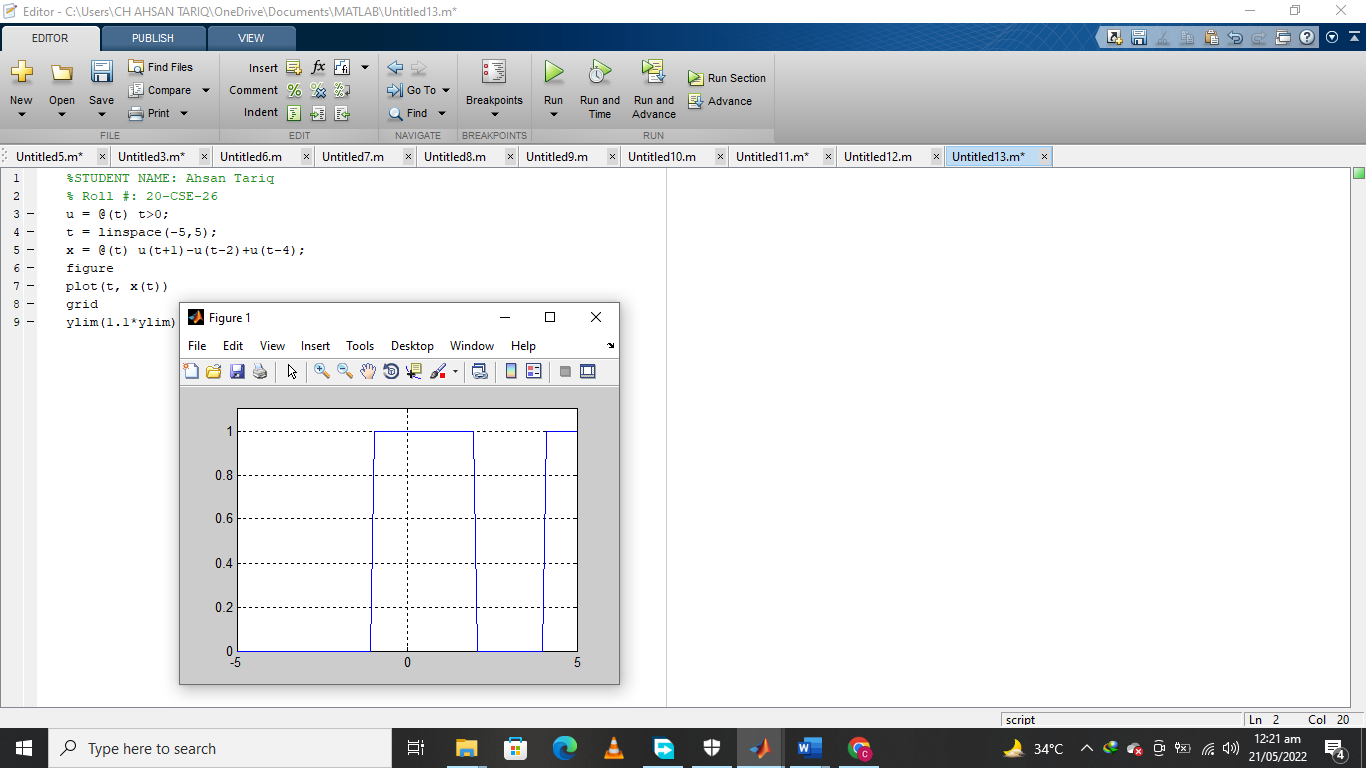


**Note:**You will use **.\*** instead of \* because we want the components of these vectors to be multiplied by their corresponding components of the other vectors. Thus, element-by-element operations require a dot before the operator.

**Question 2:**Plot the continuous-time signal 𝑥(𝑡) = 𝑢(𝑡 + 1) − 𝑢(𝑡 − 2) + 𝑢(𝑡 − 4)

1. Without using the command*heaviside.*
2. Using the command *heaviside*.

Ans:



**Question 3**: Plot the signal 𝑥(𝑡) = 𝑡𝑠i𝑛(2𝜋𝑡)(𝑢(𝑡) − 𝑢(𝑡 − 3))?

**Ans:**

