# Department of Computing

# MATH 333: Numerical Analysis

# Class: BSCS-9ABC

# Lab 1: Introduction to MATLAB

# Date: February 2nd, 2022

# Lab Engineer: Syed Muhammad Ali Musa

# Lab 1: Introduction to MATLAB

**Introduction**

MATLAB, which stands for MATrix LABoratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry.

**Objectives**

The purpose of this lab is to get familiar with Matlab.

**Tools/Software Requirement**

Matlab R2016a

**Description**

MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say

5, is a 1-by-5 matrix. The purpose of this tutorial is to familiarize the beginner to MATLAB, by introducing the basic features and commands of the program.

In order to get the most out this tutorial you are strongly encouraged to try all the commands introduced in each section and work on all the recommended exercises

Let us start with something simple, like defining a row vector with components the numbers 1, 2, 3, 4, 5 and assigning it a variable name, say x.

» x = [1 2 3 4 5]

x = 1 2 3 4 5

Note that we used the equal sign for assigning the variable name x to the vector, brackets to enclose its entries and spaces to separate them.

We could have used commas ( , ) instead of spaces to separate the entries, or even a combination of the two. The use of either spaces or commas is essential

**Column Vector**

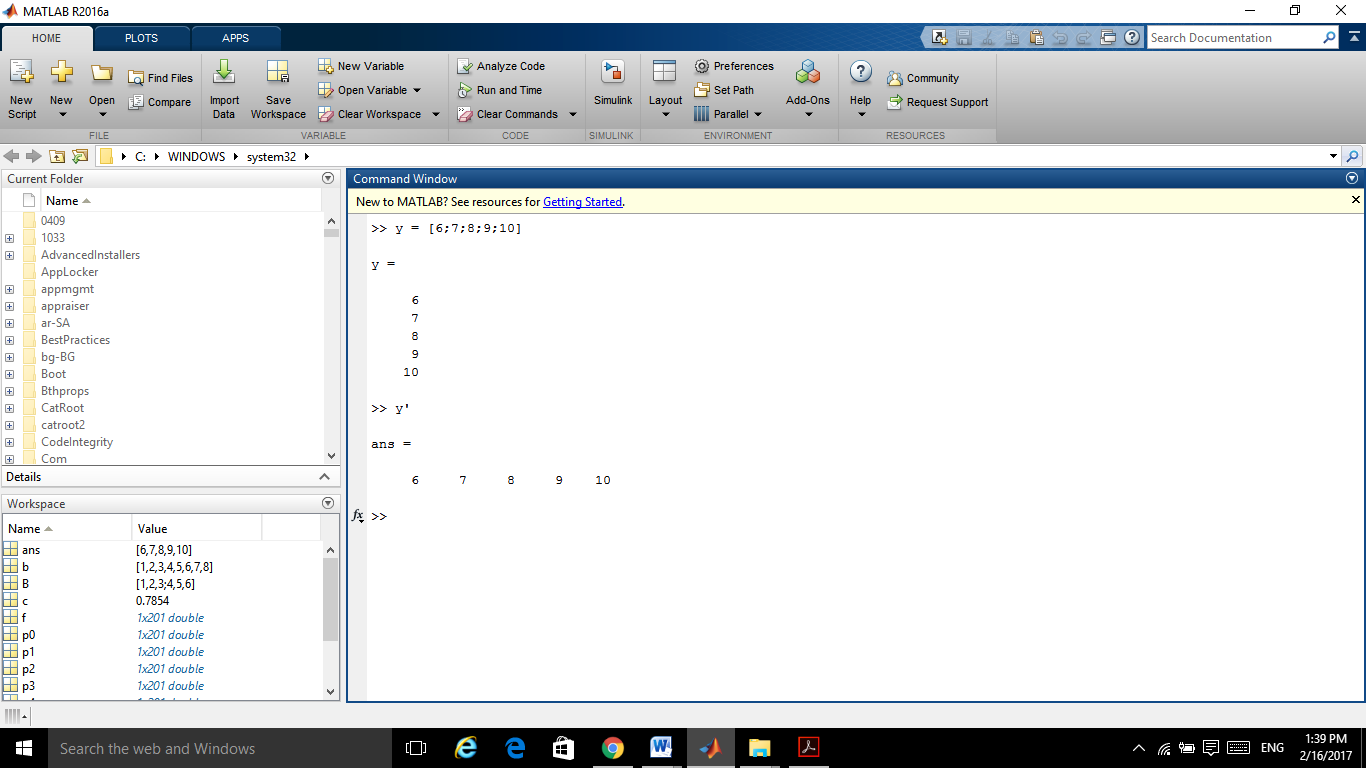
To create a column vector (MATLAB distinguishes between row and column vectors, as it should) we can either use semicolons ( ; ) to separate the entries, or first define a row vector and take its transpose to obtain a column vector.

Run following commands and see its output

» y = [6;7;8;9;10]

» y = [6,7,8,9,10]

y'



First, note that to take the transpose of a vector (or a matrix for that matter) we use the single quote ( ' ).

Also note that MATLAB repeats (after it processes) what we typed in. Sometimes, however, we might not wish to “see” the output of a specific command. We can suppress the output by using a semicolon ( ; ) at the end of the command line.

**Variable : ans**

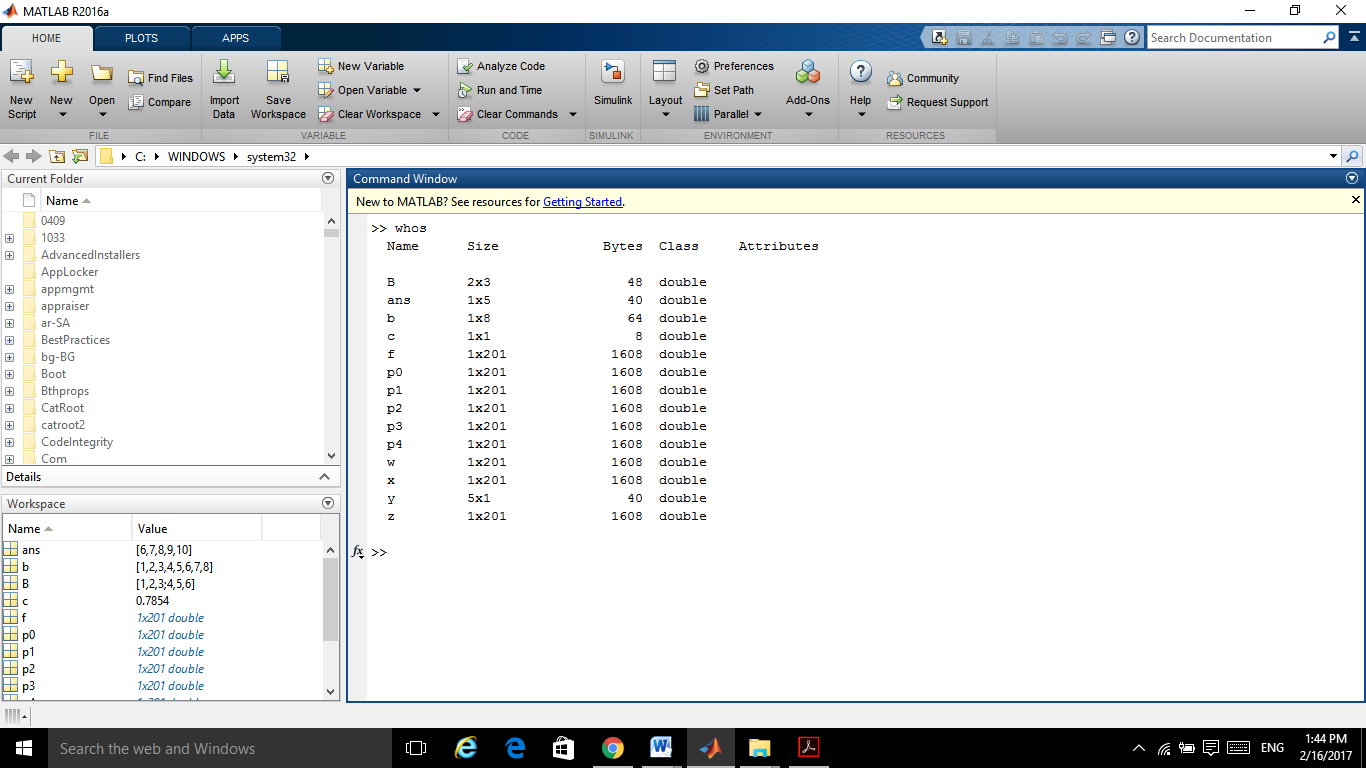
keep in mind that MATLAB automatically assigns the variable name “ans” to anything that has not been assigned a name. In the example above, this means that a new variable has been created with the column vector entries awhoss its value.

The variable ans, however, gets recycled and every time we type in a command without assigning a variable, ans gets that value

**Command: whos**

It is good practice to keep track of what variables are defined and occupy our workspace. Due to the fact that this can be cumbersome, MATLAB can do it for us. The command whos gives all sorts of information on what variables are active.

Run whos command and see its output.



**Command: who**

Run “who” command and see its behavior. How is it different from “whos” command.

» who

**Variable: clear**

If we no longer need a particular variable we can “erase” it from memory using the command clear variable\_name. Let us clear the variable ans and check that we indeed did so.

» clear ans

» who

The command clear used by itself, “erases” all the variables from the memory. Be careful, as

this is not reversible and you do not have a second chance to change your mind.

**Save and load variables**

You may exit the program using the quit command. When doing so, all variables are

lost. However, invoking the command save filename before exiting, causes all variables to be written to a binary file called filename.mat. When we start MATLAB again, we may retrieve the

information in this file with the command load filename.

**Command: help**

Help command provides help for any existing MATLAB command. Let us try this command on the command who.

» help who

WHO List current variables.

**Defining matrix**

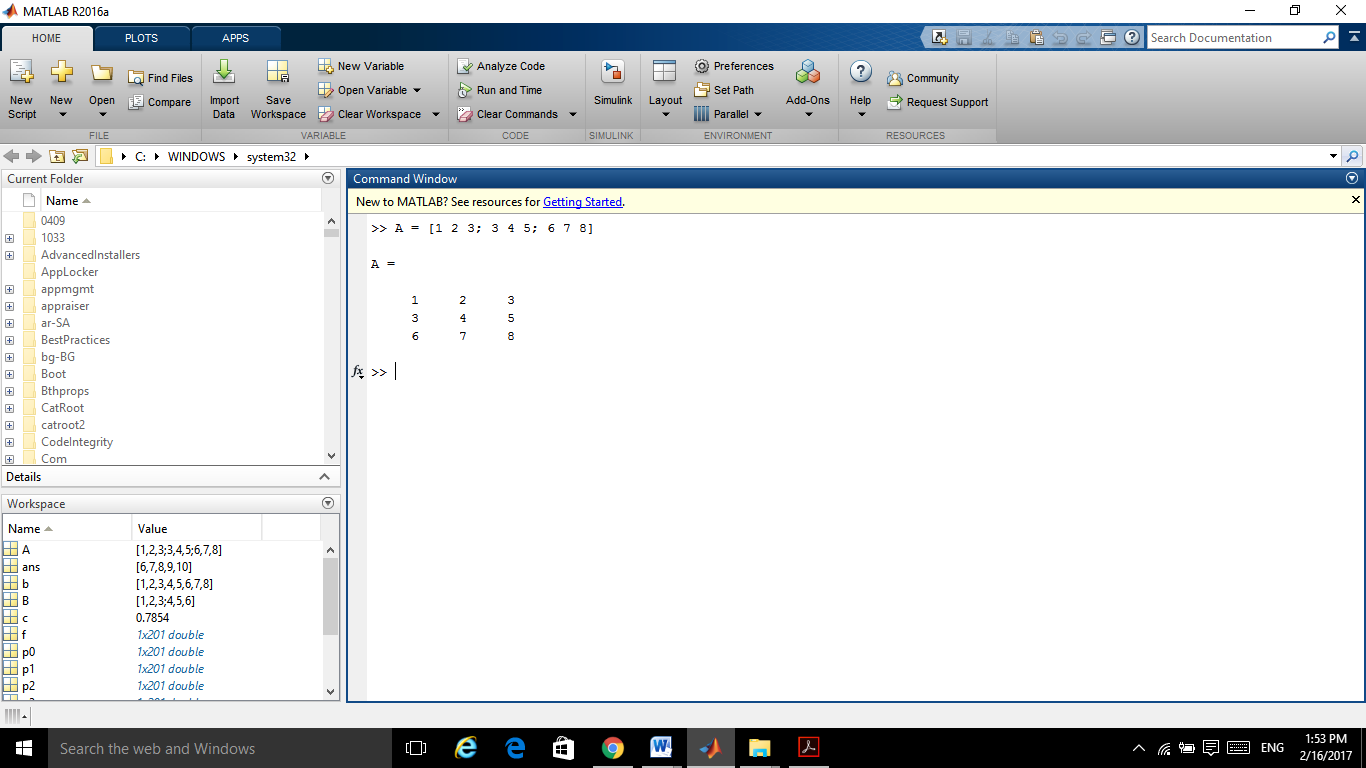
Defining a matrix is similar to defining a vector. To define a matrix A, you can treat it like a

column of row vectors.

That is, you enter each row of the matrix as a row vector (remember to separate the entries

either by commas or spaces) and you separate the rows by semicolons ( ; ).

» A = [1 2 3; 3 4 5; 6 7 8]



You can refer to a particular entry in a matrix by using parentheses. For example, the number 5

lies in the 2nd row, 3rd column of A, thus

» A(2,3)

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

A =

1 2 3

3 4 5

6 7 8

>> A(2,3)

ans =

5

**Extracting row and column of matrix**

We could even extract an entire row or column of a matrix, using the colon ( : ) as follows.

Suppose we want to get the 2 nd column of A. We basically want the elements [A(1,2) A(2,2) A(3,2)]. We type

» A(:,2)

Scalar functions

**Function Description**

sin Trigonometric sine

cos Trigonometric cosine

tan Trigonometric tangent

asin Trigonometric inverse sine

acos Trigonometric inverse cosine

atan Trigonometric inverse tangent

exp exponential

log Natural logarithm

abs Absolute value

sqrt Square root

rem reminder

floor Round towards nearest integer

**Scalar functions**

Investigate floor and ceil function using different numbers.

The trigonometric functions take as input radians. Since MATLAB uses pi for the number

π = 3.1415…

» sin(pi/2)

» cos(pi/2)

» rem(12,4)

» rem(12,5)

\_» floor(1.4)

\_» ceil(1.4)

\_» round(1.4)

**Vector Functions**

Keep in mind that all of the above commands can be used on vectors with the operation taking place element-wise. For example, if x = [0, 0.1, 0.2, . . ., 0.9, 1], then y = exp(x) will produce another vector y , of the same length as x, whose entries are given by y = [e 0 , e 0.1

, e 0.2 , . . ., e 1 ].

» x = [0:0.1:1]

» y = exp(x)

**Vector functions**

MATLAB functions operate essentially on vectors returning a scalar value. Some of these

functions are given in the table below.

**Functions Description**

max Largest component

min Smallest component

length Length of a vector

sort Sort in ascending order

sum Sum of elements

prod Product of elements

median median

mean Mean

std Standard deviation

**Matrix Functions**

**Functions Description**

eye Identity matrix

zeros Matrix of zeros

ones Matrix of ones

diag Extract or create diagonal matrix

triu Upper triangular part of matrix

tril Lower triangular part of matrix

size Size of matrix

det Determinant of matrix

inv Inverse of matrix

rank Rank of a matrix

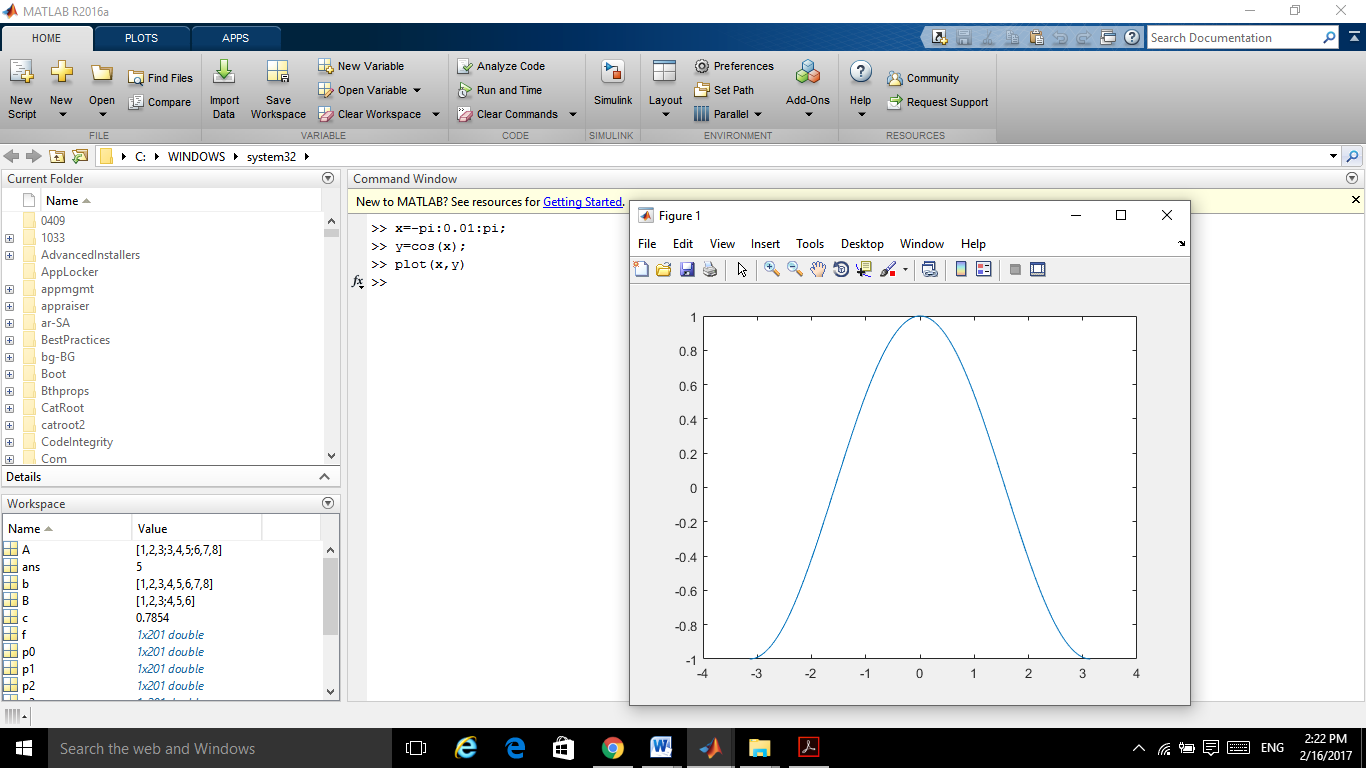
**Plot**

>> x=-pi:0.01:pi;

>> y=cos(x);

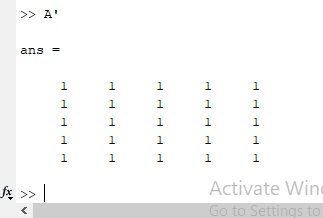
>> plot(x,y)

>>

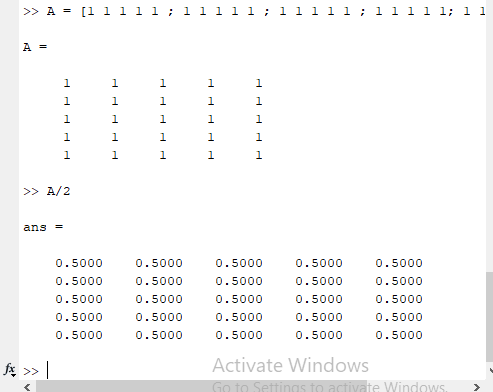


**Lab Tasks**

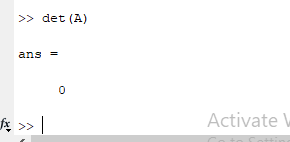
1. **Enter a matrix A of size 5\*5 with all entries non zero of your choice. (2 Marks)**
2. Get the matrix transpose.A’



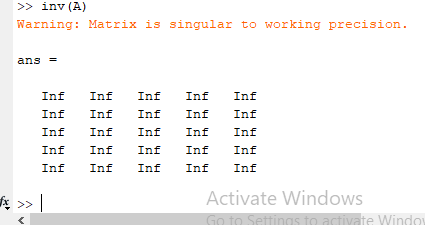
1. Divide A by 2.



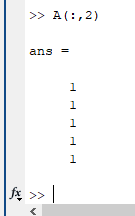
1. Get determinant of A.



1. Get inverse of A if possible.



1. Get 2nd column of A



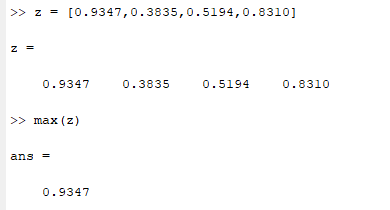
1. Get 4th row 0f A.



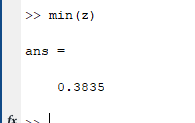
1. **z = [0.9347,0.3835,0.5194,0.8310] (2 Marks)**

**Find**

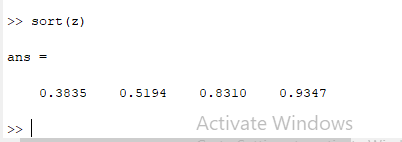
» max(z)



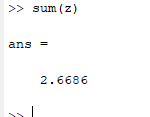
» min(z)



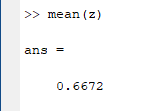
» sort(z)



» sum(z)

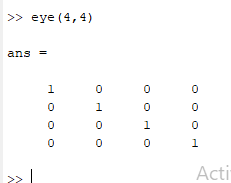


» mean(z)

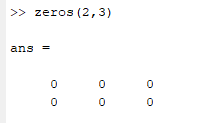


1. **Run the following commands (4 Marks)**

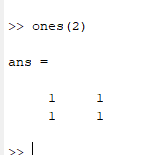
» eye(4,4)



» zeros(2,3)

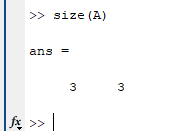


» ones(2)

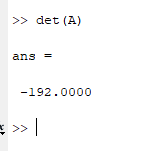


**A = [9,7,0;0,8,6;7,1,-6]**

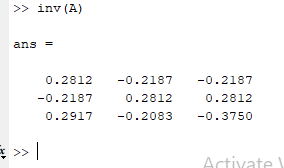
» size(A)



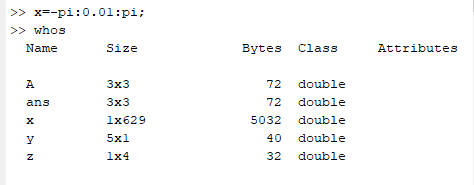
» det(A)



» inv(A)

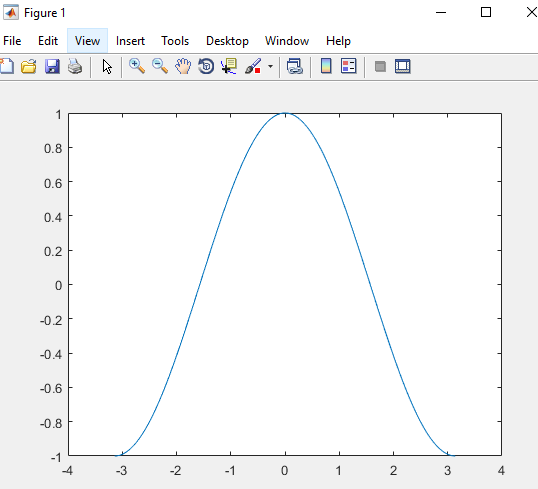


» x=-pi:0.01:pi;



» y=cos(x);

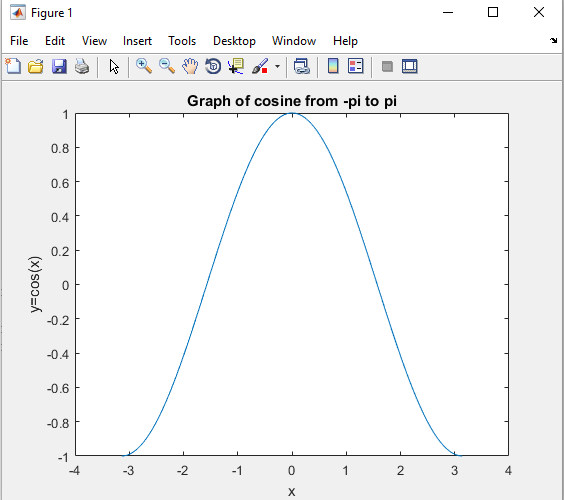
» plot(x,y)



» xlabel('x')

» ylabel('y=cos(x)')

» title('Graph of cosine from -pi to pi')

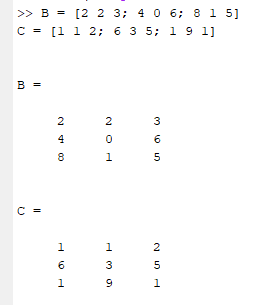


1. **We have the following two matrices (2 Marks)**

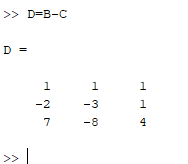
**B = [2 2 3; 4 0 6; 8 1 5]**

**C = [1 1 2; 6 3 5; 1 9 1]**

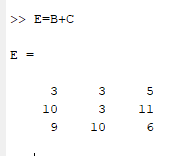
**Calculate:**



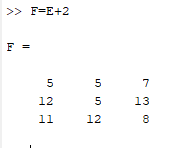
1. D = B – C



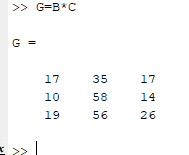
1. E = B + C



1. F= E+2



1. G=BC



(e) H=Matrix whose entries are product of corresponding entries of B and C.

