National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering

Faculty Member: Usman Ilyas **Dated: 5-2-2016**

Course/Section: BEE-6B **Semester: 4th Semester**

EE-232 Signals and Systems

Lab #1 Introduction to Matlab

Name	Reg. no.	Report Marks / 10	Viva Marks / 5	Total/15
Saad Iqbal	32903			

1.1 Lab Tasks

1.1.1 Lab Task 1:

(a) Make sure that you understand the **colon** notation. In particular, explain in words what the following MATLAB code will produce

```
a = 0 : 6
b = 2 : 4 : 17
c = 99 : -1 : 88
d = 2 : (1/9) : 4
e = pi * [ 0:0.1:2 ];
```

MATLAB CODE:

```
a = 0 : 6
b = 2 : 4 : 17
c = 99 : -1 : 88
d = 2 : (1/9) : 4
e = pi * ( 0:0.1:2 )
```

MATLAB OUTPUT:

```
a =

     0     1     2     3     4     5     6

b =

     2     6    10    14

c =

    99    98    97    96    95    94    93    92    91    90    89    88

d =

  Columns 1 through 9

    2.0000    2.1111    2.2222    2.3333    2.4444    2.5556    2.6667    2.7778    2.8889

  Columns 10 through 18

    3.0000    3.1111    3.2222    3.3333    3.4444    3.5556    3.6667    3.7778    3.8889

  Column 19

    4.0000

e =

  Columns 1 through 9

         0    0.3142    0.6283    0.9425    1.2566    1.5708    1.8850    2.1991    2.5133

  Columns 10 through 18

    2.8274    3.1416    3.4558    3.7699    4.0841    4.3982    4.7124    5.0265    5.3407

  Columns 19 through 21
```

(b) Extracting and/or inserting numbers into a vector is very easy to do. Consider the following definition of f:

f = [zeros(1,3), linspace(0,1,5), ones(1,4)]
f(4:6)
size(f)
length(f)
f(2:2:length(f))

MATLAB CODE:

```
f = [ zeros(1,3), linspace(0,1,5), ones(1,4) ]
f(4:6)
size(f)
length(f)
f(2:2:length(f))
```

Explain the results echoed from the last four lines of the above code.

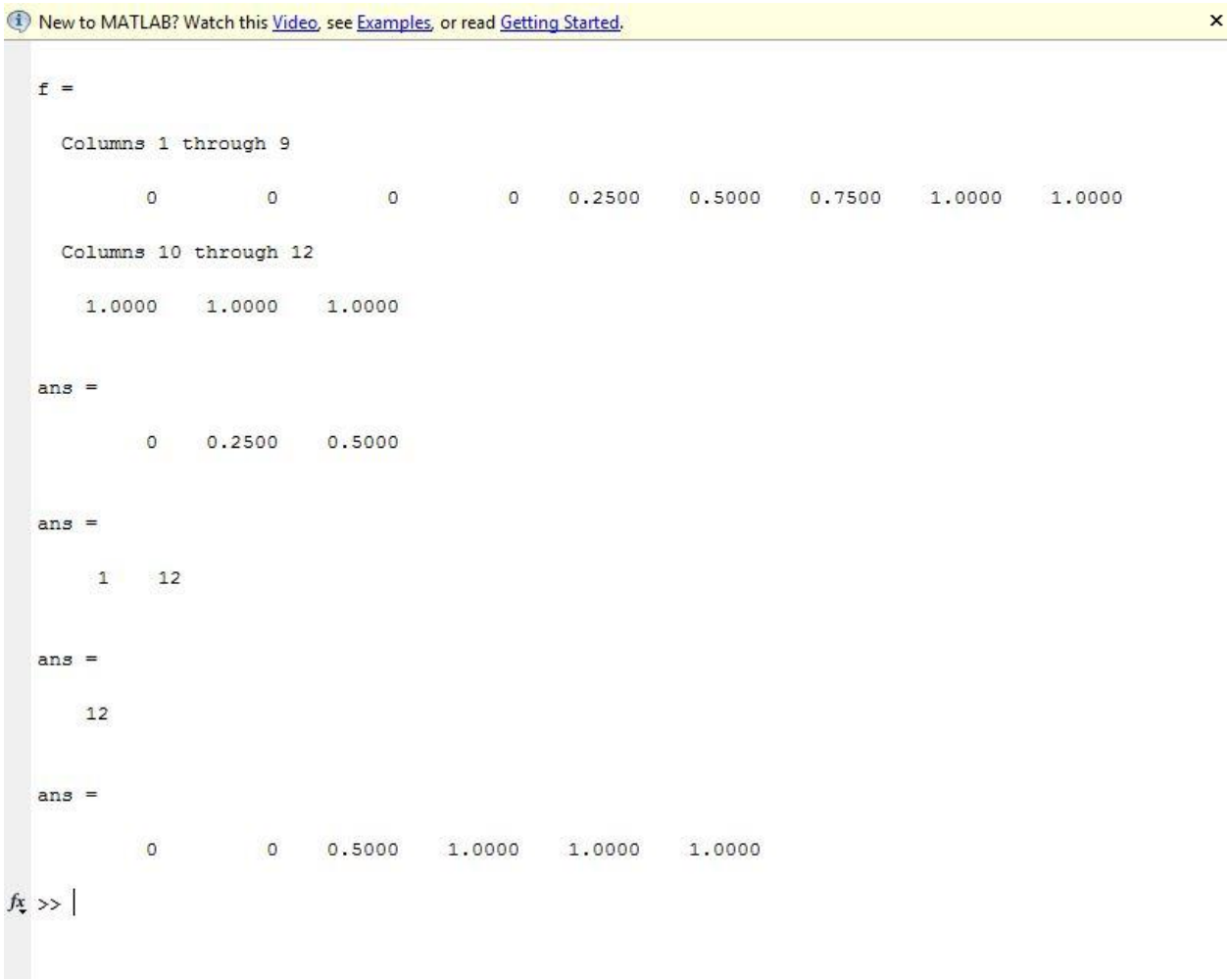
In first line, we add three zeros using `zeros(1,3)` , then equally spaced five numbers from 0 to 1 are insert into perivous array using `linspace(0,1,5)` and four ones using `ones(1,4)` .

`f(4:6)` will display number from index 4 to 6.

`size(f)` will display number of rows and columns
`length(f)` will display width(number of columns) of f

`f(2:2:length(f))` will display number from index 2 to last index with increment of 2.

MATLAB OUTPUT:



(c) Observe the result of the following assignments:
g = f; g(4:6) = pi*(1:3)

MATLAB CODE:

```
...
g = f;
g(4:6) = pi*(1:3)
```

MATLAB OUTPUT:

```
g =

  Columns 1 through 9

         0         0         0    3.1416    6.2832    9.4248    0.7500    1.0000    1.0000

  Columns 10 through 12

    1.0000    1.0000    1.0000

fx >>
```

1.1.2 **Lab Task 2:**

Now write a statement that will take the vector f defined in part (b) and replace the even indexed elements (i.e., f(2), f(4), etc) with the constant 'π^{π} ' (pi raised to the power pi) (Try: finding help on '^' operator or the function 'power'). *Use a vector replacement, not a loop.* Experiment with vectors in MATLAB. Think of the vector as a set of numbers. Try the following:
h = cos(pi*(0:11)/4) %<---comment: compute cosines
Explain how the different values of cosine are stored in the vector h. What is h(1)? Is h(0) defined?

MATLAB CODE:

```
f(2:2:length(f))=pi^pi
```

```
h = cos( pi*[0:11]/4 )
h(1)
h(0)
```

In Matlab, index starts from 1, therefore h(0) will produce error as shown in MATLAB OUTPUT.

MATLAB OUTPUT:

```
h =

  Columns 1 through 9

    1.0000    0.7071    0.0000   -0.7071   -1.0000   -0.7071   -0.0000    0.7071    1.0000

  Columns 10 through 12

    0.7071    0.0000   -0.7071


ans =

     1

Subscript indices must either be real positive integers or logicals.

Error in code1 (line 33)
h(0)

fx >>
```

1.1.3 **Lab task 3:**

Loops can be written in MATLAB, but they are NOT the most efficient way to get things done. It's better to **always avoid loops** and use the colon notation instead. The following code has a loop that computes values of the cosine function. (The index of yy() must start at 1.) Rewrite this computation without using the loop (follow the style in the previous part).

g = [ ]; %<--- initialize the yy vector to be empty
for k=-5:5
g(k+6) = cos(k*pi/3)
end
g
Explain why it is necessary to write g(k+6). What happens if you use g(k) instead?

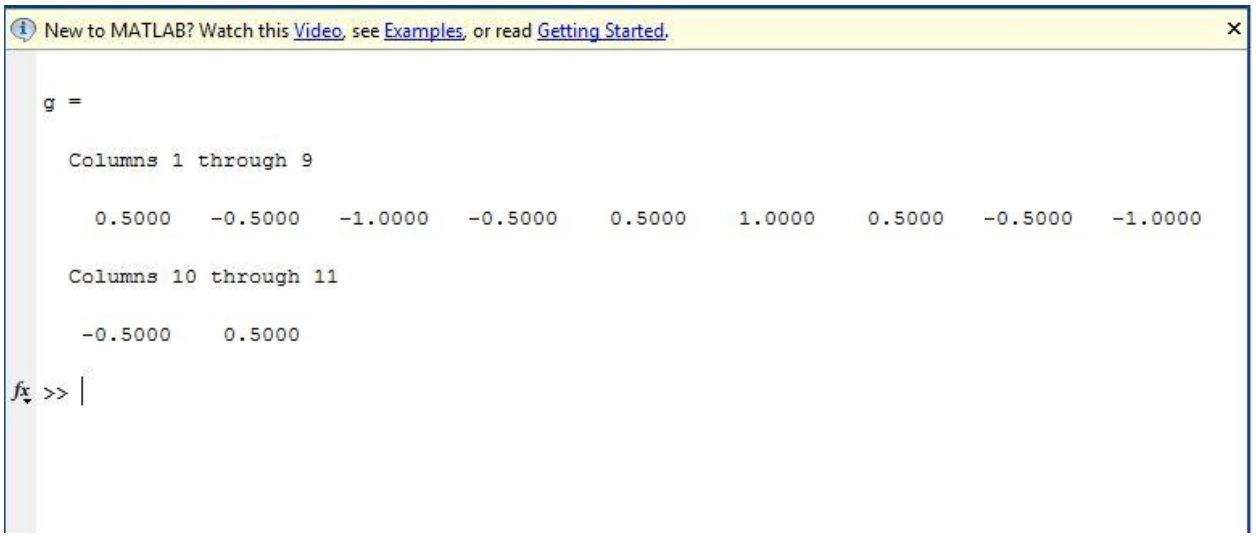
```
f(2:2:length(f))=pi^pi;
f
```

MATLAB CODE:

```
g = [ ];
for k=-5:5
g(k+6) = cos( k*pi/3 ) ;
end
g
```

As, Loop starts from -5, therefore if we use g(k),then it leads in initial loops g(-5), g(-4),…etc. As, in Matlab, we suppose to start index from 1. Thus, we use g(k+6) instead of g(k).

MATLAB OUTPUT:



1.1.4 **Lab task 4:**

Go to File > New > M –file. MATLAB editor will open up. Enter the following code in the editor and then save the file as mylab1.m

```
clear all;
clc;
t = -1 : 0.01 : 1;
x = cos( 5*pi*t );
y = 1.4*exp(j*pi/2)*exp(j*5*pi*t);
plot( t, x, 'b-', t, real(y), 'r--' ), grid on
%<--- plot a sinusoid
title('TEST PLOT of a SINUSOID')
xlabel('TIME (sec)')
```

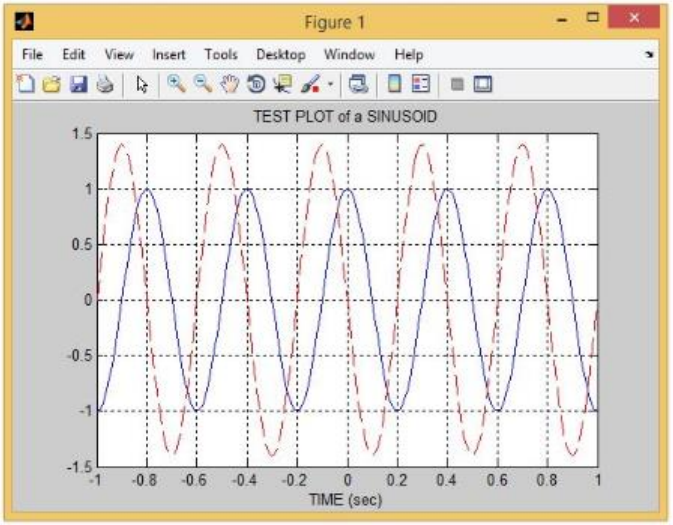
Explain why the plot of real(y) is a sinusoid. What is its phase and amplitude? Make a calculation of the phase from a time-shift measured on the plot.

MATLAB CODE:

```
t = -1 : 0.01 : 1;
x = cos( 5*pi*t );
y = 1.4*exp(j*pi/2)*exp(j*5*pi*t);
plot( t, x, 'b-', t, real(y), 'r--' ), grid on
title('TEST PLOT of a SINUSOID')
xlabel('TIME (sec)')
```

Amplitude and phase can obtain using cursor in Matlab.
Amplitude = 1.4 units
Phase = -90^0 or pi/2.

MATLAB OUTPUT:



1.1.5 Lab task 6:

Create a function "sigadd" to add two sequences 'x1' and 'x2'.
Function [y,n]=sigadd(x1,n1,x2,n2)
Where 'x1' and 'x2' are two sequences and 'n1' and 'n2' are their respective indices vectors. Add values of 'x1' and 'x2' at corresponding indices, pad zeros if length of two sequences are not same.

Suppose x1= [1 2 3 4 5 6 7 8 9] with index n1= 3:11 and x2= [2 4 6 8 10 12 14 16 18 20 22 24] with index n2=1:12. Here you can observe that the length of both the signals is not same and the indexes of both the signals are not starting from the same point. So you have to pad zeros before adding both the sequences so that the output y will have the index values starting from 1 up to 12.

Hint: You may need the loops and if else checks. Loops syntax is already discussed above and syntax of if else is given below.

MATLAB CODE:

Sigadd.m file

```
function [y,n]=sigadd(x1,n1,x2,n2)

   if length(x1)==length(n1) & length(x2)==length(n2) %check
    a1=n1(1); b1= a1+length(n1)-1;
    a2=n2(1); b2= a2+length(n2)-1; %defining variable for
                                   %initial index and final index
   if (a1>= a2 & b1<= b2)          %for x1 intial index >= to x2 and
                                   %x2 final index >= to x1
       x1 = [zeros(1,a1-a2),x1(1:length(x1)),zeros(1,b2-b1)];
       n=[a2:b2];
   elseif (a1>= a2 & b1> b2)       %for x1 intial index >= to x2 and
                                   %x1 final index >= to x2
       x1 = [zeros(1,a1-a2),x1(1:length(x1))];
       x2 = [x2(1:length(x2)),zeros(1,b2-b1)];
       n=[a2:b1];
   elseif (a1< a2 & b1<= b2)       %for x1 intial index < to x2 and
                                   %x1 final index >= to x2
       x2 = [zeros(1,a2-a1),x2(1:length(x1))];
       x1 = [x1(1:length(x1)),zeros(1,b2-b1)];
       n=[a2:b1];
   else                            %for x2 intial index >= to x1 and
                                   %x1 final index >= to x2
       x2 = [zeros(1,a2-a1),x2(1:length(x1)),zeros(1,b1-b2)];
       n=[a1:b1];
   end
   y=x1+x2;
   n
   else
       disp('ERROR');
   end
```

MATLAB OUTPUT: