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

Department of Electrical Engineering and Computer Science

Faculty Member: Dr. Ahmad Salman Dated: 6/02/2023

Semester: 6th Section: BEE 12C

EE-330 Digital Signal Processing

Lab 1: MATLAB Review - Signals & Systems Fundamentals

Group Members

		PLO4 - CLO4		PLO5 - CLO5	PLO8 - CLO6	PLO9 - CLO7
Name	**Reg. No**	**Viva / Quiz / Lab Performance**	**Analysis of data in Lab Report**	**Modern Tool Usage**	**Ethics and Safety**	**Individual and Teamwork**
		5 Marks	**5 Marks**	**5 Marks**	**5 Marks**	**5 Marks**
Danial Ahmad	331388					
Muhammad Umer	345834					
Tariq Umar	334943					

1 Table of Contents

2 Signals & Systems Fundamentals

2.1 Objectives

The purpose of this lab is to review the fundamentals of signals and systems with MATLAB, particularly:

- Signal transformations (shifting, inversion, scaling)
- Even and Odd parts of a signal
- Convolution operator-the basic property of Linear Time Invariant (LTI) Systems

2.2 Introduction

Signals and systems are essential concepts in various fields of engineering and technology. The purpose of this lab is to provide a comprehensive overview of these concepts, with a particular focus on their implementation using MATLAB. The main objectives of this lab are to review the fundamentals of signals and systems, including signal transformations such as shifting, inversion, and scaling, the distinction between even and odd parts of a signal, and the Convolution operator as a basic property of Linear Time Invariant (LTI) Systems. Through the course of this lab, we will explore the various properties of signals and systems and gain a deeper understanding of how they can be applied in practice. By the end of this lab, we will have a solid foundation in the fundamental concepts of signals and systems and be able to use MATLAB to perform various operations and analyses on signals and systems.

2.3 Software

MATLAB is a high-level programming language and numerical computing environment. Developed by MathWorks, it provides an interactive environment for numerical computation, visualization, and programming. MATLAB is widely used in various fields, including engineering, science, and finance, due to its capabilities for matrix and vector operations, implementation of algorithms, and creation of graphical representations of data.

2.4 Lab Report Instructions

All questions should be answered precisely to get maximum credit. Lab report must ensure following items:

- Lab objectives
- MATLAB codes
- Results (graphs/tables) duly commented and discussed
- Conclusion

3 Lab Procedure

3.1 Matrices/Vectors in MATLAB

a) Make sure that you understand the colon notation. Explain in words what the following MATLAB code will produce:

```
jkl = 0:6;
% This will create a vector array from 0 to 6 with a default spacing of 1.
jkl = 4:4:17;
% This will create a vector array from 4 to 17 with a spacing of 4.
jkl = 99:-1:88;
% This will create a vector array from 99 to 88 with a spacing of -1.
ttt = 2:(1/9):4;
% This will create a vector array from 2 to 4 with a spacing of 1/9.
tpi = pi * (0:0.1:2);
% This will create a vector array from 0 to 2 with a spacing of 0.1, ...
% and then multiply each element with pi.
```

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

```
xx = [zeros(1,3), linspace(0,1,5), ones(1,5)];
[s1 s2] = size(xx)
s3 = length(xx)
```

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

The size() statement returns [row column] as the output which are assigned to the variables s1 and s2. As xx is a 1 x 13 row vector; s1 = 1 and s2 = 13. The length() statement returns the maximum of the size() statement, and hence, s3 = 13 as well.

What's the difference between a length and a size statement for a matrix? To test this define a matrix X with arbitrary inputs, having multiple rows and columns and test the output of length() and size() function on it.

Answer: The size() statement returns the [row column] dimensions of the matrix, whereas length() returns the maximum of the two elements of size() statement.

c) Assigning selective values in a matrix differently. Comment on the result of the following assignments:

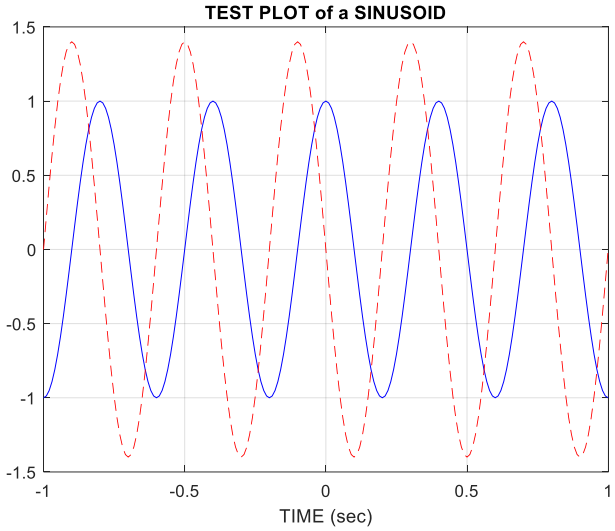
```
yy = xx;
yy(4:6) = pi*(1:3);
```

Answer: Assignment of yy(4:6) will update the elements 4, 5, and 6 with multiples of π.

3.2 Creating a (.m) file

Go to File > New > M-file. MATLAB editor will open. Enter the following code in the editor and then save the file as Namelab1.m

```
tt = -1 : 0.01 : 1;
xx = cos( 5*pi*tt );
zz = 1.4*exp(j*pi/2)*exp(j*5*pi*tt);
plot( tt, xx, 'b-', tt, real(zz), 'r--' ), grid on
%<--- plot a sinusoid
title('TEST PLOT of a SINUSOID')
xlabel('TIME (sec)')
```



3.3 Functions

It is often convenient to define functions so that they may used at multiple instances and with different inputs. Functions are a special type of M-file that can accept inputs (matrices and vectors) and may return outputs. The keyword ***function*** must appear as the first word in the M-file that defines the function, and the first line of the M-file defines how the function will pass input and output arguments. The file extension must be lower case "***m***" as in my ***func.m***. The following function has a few mistakes. Before looking at the correct one below, try to find these mistakes (there are at least three):

```
Matlab mfile [xx,tt] = badcos(ff,dur)          % Wrong function declaration
%BADCOS Function to generate a cosine wave
% xx = badcos(ff,dur)
% ff = desired frequency in Hz
% dur = duration of the waveform in seconds
tt = 0:1/(100*ff):dur; %-- gives 100 samples per period
badcos = cos(2*pi*freeq*tt);          % freeq is not defined, output is not badcos but xx
```

The corrected function should look something like:

```
function [xx, tt] = goodcos(ff, dur)
    tt = 0:1 / (100 * ff):dur; %-- gives 100 samples per period
    xx = cos(2 * pi * ff * tt);
```

3.4 Review of Basic Signals and Systems

a) Even and Odd Parts of a Signal:

Any signal *x[n]* can be decomposed into its even part and odd parts as:

$$x_e(n) = \frac{1}{2}[x(n) + x(-n)]$$

$$x_0(n) = \frac{1}{2}[x(n) - x(-n)]$$

Write a simple MATLAB code (in the form of a function) that allows you to decompose a signal into its even and odd parts.

Test your function on the following signal *x[n]* and compute its even and odd parts.

$$x[n] = \begin{cases} 2 & n = 0 \\ 5 & n = 1 \\ -1 & n = 2 \\ 4 & n = 3 \\ -5 & n = 4 \\ 0 & \text{elsewhere} \end{cases}$$

Code

```
function [x_e, x_o, n_o] = evenodd(x, n)

    x_flip = fliplr(x);
    n_flip = -1 * fliplr(n);

    x_pad = [zeros(1, length(x) - 1) x];
    x_flip_pad = [x_flip zeros(1, length(x) - 1)];
    n_o = [n_flip(1:end - 1) n];

    x_e = zeros(1, length(n_o));
    x_o = zeros(1, length(n_o));

    for idx = 1:length(n_o)
        x_e(idx) = 1/2 * (x_pad(idx) + x_flip_pad(idx));
        x_o(idx) = 1/2 * (x_pad(idx) - x_flip_pad(idx));
    end

end
```

Output

```
x_e = 0 -2.5000 2.0000 -0.5000 2.5000 2.0000 2.5000 -0.5000 2.0000
x_o = 0 2.5000 -2.0000 0.5000 -2.5000 0 2.5000 -0.5000 2.0000
n_o = -4 -3 -2 -1 0 1 2 3 4
```

b) First Order Difference Equation

Recall that one way of defining the LTI systems is through the difference equations that relate the input *x[n]* to the output *y[n]*.

Consider the first order system defined by the difference equation as follows (we'll review the discussion on how determination of order for a difference equation later):

$$y[n] = a.y[n-1] + x[n]$$

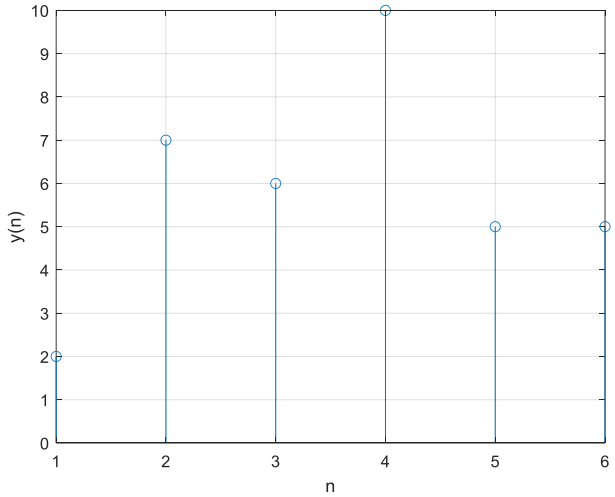
Write a function *y = diffeqn (a, x, y[-1])* which computes the output *y[n]* of the system determined by the given equation. The vectors *x[n]* contains the signal as defined in the upper part and *y[n] = 0 for n < 1*.

Code

```
function y = diffeqn(a, x, y_n1)

    y = zeros(1, length(x));
    y(1) = a * y_n1 + x(1);

    for n = 2:length(x)
        y(n) = a * y(n - 1) + x(n);
    end

end
```

Output

```
y = 2 7 6 10 5 5
```



c) Convolution of Signals

Recall that one the most convenient ways to represent an LTI system is through its impulse response *h[n]*. Once the impulse response of a system is known, the output (response) of the system to any given input can be computed using the convolution operator as:

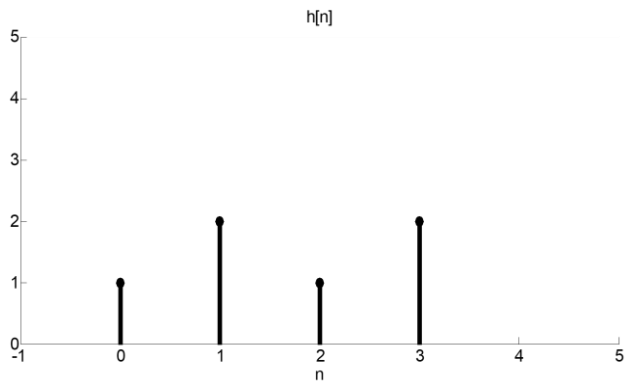
$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

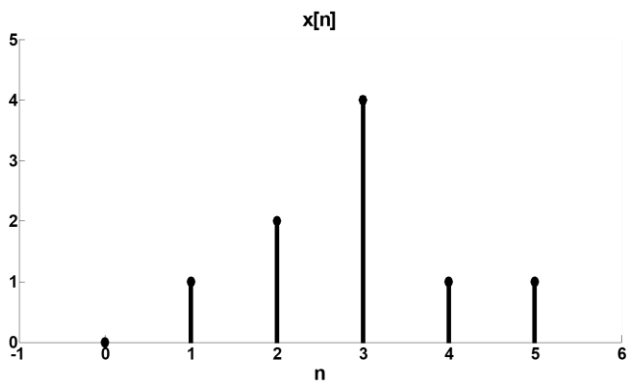
The convolution essentially involves two operations: flipping either the input signal or the impulse response (as in above equation) and then sliding the flipped signal.

1. Write your own convolution function, *myconv.m*, that computes the convolution between the two signals (or the output of passing an input signal through a system). Designate all the necessary inputs for your function, considering that the input signal and the impulse response may start at some '*n*' that is negative.

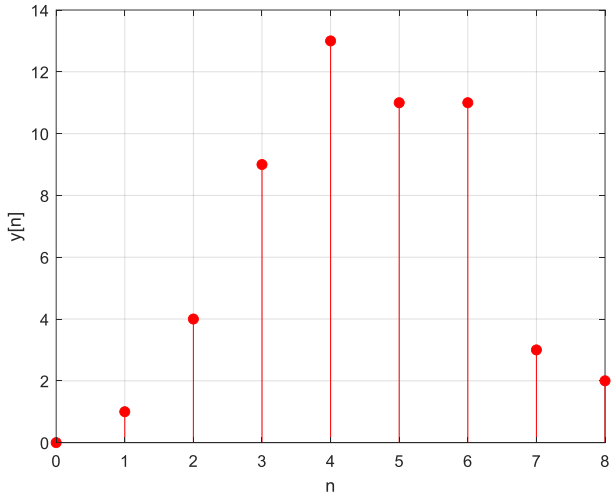
```
function y_n = myconv(x_n, h_n, x_limit, h_limit)

    if nargin < 3
        x_limit = 0;
        h_limit = 0;
    end

    len_conv = length(x_n) + length(h_n) - 1;
    start_idx = (x_limit(1)) + (h_limit(1));
    n = start_idx:len_conv - start_idx;
    y_n = zeros(1, len_conv);

    for i = 1:length(x_n)
        for j = 1:length(h_n)
            y_n(i + j - 1) = y_n(i + j - 1) + x_n(i) * h_n(j);
        end
    end

    stem(n(1:length(y_n)), y_n, 'filled', 'red');
    grid on
    ylabel('y[n]')
    xlabel('n')

end
```

2. Test your function on the signal and the impulse response provided in the figures below and verify the correctness of your function through a comparison of manual computation of the convolution for the given signal and a plot of your function's output.





```
Output
y_n = 0 1 4 9 13 11 11 3 2
```

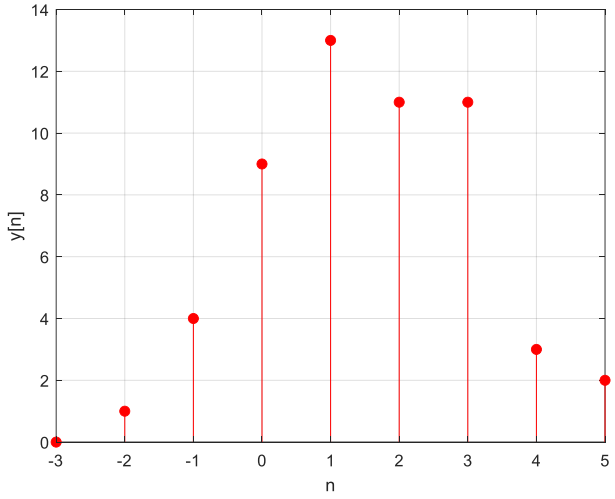


3. MATLAB has a built-in function '*conv*' that performs the same operation. Compare the results of part (ii) with the *conv* function of MATLAB.

Using the built-in 'conv' function, we get the same convolution output, i.e., [0 1 4 9 13 11 11 3 2].

4. Consider now that *x[n]* starts from *n = -1* and *h[n]* starts from *-2*. What will be the result of convolution then? Plot the corresponding output signal using the stem command and proper timing axis.

```
Output
y_n = 0 1 4 9 13 11 11 3 2
```



The result obtained is the same, however, at a different starting index.

4 Conclusion

In this lab, we have reviewed the fundamental concepts of signals and systems and explored their implementation using MATLAB. Through the course of the lab, we have covered various aspects of signals and systems, including signal transformations, the distinction between even and odd parts of a signal, and the Convolution operator as a basic property of Linear Time Invariant (LTI) Systems. By performing various operations and analyses on signals and systems, we have been able to observe the behavior of signals and systems and understand how they are affected by different operations and transformations.

In conclusion, this lab has provided us with a comprehensive overview of the fundamental concepts of signals and systems and has equipped us with the skills to apply these concepts using MATLAB. The knowledge and skills gained through this lab will be valuable for future studies and applications in the field of signals and systems.