Electronics & Telecommunication Engineering Department

Vision:

To develop the department into a full fledged centre of learning in the various fields of Electronics and Telecommunication keeping in views the latest development and making student technologically superior and ethically strong.

Mission:

To impart education and training in the field of electronics and telecommunication engineering and its ailed areas by developing competencies of the students to meet social and industrial need.

Program Educational Objectives

PEO1	Graduates will demonstrate professional engineering competencies.
PEO2	Graduates will apply engineering and science knowledge to solve technical problems with high ethical standards.
PEO3	Graduates will have effective communication skills.
PEO4	Graduates will demonstrate leadership, teamwork in their profession and adapt to current trends by engaging lifelong learning.

Program Outcomes:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review and analyze engineering problems reaching substantiated conclusions.
3. **Design/development of solutions**: Design system components or processes as per need and specification.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data in the field of electronics and telecommunication.
5. **Modern tool usage**: Select, and apply appropriate techniques, modern engineering tools, skills and equipment necessary for engineering practices.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal and safety issues.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics, responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively in both verbal and written form.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

PROGRAM SPECIFIC OUTCOMES (PSOs)

Engineering Graduates will be able to:

1. Design analog and digital system.
2. Simulate and test communication system.
3. Design and implement embedded system.

COURSE OUTCOMES

Course Name ETC273 Lab VIII: Signals And Systems **Year of Study: 2016 - 2017**

ETC273.1	Identify the signals, systems and their properties. (K)
ETC273.2	Describe the signals in time and frequency domain using Fourier series and Fourier transform. (C)
ETC273.3	Analyze systems using Convolution and Z transforms. (ANA)
ETC273.4	Analyze the signals using Energy spectral Density and Power spectral Density
ETC273.5	Simulate the signals using software tools
ETC273.6	Analyze systems using software tools

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

MASTER LIST OF EXPERIMENT

Course Name: ETC273 Lab VIII: Signals And Systems Class: SY ETC

EXPT. NO.	EXPERIMENT NAME	PAGE NO.	DATE	GRADE / REMARK	SIGN
01	To study what is ‘MATLAB’ and different Commands used in ‘MATLAB’.				
02	To generate program for discrete time signals				
03	Decomposition of Real signal into Even & Odd Components.				
04	To generate program for Convolution.				
05	Program for verification of sampling theorem.				
06	Program to find the poles and zeros of transfer function.				
07	A. Program for down sampling. B. Program for Upsampling of signal.				
08	To generate program for cross correlation & Auto correlation.				
09	Program to compute power spectrum density (PSD).				
10	Program for continuous time signals.				
11	Program for discrete time signals.				

Course Teacher **Class Teacher** **Program coordinator** **Principal**

Prepared By	Approved By
Ms. S. R. Zambre	**Dr. G. S. Sable**

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: To study what is 'MATLAB' and different Commands used in 'MATLAB'.

EXPERIMENT NO. : MIT(T)/ ETC / __________/S&S/01

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No. 1

Aim: To study what is 'MATLAB' and different Commands used in 'MATLAB'.

Theory:

What is MATLAB

MATLAB started as an interactive program for doing matrix calculations and has now grown to a high level mathematical language that can solve integrals and differential equations numerically and plot a wide variety of two and three dimensional graphs. In this subject you will mostly use it interactively and also create MATLAB scripts that carry out a sequence of commands. MATLAB also contains a programming language that is rather like Pascal.

The first version of Matlab was produced in the mid 1970s as a teaching tool. The vastly expanded Matlab is now used for mathematical calculations and simulation in companies and government labs ranging from aerospace, car design, signal analysis through to instrument control & financial analysis. Other similar programs are Mathematica and Maple. Mathematica is somewhat better at symbolic calculations but is slower at large numerical calculations.

Recent versions of Matlab also include much of the Maple symbolic calculation program.

Matlab has two different methods for executing commands: *interactive mode* and *batch mode*. In interactive mode, commands are typed (or cut-and-pasted) into the 'command window'. In batch mode, a series of commands are saved in a text file (either using Matlab's built-in editor, or another text editor such as Emacs) with a '.m' extension. The batch commands in a file are then executed by typing the name of the file at the Matlab command prompt. The advantage to using a '.m' file is that you can make small changes to your code (even in different Matlab sessions) without having to remember and retype the entire set of commands. Also, when using Matlab's built-in editor, there are simple debugging tools that can come in handy when your programs start getting large and complicated. More on writing .m files later.

In this lab you will cover the following basic things:

1. Using Matlab as a numerical calculator.
2. Entering row vectors and column vectors.
3. Entering matrices.
4. Forming matrix and vector products.
5. Doing matrix products sums etc.
6. Using Matlab to solve linear equations.
7. Matlab functions that operate on arrays.
8. Plotting basic graphs using Matlab.

Prepared By	Approved By
Ms. S. R. Zambre	**Dr. G. S. Sable**

Scalar Variables and Arithmetic Operators

Scalar variables are assigned in the obvious way:

```
>>x=7;
x=7
```

The >> is Matlab's command prompt. Notice that Matlab echos back to you the value of the assignment you have made. This can be helpful occasionally, but will generally be unwieldy when working with vectors. By placing a semicolon at the end of a statement, you can suppress this verbose behavior.

```
>> x = 7;
```

Variables in Matlab follow the usual naming conventions. Any combination of letters, numbers and underscore symbols ('_') can be used, as long as the first character is a letter. Note also that variable names are case sensitive ('X' is different from 'x').

All of the expected scalar arithmetic operators are available:

```
>> 2*x
ans =
   14

>> x^2
ans =
   49
```

Notice that the multiply operation is not implied as it is in some other computational environments and the '*' operator has to be specified (typing '>>2x' will result in an error). This is also a good time to point out that Matlab remembers the commands that you are entering and you can use the up-arrow button to scroll through them and edit them for re-entry.

This is very handy, especially when you are repeatedly making minor changes to a long command line. There are a few predefined variables in Matlab that you will use often: pi, i, and j. Both i and j are defined to be the square-root of -1 and pi is defined as the usual 3.1416... . These variables can be assigned other values, so the statement would only remove the variable named 'x' from Matlab's memory.

```
>> pi = 4;
```

Care is needed to avoid changing a predefined variable and then forgetting about that change later. That may seem unreasonable with `pi`, but `i` and `j` are both natural variables to use as indices and they are often changed. A useful command is the '`clear`', or '`clear all`' command. Typing either of these at the command prompt will remove all current variables from Matlab's memory and reset predefined variables to their original values. The `clear` command can also remove one specific variable from memory. The command

```
>> clear x
```

would only remove the variable named 'x' from Matlab's memory

Matrix Variables and Arithmetic Operators

Another useful Matlab command is 'whos', which will report the names and dimensions of all variables currently in Matlab's memory. After typing the command at the prompt we see:

```
>> whos

Name     Size       Bytes    Class
x         1x1         8      double array
```

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

Grand total is 1 elements using 8 bytes

The important thing to note right now is that the size is given as '1x1'. Matlab is really designed to work with vectors and matrices, and a scalar variable is just a special case of a 1x1 dimensional vector. To assign a vector containing the first 5 integers to the variable x, we could type this command:

```
>> x = [1 2 3 2^2 2*3-1]
x =
   1    2    3    4    5
```

4   5   6

We won't have much occasion to operate on matrices that are higher dimension, but if you wanted to create a 2-D matrix you could use a command something like:

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

To create larger vectors than the toy examples above (say, the integers up to 100), we would need to type a lot of numbers. Not surprisingly, there are easier ways built in to create vectors. To create the same 5-element vector we did above, we could also type:

```
>> x = [1:5]
x =
     1     2     3     4     5
```

The colon operator creates vectors of equally spaced elements given a beginning point, and maximum ending point and the step size in between elements. Specifically, `[b:s:e]` creates the vector `[b b+s b+2*s ... e]`. If no step size is specified (as in the example above), a step of 1 is assumed. So, the command `[1:2:10]` would create the vector of odd integers less than 10, `[1 3 5 7 9]` and the command `[1:3:10]` would create the vector of elements `[1 4 7 10]`. Let's create the vector of odd elements mentioned above:

```
>> x_odd = [1:2:10]
x_odd =
     1     3     5     7     9
```

You can access any element by indexing the vector name with parenthesis (indexing starts from one, not from zero as in many other programming languages). For instance, to access the 3rd element, we would type:

```
>> x_odd(3)
ans =
     5
```

We can also access a range of elements (a subset of the original vector) by using the colon operator and giving a starting and ending index.

```
>> x_odd(2:4)
ans =
     3     5     7
```

If we want to do simple arithmetic on a vector and a scalar, the expected things happen,

```
>> 3+[1 2 3]
ans =
     4     5     6
```

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

```
>> 3*[1 2 3]
ans =
     3     6     9
```

and the addition or subtraction of matrices is possible as long as they are the same size:

```
>> [1 2 3]+[4 5 6]
ans =
     5     7     9
```

The operators '*' and '/' actually represent matrix multiplication and division which is not typically what we will need in this course. However, a common task will be to form a new vector by multiplying (or dividing) the elements of two vectors together, and the special operators '.*' and './' serve that purpose.

```
>> [1 2 3].*[4 5 6]
ans =
     4    10    18

>> [1 2 3]./[4 5 6]
ans =
    0.2500    0.4000    0.5000
```

Beware that the operator '^' is a shortcut for repeated *matrix* multiplications ('*'), whereas the operator '.^' is the shortcut for repeated *element-by-element* multiplications ('.*'). So, to square all of the elements in a vector, we would use

```
>> [1 2 3].^2
ans =
     1     4     9
```

Built-in Commands

Matlab has many built-in commands to do both elementary mathematical operations and also complex scientific calculations. The 'help' command will be useful in learning about built-in commands. By typing help at the command prompt, you are given a list of the different categories that Matlab commands fall into (e.g., general, elementary matrix operations, elementary math functions, graphics, etc.). Notice that there are probably even specific toolboxes included with your Matlab package for performing computations for disciplines such as signal processing. To see all of the commands under a certain topic, type 'help topic'. To get a description of a specific command, type 'help command'.

We'll look at a couple of example commands. First, the square-root command, sqrt. To calculate the square root of a number, type:

```
>> sqrt(2)
ans =
    1.4142
```

Again, to illustrate that Matlab understands complex numbers, we can have it calculate the root of a negative number:

```
>> sqrt(-9)
ans =
        0 + 3.0000i
```

Many Matlab commands that operate on scalars will also operate element-by-element if you give it a vector. For instance: represented by a vector. Because Matlab allows functions like `sin` and `sqrt` (as well as many others) to operate on vectors, many of the signal definitions and calculation we would like to perform are very straight-forward. We will illustrate this a little more explicitly in the next section.

```
>> sqrt([1 2 4 6])
ans =
```

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

```
    1.0000    1.4142    2.0000    2.4495

>> sqrt([1:8])
ans =
    1.0000    1.4142    1.7321    2.0000    2.2361    2.4495    2.6458    2.8284
```

Another useful command is `sin` function, which also operates on a vector.

```
>> sin([pi/4 pi/2 pi])
ans =
  0.7071    1.0000    0.0000
```

It is important to realize that digital signals are really just a collection of points and can be

It is important to realize that digital signals are really just a collection of points and can be represented by a vector. Because Matlab allows functions like sin and sqrt (as well as many others) to operate on vectors, many of the signal definitions and calculation we would like to perform are very straight-forward. We will illustrate this a little more explicitly in the next section.

Signals, Plotting and Batch Operation

There are a few commands necessary to do basic plotting of these signals. The figure command will bring up a new figure window and the close command closes a specific window ('close all' closes all open windows). The subplot(r, c, p) command allows many plots to be 'tiled' into r rows and c columns on one figure window. After this command is issued, any succeeding commands will affect the p^{th} tile. The plot(x,y) command will then plot the equal-length vectors x and y on the appropriate axis and in the obvious way. The two sinewaves created above are then plotted along with the sum of the two signals in separate tiles with the commands:

MATLAB Program:

```
%===========================START===========================================
%% Clear previous Data (if required)
clc;
close all;
clear all;
%% define function x & y
x=[0:0.1:1];
y=x.*sin(x);
%% plot the first graphs
subplot(2,2,1)
plot(x,'r')
hold on
plot(y,'--')
%% label the graps
title('plot of xsin(x) Vs x')
grid on
xlabel('x')
ylabel('y')
legend('x','y')
%% plot the first graphs
subplot(2,2,2)
plot(x,'r')
hold on
plot(y,'--')
%% label the graps
title('plot of xsin(x) Vs x')
grid on
xlabel('x')
ylabel('y')
legend('x','y')%% plot the first graphs
subplot(2,2,3)
plot(x,'r')
hold on
plot(y,'--')
%% label the graps
title('plot of xsin(x) Vs x')
grid on
xlabel('x')
```

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

```
ylabel('y')
legend('x','y')%% plot the first graphs
subplot(2,2,4)
plot(x,'r')
hold on
plot(y,'--'

%% label the graps
title('plot of xsin(x) Vs x')
grid on
xlabel('x')
ylabel('y')
legend('x','y')
%================================End of program==========================
```

Conclusion:

____

____

____

____

Questions:

1.

____

____

____

____

____

____

2.

____

____

____

____

____

____

3.

____

____

____

____

____

____

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: To generate program for discrete time signals

EXPERIMENT NO. : MIT(T)/ ETC / __________/S&S/02

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No. 2

Aim: Write a MATLAB program for plotting basic discrete time signals
.

Theory: Different types of basic discrete time signals are as follows:

1] Unit Impulse signal

It is denoted as d(n) and defined as

d(n) = 1 for n=0
= 0 otherwise

Also known as unit impulse signal. The amplitude of signal is zero everywhere, Except at n=0 where its value is unity.

2] Unit Step signal

It is denoted by u(n) and defined as

U(n) = 1 for n>_o
=0 for n<0

3] Unit Ramp signal

It is denoted as Ur(n) and defined as

Ur(n) = n for n>_o
=0 for n<0

4] The Exponential signal

It is a sequence of form

X(n) = a^n

Further if 0<a<1 exponentially decaying function
a>1 exponentially increasing function

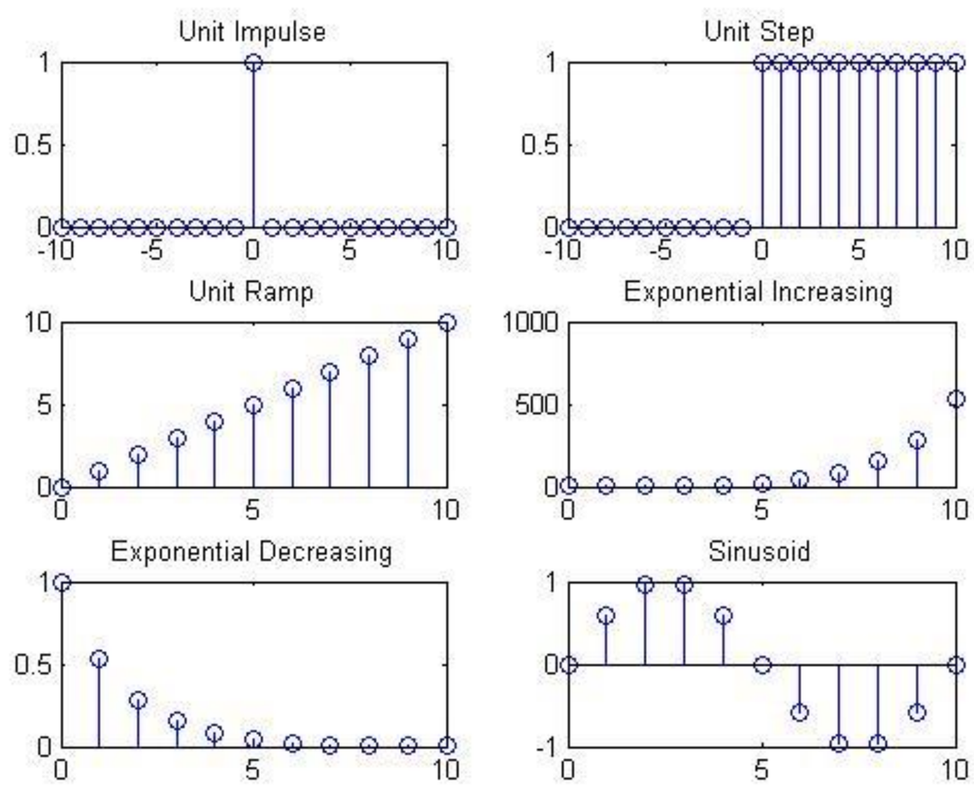
Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

Program for Discrete Time Signals
<u>Program</u>

```
% ==================== START ========================
%% Aim:- To write a MATLAB program for plotting basic discrete time sequence.

% Program:-

clc;
close all;
clear all;
x=-10:10;  %define x=[-10,-9,-8,...10]
y1=zeros(1,21);  %define y1=[0,0,0...0]

y1(1,11)=1;  %define y1=[1,1,1,1,...1,0,0,0...0]

y2=zeros(1,21);  %define y2=[0,0,0,0...0]

y2(1,11:21)=1;  %define y2=[0,0,0,...0,1,1,1...1]

y3=0:10;  %define y3=[0,1,2,3,...10]

y4=exp(0:2*pi/10:2*pi);  %define y4=e[0,pi/5,pi/10,...2pi]

y5=exp(-(0:2*pi/10:2*pi));  %define y5=e=[0,-pi/5,pi/10,...-2pi]

y6=sin(0:2*pi/10:2*pi);  % define y6=sin[0,pi/5,pi/10,...2pi]

subplot(2,3,1)

stem(x,y1)  %2-dimensional plot x vs y1

title('Discrete impluse function');  % name of figure

xlabel('Time');  % scale of x axis

ylabel('Amplitude');  % scale of y axis

subplot(2,3,2)

stem(x,y2)  %2-dimensional plot of x vs y2

title('Discrete step Function');  %name of figure

xlabel('Time');  %scale x-axis

ylabel('Amplitude')

subplot(2,3,3)

stem(y3,y3)  % plot of y3 vs y3

title('Discrete Ramp Function');  % name of figure

xlabel('Time');  % scale x axis

ylabel('Amplitude');  % scale y axis
```

```
subplot(2,3,4)

stem(y3,y4)  % plot y3 vs y4

title('Discrete Exponentially Increasing Function');  % name of figure

xlabel('Time');  % scale x axis

ylabel('Amplitude');  % scale y axis

subplot(2,3,5)

stem(y3,y5)

title('Discrete Exponentially Decaying Function');   %name of figure

xlabel('Time');   % scale x axis

ylabel('Amplitude');  % scale y axis

subplot(2,3,6)

stem(y3,y6)   % plot y3 vs y6

title('Discrete Sine Function');  % name of figure

xlabel('Time');  % scale x axis

ylabel('Amplitude');  % scale y axis
```

OUTPUT:



Prepared By	Approved By
Ms. S. R. Zambre	**Dr. G. S. Sable**

Conclusion:

Questions:

1.

2.

3.

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

Experiment No.3

Aim: Write a MATLAB program for the Decomposition of real signal into even & odd components

Theory:

Even and odd signals bear some important symmetry properties. Under reversal of independent variable, these signals either remain the same (even signal) or get reflected or flipped (odd signal) about the horizontal axis. Equations or definitions (1) and (2) mathematically express these properties for both continuous and discrete time cases.

Even Signals: $x(t) = x(-t)$, $x[n] = x[-n]$ ------------ (1)

Odd Signals: $x(t) = -x(-t)$, $x[n] = -x[-n]$ ---------- (2)

Even and Odd Functions

A function, f , is even (or symmetric) when

$$f(x) = f(-x)$$

A function, f , is odd (or antisymmetric) when

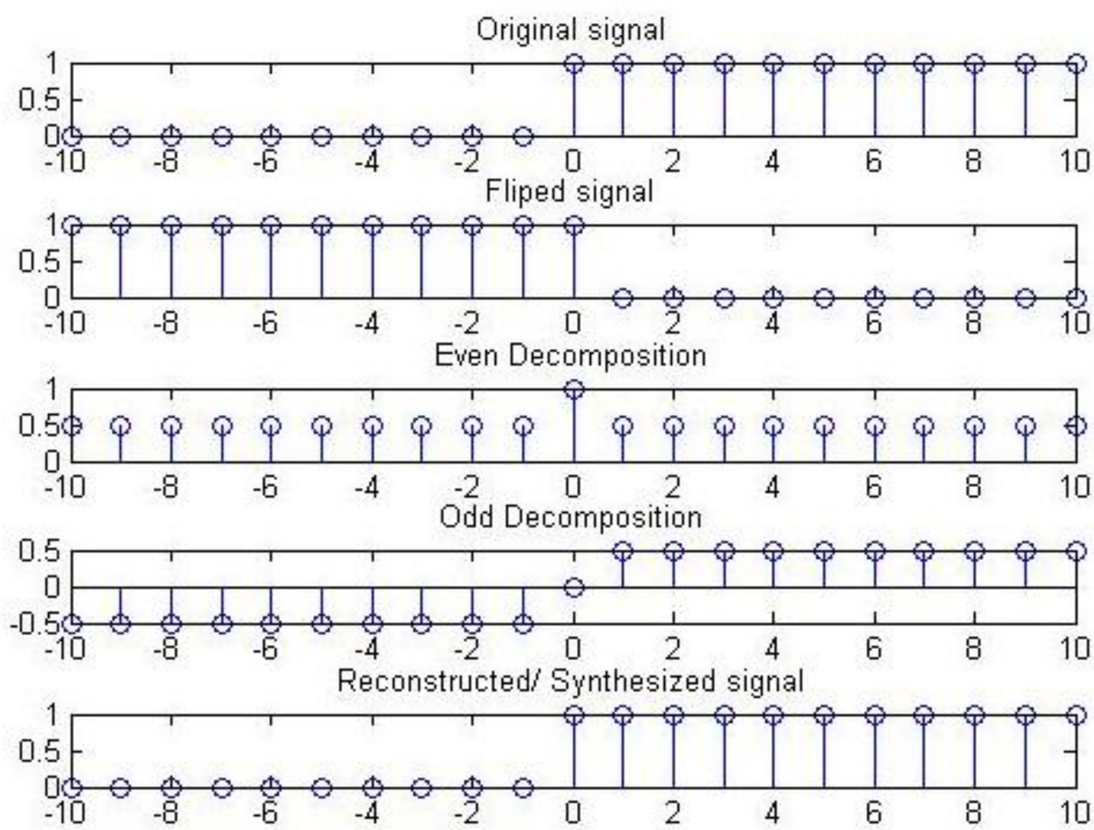
$$f(x) = -f(-x)$$

Program

```
%% Clear previous Data (if required)

clc;

close all;

clear all;

x = -10:10; % Time axis

y = zeros(1,21);

y(1,11:21) = 1; % Step Signal

e = (y + fliplr(y))/2; % Even decomposition

o = (y - fliplr(y))/2; % Odd decomposition %% Plot desired Signals
```

```
subplot(5,1,1);

stem(x,y);

title('Original signal'); % Plot original signal

subplot(5,1,2);

stem(x,fliplr(y));

title('Fliped signal'); % Its Fliped ver

subplot(5,1,3);

stem(x,e);

title('Even Decomposition');

subplot(5,1,4);

stem(x,o);

title('Odd Decomposition');

subplot(5,1,5);

stem(x,e+o);

title('Reconstructed/ Synthesized signal');

% ================================ END =======================
```


OUTPUT:



Conclusion:

Questions:

1.

2.

3.

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: To generate program for Convolution.

EXPERIMENT NO. : MIT(T)/ ETC / __________/S&S/04

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No. 4

Aim: Write a MATLAB program for calculating 1-D linear convolution.
.

Theory: There are two types of Convolution:-
1) Linear Convolution
2) Circular Convolution

- The output y[n] of a LTI (linear time invariant) system can be obtained by convolving the input x[n] with the system's impulse response h[n].
- The convolution sum is $y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] = \sum_{k=-\infty}^{+\infty} x[n-k]h[k]$
- x[n] and h[n] can be both finite or infinite duration sequences.
- If both the sequences are of finite duration, then we can use the MATLAB function '***conv***' to evaluate the convolution sum to obtain the output y[n].
- The length of y[n] = xlength + hlength -1.

Algorithm:
1. Input the two sequences as xn, hn
2. Convolve both to get output yn.
3. Plot the sequences.

1) Program for Linear convolution

Program(Method1)

```
% ============ START =====================
clc; clear all; close all; %% Clear previous Data (if required)
%% Parameters
x = [1 2 3 4]; % First Signal
h = [5 6 7 8]; % Second Signal
%% Calculate Linear Convolution Length
l1 = length(x); l2 = length(h);
l = l1+l2-1;
x = [x zeros(1,l2-1)]; % Zero Padding
h = [h zeros(1,l1-1)]; % Zero Padding
y = zeros(1,l);
%% Linear Convolution
for i = 1:1:l
for k = 1:1:i
y(i) = y(i)+x(k)*h(i+1-k);
end
end
disp(['Convolution: ', num2str(y)]);
%=============END=======================
```

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

OUTPUT:

Convolution: 5 16 34 60 61 52 32

Program(Method2):

```
% program for linear convolution
 clc;
clear all;
close all;
x=input('Enter the first sequence');
h=input('Enter the second sequence');
y=conv(x,h);
figure(1);
subplot(3,1,1);
stem(x);
ylabel('amplitude----->');
xlabel('a(n)------>');
subplot(3,1,2);
stem(h);
ylabel('amplitude----->');
xlabel('b(n)-------->');
subplot(3,1,3);
stem(y);
ylabel('amplitude---->');
xlabel('c(n)------->');
disp('The result signal is');
disp(y)
```

OUTPUT:

```
Enter the first sequence[1 2 3 4]
Enter the second sequence[3 4 5 7]
The result signal is
    3    10    22    41    45    41    28
```

Conclusion:

____
____
____
____

Questions:

1.

____
____
____
____
____
____

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

2.

____

____

____

____

____

____

3.

____

____

____

____

____

____

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

	Maharashtra Institute of Technology, Aurangabad	LABORATORY MANUAL

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: Program for verification of sampling theorem.

EXPERIMENT NO. : **MIT(T)/ ETC /** __________/S&S/05

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No. 5

Aim: Write a MATLAB program for verification of sampling theorem

.

Theory:

- Sampling is a process of converting a continuous time signal (analog signal) x(t) into a discrete time signal x[n], which is represented as a sequence of numbers. (A/D converter)
- Converting back x[n] into analog (resulting in $\hat{x}(t)$) is the process of reconstruction. (D/A converter)
- Techniques for reconstruction-(i) ZOH (zero order hold) interpolation results in a staircase waveform, is implemented by MATLAB plotting function ***stairs(n,x),*** (ii) FOH (first order hold) where the adjacent samples are joined by straight lines is implemented by MATLAB plotting function ***plot(n,x),***(iii) spline interpolation, etc.
- For $\hat{x}(t)$ to be exactly the same as x(t), sampling theorem in the generation of x(n) from x(t) is used. The sampling frequency fs determine the spacing between samples.
- Aliasing-A high frequency signal is converted to a lower frequency, results due to under sampling. Though it is undesirable in ADCs, it finds practical applications in stroboscope and sampling oscilloscopes.

Algorithm:

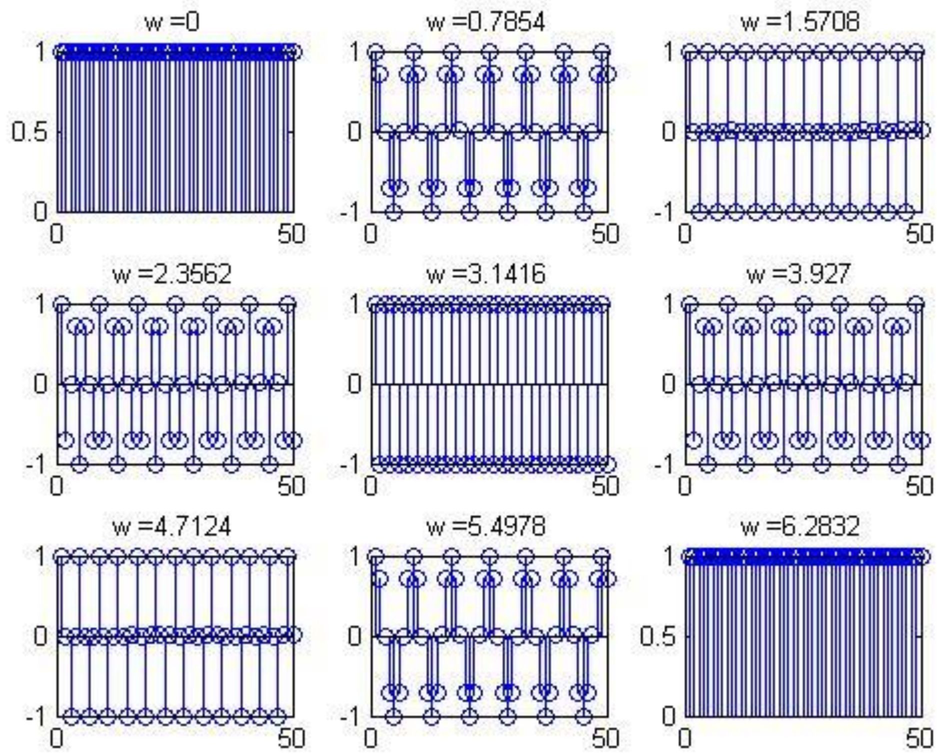
1. Input the desired frequency fd (for which sampling theorem is to be verified).
2. Generate an analog signal xt of frequency fd for comparison.
3. Generate oversampled, nyquist & under sampled discrete time signals.
4. Plot the waveforms and hence prove sampling theorem.

MATLAB PROGRAM (Range of digital frequency of a sinusoid.**)**

```
% ======================== START =========================
%% Clear previous Data (if required)
clc; clear all; close all;
%% Define required Parameters & Plot Signals
w = 0:pi/4:2*pi; % Omega ( For unique sequences maximium value is pi )
n = 0:49; % Number of samples = 50
t = 0; % Theta
for k = 1:9
subplot(3,3,k); stem(cos(w(k)*n + t));
title(strcat('w =',num2str(w(k))));
end
% ======================== END =========================
```

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

OUTPUT:



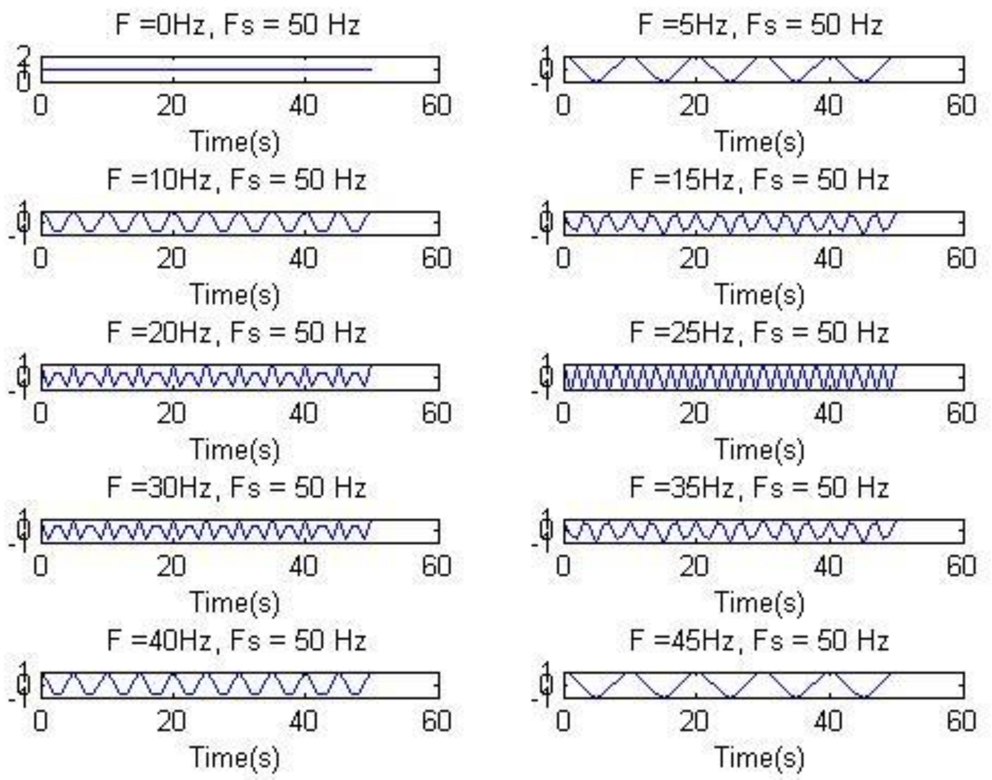
MATLAB PROGRAM(verification of sampling theorem)

```
% ================== START ======================
%% Clear previous Data (if required)
clc; clear all; close all;
%% Specifications
F = 0:5:45; % Frequency varied from 0 to 45 Hz
Fs = 50; % Sampling Frequency
n = 0:Fs; % Sampling instant
f = -Fs/2:Fs/2; % Frequency Axis (Symmetric)
%% Calculate & Plot desired Signals
for k = 1:size(F,2)
y = cos(2*pi*F(k)/Fs*n);
figure(1);
subplot(5,2,k); plot(n,y); xlabel('Time(s)');
title(strcat('F =',num2str(F(k)),'Hz, Fs = 50 Hz'));
figure(2);
subplot(5,2,k); stem(f,abs(fftshift(fft(y)))); xlabel('Frequency(Hz)');
end
% ================= END =======================
```

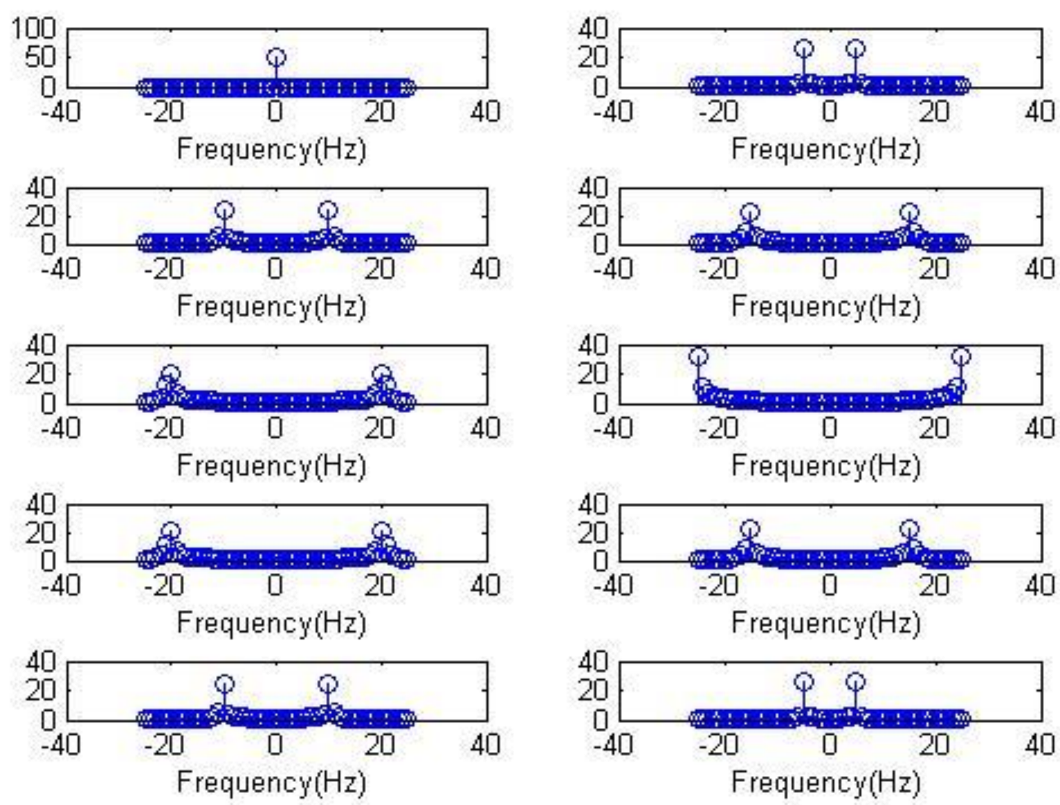
Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

OUTPUT

1) Continuous Time:



2) Discrete Time:



Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

Conclusion:

Questions:

1.

2.

3.

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

Experiment No. 6

Aim: Write a MATLAB program to find the poles and zeros of transfer function

Theory:

Poles and **Zeros** of a transfer function are the frequencies for which the value of the denominator and numerator of transfer function becomes zero respectively. The values of the poles and the zeros of a system determine whether the system is stable, and how well the system performs.

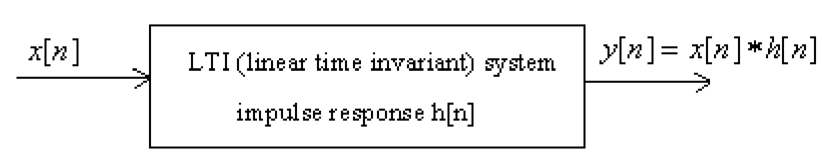


Fig.6.1 A LTI system

- A discrete time LTI system (also called digital filters) as shown in Fig.6.1 is represented by
  - A linear constant coefficient difference equation, for example,
    $y[n] + a_1 y[n-1] - a_2 y[n-2] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2];$
  - A system function H (z) (obtained by applying Z transform to the difference equation).
    $$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$
- Given the difference equation or H (z), the impulse response of the LTI system is found using **filter or impz** MATLAB functions.

MATLAB Programs:

```
clear all;
close all;
clc;
disp('Difference Equation of a digital system');
N=input('Desired Impulse response length = ');
b=input('Coefficients of x[n] terms = ');
a=input('Coefficients of y[n] terms = ');
h=impz(b,a,N);
disp('Impulse response of the system is h = ');
disp(h);
n=0:1:N-1;
figure(1);
stem(n,h);
xlabel('time index');
ylabel('h[n]');
title('Impulse response');
figure(2);
zplane(b,a);
xlabel('Real part');
ylabel('Imaginary part');
title('Poles and Zeros of H[z] in Z-plane');
```

Result:

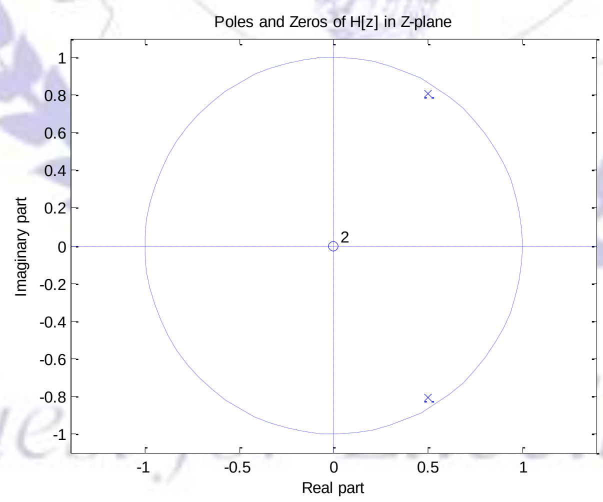
[Given y(n)-y(n-1)+0.9y(n-2)= x(n)]

Difference Equation of a digital system

Desired Impulse response length = 100

Coefficients of x[n] terms = 1

Coefficients of y[n] terms = [1 -1 0.9]



Conclusion:

____

____

____

____

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

Questions:

1.

2.

3.

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: Program for down sampling.			
EXPERIMENT NO. : MIT(T)/ ETC / __________/S&S/07A			
Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No. 7(A)

Aim: - Write MATLAB based program for down sampling.

Theory:-

DOWNSAMPLE

Downsampling (or subsampling) is the process of reducing the sampling rate of a signal by integer factor. This is usually done to reduce the data rate or the size of the data.

Syntax

y = downsample(x,n)
y = downsample(x,n,phase)

Description

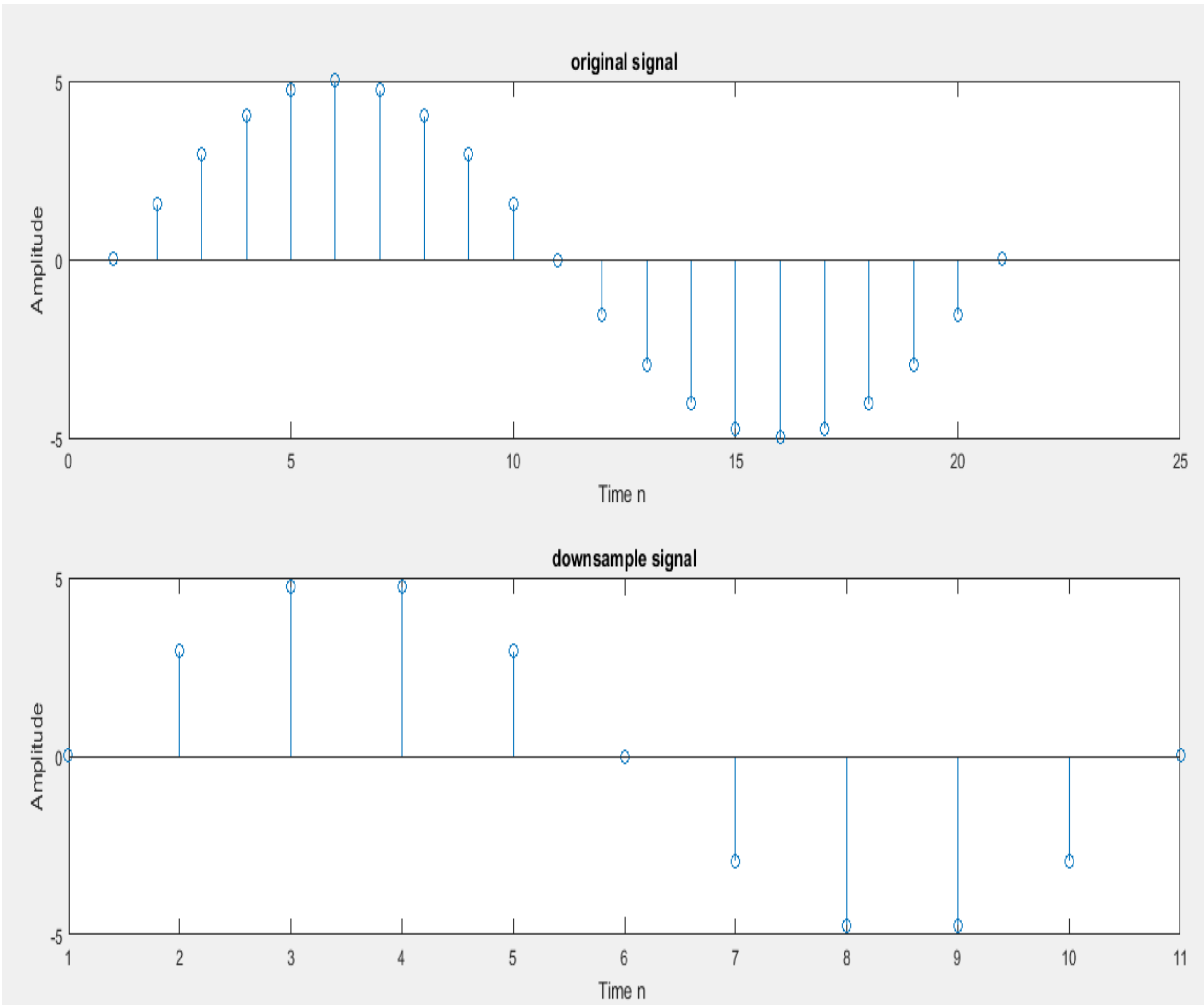
y = downsample(x,n) decreases the sampling rate of x by keeping every n^{th} sample starting with the first sample. x can be a vector or a matrix. If x is a matrix, each column is considered a separate sequence.

y = downsample(x,n,phase) specifies the number of samples by which to offset the downsampled sequence. Phase must be an integer from 0 to n-1.

PROGRAM:-

```
clc; close all; clear all;
A=5; F=50;
T=1/F;
D= input(' Enter the downsampling factor');
 n=0:T/20:T;
y1=A*sin(2*pi*n*F);
subplot(2,1,1);
stem(y1);
title('original signal');
xlabel ('Time n ');
ylabel('Amplitude');
 y2=downsample(y1,D);
subplot(2,1,2);
stem(y2);
title('downsample signal');
xlabel ('Time n ');
ylabel('Amplitude');
```

OUTPUT:-



Downsampling factor =2

Conclusion:

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

Questions:

1.

____

____

____

____

____

____

2.

____

____

____

____

____

____

3.

____

____

____

____

____

____

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

EXPERIMENT TITLE: Program for Upsampling of signal.

EXPERIMENT NO. : MIT(T)/ ETC / __________/S&S/07B			
Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No.7(B)

Aim: - Write MATLAB based program for Upsampling of signal.

Theory:-

Upsample

Upsampling is the process of increasing the sampling rate of a signal by integer factor. This is usually done to increase the data rate or the size of the data.

Syntax

y = upsample(x,n)
y = upsample(x,n,phase)

Description

y = upsample(x,n) increases the sampling rate of x by inserting n-1 zeros between samples. x can be a vector or a matrix. If x is a matrix, each column is considered a separate sequence. The upsampled y has x*n samples.

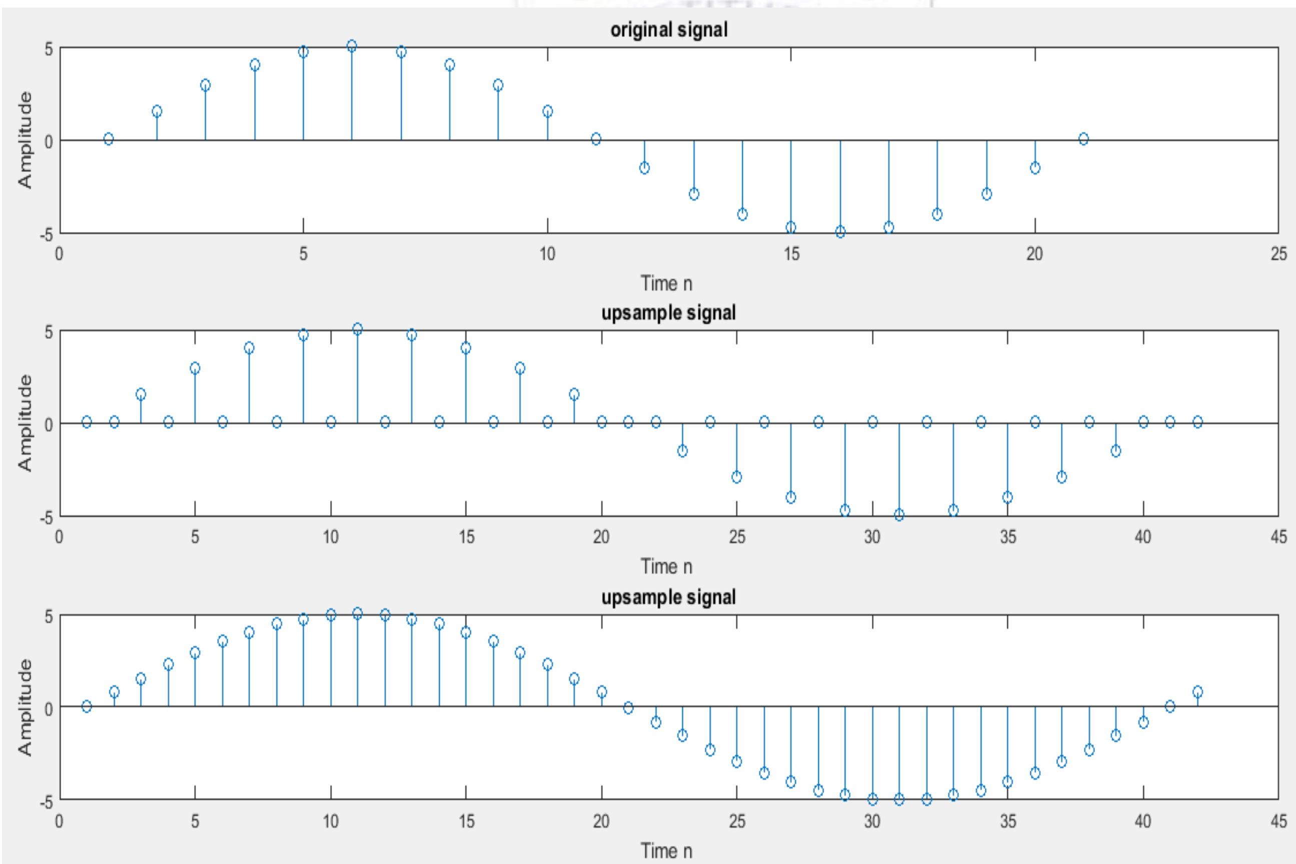
y = upsample(x,n,phase) specifies the number of samples by which to offset the upsampled sequence. phase must be an integer from 0 to n-1.

Program:-

```
clc;
close all;
clear all;
A=5;
F=50;
T=1/F;
I= input(' Enter the upsampling factor= ');

n=0:T/20:T;
y1=A*sin(2*pi*n*F);
subplot(3,1,1);
stem(y1);
title('original signal');
xlabel ('Time n ');
ylabel('Amplitude');
```

```
y2=upsample(y1,I);
subplot(3,1,2);
stem(y2);
title('upsample signal');
xlabel ('Time n ');
ylabel('Amplitude');

y3=interp(y1,I);
subplot(3,1,3);
stem(y3);
title('upsample signal');
xlabel ('Time n ');
ylabel('Amplitude');
```

OUTPUT



Upsampling Factor I = 2

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

Conclusion:

____
____
____
____

Questions:

1.

____
____
____
____
____
____

2.

____
____
____
____
____
____

3.

____
____
____
____
____
____

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: To generate program for cross correlation & Auto correlation.

EXPERIMENT NO. : **MIT(T)/ ETC** / __________/S&S/08

Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

Experiment No. 8

Aim: Write a MATLAB program to calculate cross correlation & Auto correlation.

Theory:

Auto correlation

- Correlation is mathematical technique which indicates whether 2 signals are related and in a precise quantitative way how much they are related. A measure of similarity between a pair of energy signals x[n] and y[n] is given by the cross correlation sequence $r_{xy}[l]$ defined by $r_{xy}[l] = \sum_{n=-\infty}^{\infty} x[n]y[n-l]; l = 0, \pm1, \pm2, \ldots$.
- The parameter 'l' called 'lag' indicates the time shift between the pair.
- Autocorrelation sequence of x[n] is given by $r_{xx}[l] = \sum_{n=-\infty}^{\infty} x[n]x[n-l]; l = 0, \pm1, \pm2, \ldots$
- Some of the properties of autocorrelation are enumerated below
  - The autocorrelation sequence is an even function i.e., $r_{xx}[l] = r_{xx}[-l]$
  - At zero lag, i.e., at l=0, the sample value of the autocorrelation sequence has its maximum value (equal to the total energy of the signal ε_x) i.e., $r_{xx}[l] \le r_{xx}[0] = \varepsilon_x = \sum_{n=-\infty}^{\infty} x^2[n]$.

    This is verified in Fig. 5.1, where the autocorrelation of the rectangular pulse (square) has a maximum value at l=0. All other samples are of lower value. Also the maximum value = 11 = energy of the pulse $[1^2+1^2+1^2..]$.
  - A time shift of a signal does not change its autocorrelation sequence. For example, let y[n]=x[n-k]; then $r_{yy}[l] = r_{xx}[l]$ i.e., the autocorrelation of x[n] and y[n] are the same regardless of the value of the time shift k. This can be verified with a sine and cosine sequences of same amplitude and frequency will have identical autocorrelation functions.
  - For power signals the autocorrelation sequence is given by

    $r_{xx}[l] = \lim_{k\to\infty} \frac{1}{2k+1} \sum_{n=-k}^{k} x[n]x[n-l]; l = 0, \pm1, \pm2, \ldots$ and for periodic signals with period N it is

    $r_{xx}[l] = \frac{1}{N} \sum_{n=0}^{N-1} x[n]x[n-l]; l = 0, \pm1, \pm2, \ldots$ and this **r_{xx}[l] is also periodic** with N. This is verified in Fig. 5.3 where we use the periodicity property of the autocorrelation sequence to determine the period of the periodic signal y[n] which is x[n] (=cos(0.25*pi*n)) corrupted by an additive uniformly distributed random noise of amplitude in the range [-0.5 0.5].

Algorithm:

1. Input the sequence as x.
2. fliplr function x and use the conv to get Auto correlated output rxx.
3. Plot the output sequence.

Prepared By	Approved By
Ms. S. R. Zambre	**Dr. G. S. Sable**

Cross correlation:

- Cross Correlation has been introduced in the last experiment. Comparing the equations for the linear convolution and cross correlation we find that $r_{xy}[l] = \sum_{n=-\infty}^{\infty} x[n]y[n-l] = \sum_{n=-\infty}^{\infty} x[n]y[-(l-n)] = x[l] * y[-l]$. i.e., convolving the reference signal with a folded version of sequence to be shifted (y[n]) results in cross correlation output. (Use 'fliplr' function for folding the sequence for correlation).
- The properties of cross correlation are 1) the cross correlation sequence sample values are upper bounded by the inequality $r_{xx}[l] \le \sqrt{r_{xx}[0] r_{yy}[0]} = \sqrt{\varepsilon_x \varepsilon_y}$

  2) The cross correlation of two sequences x[n] and y[n]=x[n-k] shows a peak at the value of k. Hence cross correlation is employed to compute the exact value of the delay k between the 2 signals. Used in radar and sonar applications, where the received signal reflected from the target is the delayed version of the transmitted signal (measure delay to determine the distance of the target).

  3) The ordering of the subscripts xy specifies that x[n] is the reference sequence that remains fixed in time, whereas the sequence y[n] is shifted w. r .t x[n]. If y[n] is the reference sequence then $r_{yx}[l] = r_{xy}[-l]$. Hence $r_{yx}[l]$ is obtained by time reversing the sequence $r_{xy}[l]$.

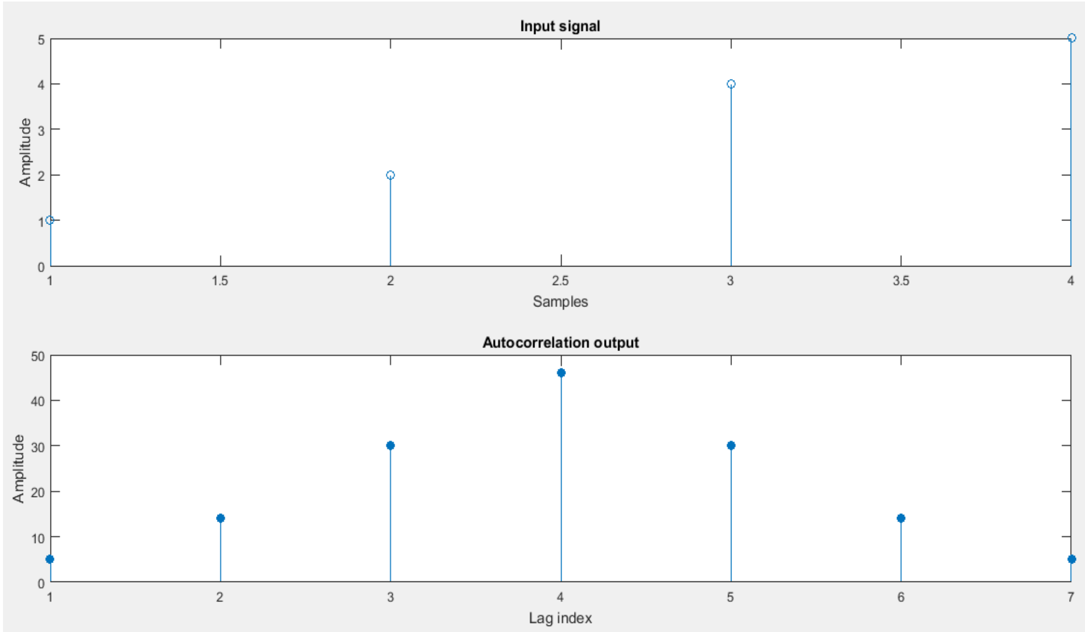
Algorithm:

3. Input the sequence as x and y.
4. fliplr function y and use the conv to get cross correlated output rxy.
3. Plot the output sequence

MATLAB Program for Auto correlation:

```
%=====================START=========================
clc;
close all;
clear all;

x=input ('Enter sequence x(n)=');
h= fliplr(x);
rxx= conv(x, h);
disp('rxx=');
disp(rxx);
subplot(2,1 ,1);
stem(x);
title('Input signal');
xlabel('Samples');
ylabel('Amplitude');

subplot(2,1,2);
stem(rxx,'filled');
title('Autocorrelation output');
xlabel('Lag index');
ylabel('Amplitude');
%=====================END=========================
```

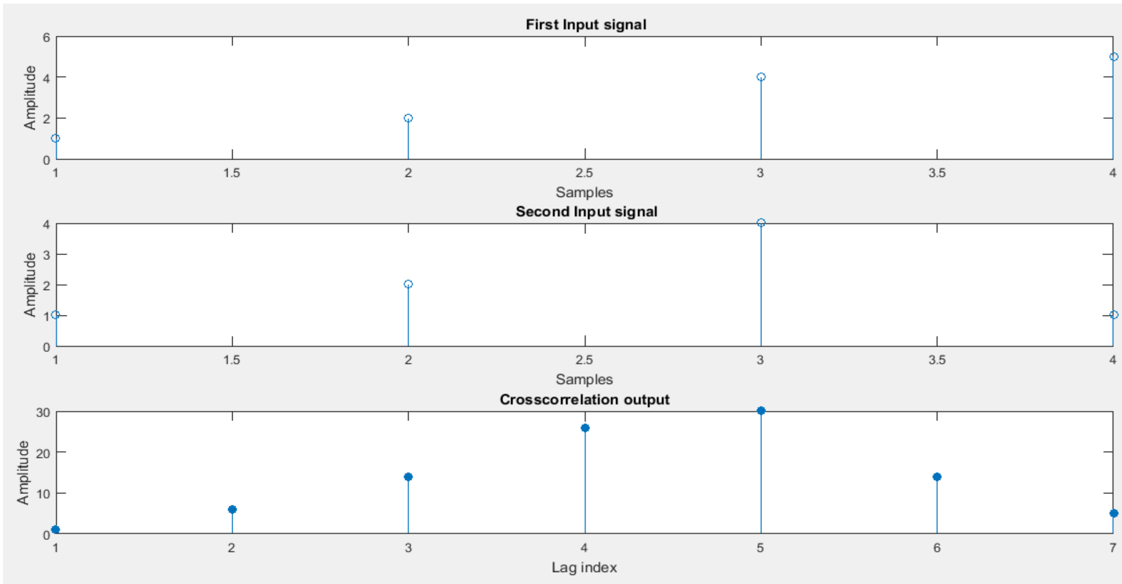
Result:
Enter sequence x(n)=[1 2 4 5]
rxx=
5 14 30 46 30 14 5



To obtain cross correlation of the given sequence
MATLAB Program

```
clc;
close all;
clear all;

x=input ('Enter sequence x(n)=');
y=input('Enter the second sequence y(n)=');
h= fliplr(y);
rxy= conv(x, h);
disp('rxy=');
disp(rxy);
subplot(3,1 ,1);
stem(x);
title(' First Input signal');
xlabel('Samples');
ylabel('Amplitude');

subplot(3,1 ,2);
stem(y);
title(' Second Input signal');
xlabel('Samples');
ylabel('Amplitude');
subplot(3,1,3);
stem(rxy,'filled');
title('Crosscorrelation output');
xlabel('Lag index');
ylabel('Amplitude');
```

Result:
Enter sequence x(n)=[1 2 4 5]
Enter the second sequence y(n)=[1 2 4 1]
rxy=
1 6 14 26 30 14 5



Conclusion:

Questions:

1.

2.

3.

Prepared By	Approved By
Ms. S. R. Zambre	**Dr. G. S. Sable**

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Experiment No. 9

Aim: - Write a MATLAB based program to compute power spectrum density (PSD).

Theory: -

The Power Spectrum Density describes the distribution of power into frequency components composing that signal. According to Fourier analysis any physical signal can be decomposed into a number of discrete frequencies or a spectrum of frequencies over a continuous range. The statistical average of a certain signal or sort of signal (including noise) as analyzed in terms of its frequency content is called its spectrum.

PERIODOGRAM:-PSD using periodogram

Syntax

[Pxx,w] = periodogram(x)
[Pxx,w] = periodogram(x,window)
[Pxx,w] = periodogram(x,window,nfft)
[Pxx,w] = periodogram(x,window,w)
[Pxx,f] = periodogram(x,window,nfft,fs)
[Pxx,f] = periodogram(x,window,f,fs)
[Pxx,f] = periodogram(x,window,nfft,fs,*'range'*)
[Pxx,w] = periodogram(x,window,nfft,*'range'*)
periodogram(...)

Description

[Pxx,w] = periodogram(x) returns the power spectral density (PSD) estimate Pxx of the sequence x using a periodogram. The power spectral density is calculated in units of power per radians per sample. The corresponding vector of frequencies w is computed in radians per sample, and has the same length as Pxx.

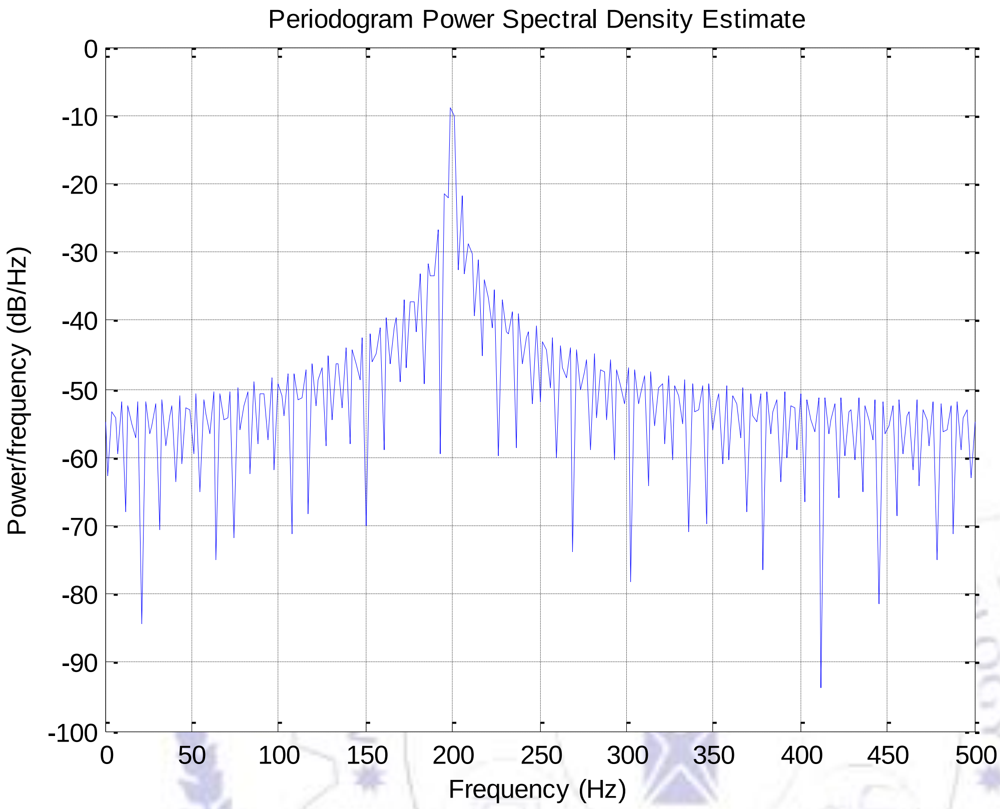
A real-valued input vector x produces a full power one-sided (in frequency) PSD (by default), while a complex-valued x produces a two-sided PSD.

In general, the length *N* of the FFT and the values of the input x determine the length of Pxx and the range of the corresponding normalized frequencies. For this syntax, the (default) length *N* of the FFT is the larger of 256 and the next power of 2 greater than the length of x. The following table indicates the length of Pxx and the range of the corresponding normalized frequencies for this syntax.

PROGRAM:-

```
Fs = 1000;
t = 0:1/Fs:.3;
x = cos(2*pi*t*200)
periodogram(x,[],'onesided',512,Fs)
```

OUTPUT:-



Conclusion:

Questions:

1.

2.

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

3.

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

Practical Experiment Instruction Sheet

EXPERIMENT TITLE: Program for continuous time signals.			
EXPERIMENT NO. : MIT(T)/ ETC / __________/S&S/10			
Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.		
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

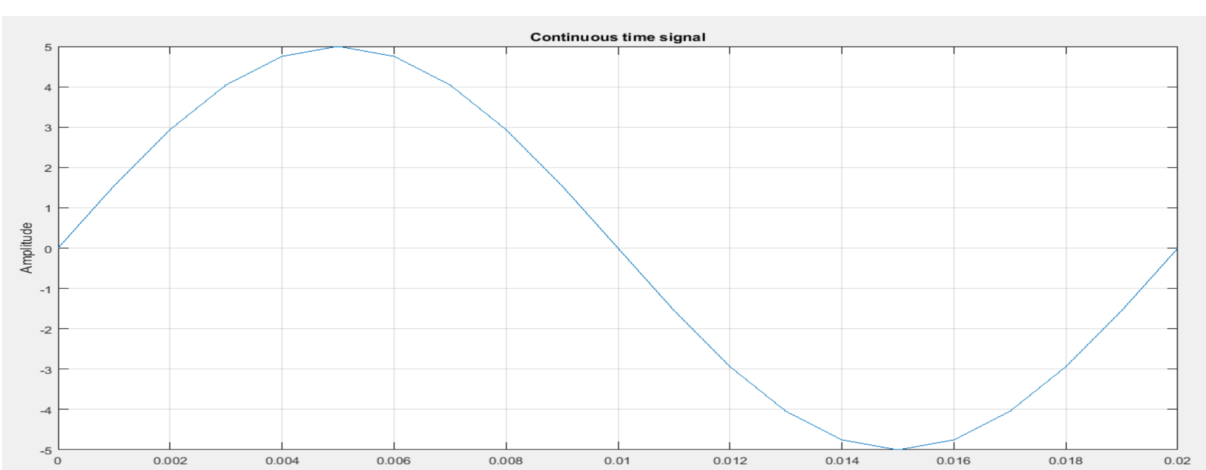
Experiment No. 10

Aim:- Write a MATLAB program to plot the following continuous time signals
X(t)=5*sin(2*pi*f*t)

```
%====================START=====================
clc;
clear all;
close all;
%% enter the amplitude & frequency of sinusoidal signal

A=input('Enter the Amplitude =');
F=input('Input frequency of the sinusoidal signal=');
T=1/F;
t=0:T/20:T;
x=A*sin(2*pi*F*t);

%% plot the signal
plot(t,x);
grid on
title('Continuous time signal');
xlabel('Time t');
ylabel('Amplitude');

%====================End=====================
```

Output:

Enter the Amplitude =5
Input frequency of the sinusoidal signal=50



Conclusion:

____

Questions:

1.

____

2.

____

3.

____

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable

	Maharashtra Institute of Technology, Aurangabad	LABORATORY MANUAL
	Practical Experiment Instruction Sheet	
EXPERIMENT TITLE: Program for discrete time signals.		
EXPERIMENT NO. : MIT(T)/ ETC / __________/S&S/11		
Class: SY E&TC	DEPARTMENT: Electronics & Telecommunication Engineering.	
LABORATORY :	Location:-	PART: II	Date:_ _/_ _/20_ _

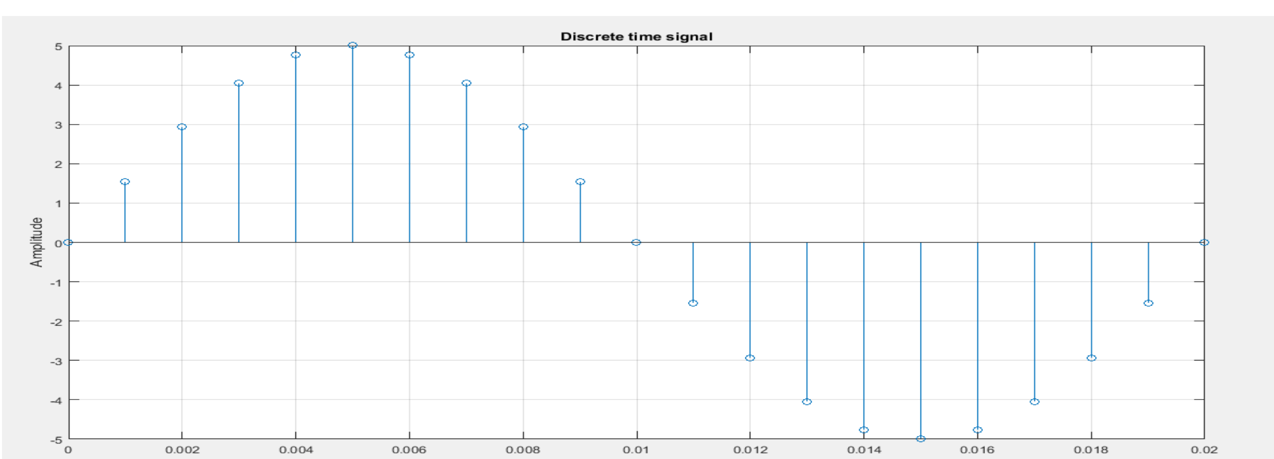
Experiment No. 11

Aim:- Write a MATLAB program to plot the following discrete time signals
X[n]=5*sin(2*pi*f*n)

```
%====================START=====================
clc;
clear all;
close all;
%% enter the amplitude & frequency of sinusoidal signal

A=input('Enter the Amplitude =');
F=input('Input frequency of the sinusoidal signal=');
T=1/F;
n=0:T/20:T;
x=A*sin(2*pi*F*n);

%% plot the signal
stem (n,x);
grid on
title('Discrete time signal');
xlabel('Time n');
ylabel('Amplitude');

%====================End=====================
```

Output:

Enter the Amplitude = 5
Input frequency of the sinusoidal signal=50



Prepared By	Approved By
Ms. S. R. Zambre	**Dr. G. S. Sable**

Conclusion:

____
____
____
____

Questions:

1.

____
____
____
____
____
____

2.

____
____
____
____
____
____

3.

____
____
____
____
____
____

Rubrics for Practical Assessment:

Cognitive (3)	Affective (3)	Psychomotor (3)	Total (9)

Sign of Course Teacher with Date

Prepared By	Approved By
Ms. S. R. Zambre	Dr. G. S. Sable