IQRA NATIONAL UNIVERSITY

Department of Electrical Engineering



Digital Signal Processing

Instructor Lab Manual

Instructor Name: Engr.Khalil Muhammad

Contents

[Lab 01 3](#_Toc413408137)

[MATLAB BASIC COMMANDS 3](#_Toc413408138)

[Lab 02 8](#_Toc413408139)

[Signal Representation 8](#_Toc413408140)

[Lab 03 13](#_Toc413408141)

[Convolution 13](#_Toc413408142)

[Lab 04 15](#_Toc413408143)

[Sampling 15](#_Toc413408144)

[Lab 05 16](#_Toc413408145)

[Z Transform 16](#_Toc413408146)

[Lab 06 18](#_Toc413408147)

[Frequency and Phase Response of LTI Systems 18](#_Toc413408148)

[Lab 07 20](#_Toc413408149)

Discrete [Fourier Transform 20](#_Toc413408150)

[Lab 08 22](#_Toc413408151)

[Implementation of FIR Filters 22](#_Toc413408152)

[Lab 09 25](#_Toc413408153)

[Implementation of IIR Filters 25](#_Toc413408154)

[Lab 10 27](#_Toc413408155)

[Demonstration of Down Sampling and Up Sampling 27](#_Toc413408156)

[Lab 11 28](#_Toc413408157)

[Signal Plotting Using Simulink 28](#_Toc413408158)

[Lab 12 30](#_Toc413408159)

[Fourier Transform and Filters Demonstration Using Simulink 30](#_Toc413408160)

[Lab 13 31](#_Toc413408161)

[Mini Project 31](#_Toc413408162)

# Lab 01

## MATLAB BASIC COMMANDS

This lab is to familiarize the students with MATLAB environment through some preliminary MATLAB functions.

**Basics:**

1. To add a comment the following symbol is used “%”.
2. Help is provided by typing “help” or if you know the topic then “help function\_name”.
3. If you don’t know the exact name of the topic or command you are looking for, type “lookfor keyword”.
4. Three dots “…” are used to continue a statement to next line (row).
5. If after a statement “;” is entered then MATLAB will not display the result of the statement.
6. MATLAB is case sensitive.

**Basic commands in MATLAB:**

1. Defining a scalar:

x = 1

**Output:**

1. Defining a column vector

v = [1;2;3]

**Output:**

1. X=Defining a row vector

w = [1 0 1]

**Output:**

1. Transpose of a vector

Y = w’

**Output:**

1. Defining range of a vector

X = 1:0.5:5

**Output:**

1. Empty vector

Y = []

**Output:**

1. Defining a matrix

M = [1 2 3;3 2 1]

**Output:**

1. Defining a zero matrix

M = zeros(2,3) % 1st parameter is row, 2nd parameter is column.

 **Output:**

1. Defining a ones matrix

M = ones(2,3)

**Output:**

1. Defining the identity matrix

I = eye(3)

**Output:**

1. Defining random matrix or vectors

R = rand(1,3)

**Output:**

1. Access a vector or matrix

I(2,2)

**Output:**

1. Access a row or column of a matrix

I(2,:) % 2nd row

 **Output:**

 I(:,2) %2nd column

 **Output:**

1. Size and length

size(I)

**Output:**

length(I)

**Output:**

**Operations on vectors and matrices in MATLAB**

1. x = [1 2 3 4 5]

y = 2\*x

**Output:**

1. z = y/2

**Output:**

1. w = [1 2 3 4 5]

v = x + y

**Output:**

1. Point by point multiply or divide

W = x.\*w

**Output:**

1. Round of data

Round([1.5 2; 2.2 3.1])

**Output:**

1. Summation of a vector

a = [1 4 6 3]

sum(a)

**Output:**

**Relational operators in MATLAB**

1. a = [1 1 3 4 1]

ind = (a == 1)

Output:

 ind = (a<1)

 Output:

 ind = (a>1)

 Output:

 ind = (a<=1)

 Output:

 ind = (a>=1)

 Output:

 ind = (a~=1)

 Output:

**Control Flow in MATLAB**

To control the flow of commands, MATLAB provides four commands a programmer can use.

* FOR loops
* WHILE loops
* IF-ELSE-END constructions
* SWITCH-CASE constructions
1. Suppose one need values of the sine function at eleven evenly spaced points.

for n=0:10;

x(n+1) = sin(pi\*n/10);

end

x

**Output:**

1. Suppose a number is divided by 2. The resulting quotient is divided by 2 again. This process must continue till the current quotient is less than or equal to 0.01.

q = pi;

while q>0.01;

q = q/2;

end

q

**Output:**

1. In the following example a random integer number x from the set {1,2,3…10} is generated. If x = 1 or x = 2, then the message probability = 20%. If x = 3 or 4 or 5, then the probability =30 % is displayed otherwise probability = 50% is generated.

x = ceil(10\*rand);

switch x

case {1,2}

disp('Probability = 20%');

case {3,4,5}

disp('Probability = 30%');

otherwise

disp('Probability = 50%');

end

# Lab 02

## Signal Representation

This lab is to familiarize the students with MATLAB commands for the generation of elementary continuous time and discrete time signals.

**Description**

A signal is described by a function of one or more independent variables. The value of the function (dependent variable) can be a real valued scalar quantity, a complex valued quantity or a vector. For example:

The signals may be classified into different categories such as

* Analog
* Digital
* Continuous time discrete
* Discrete time continuous

A discrete time analog signal is denoted as . To emphasize the discrete time nature of a signal, a signal is represented as instead of .

A discrete time signal can be represented in any of the following forms:

1. Functional representation
2. Tabular representation
3. Sequence representation
4. Graphical representation

The set of rules for implementing the system by a program that performs the corresponding mathematical operations is called Algorithm. Some of the continuous discrete time signals have been discussed in this Lab.

**Continuous Time Signals**

**Sinusoidal waveform:**

clc;

clear all;

close all;

t = 0 : 0.001 : 15;

f = input ('Enter the value of frequency = ');

a = input ('Enter the value of amplitude = ');

y =a\*sin(2\*pi\*f\*t);

plot (t,y,'r');

xlabel ('time');

ylabel ('a');

title ('sine wave');

grid on;

**Output:**

**Note: (Frequency = 2, Amplitude = 1)**

Q 1. Solve and plot the signal for , compare your result with MATLAB output.

**Square Waveform:**

clc;

clear all;

close all;

t = 0 : 0.001 : 5;

f = input ('Enter the value of frequency');

a = input ('Enter the value of amplitude');

subplot (2,1,1);

y =a\*square(2\*pi\*f\*t);

plot (t,y,'r');

xlabel ('time');

ylabel ('amplitude');

title ('Square wave')

grid on;

**Output:**

**Note: (Frequency = 1, Amplitude = 1)**

**Complex Waveform:**

clc;

clear all;

close all;

t = 0 : 0.001 : 5;

f = input ('Enter the value of frequency');

a = input ('Enter the value of amplitude');

y = [exp((-0.2+2\*i)\*t)];

plot (t,y,'r');

xlabel ('time');

ylabel ('amplitude');

title ('Complex wave')

grid on;

**Output:**

**Note: (Frequency = 2, Amplitude = 1)**

**Unit Step Waveform:**

clc;

clear all;

close all;

t = 0 : 0.001 : 5;

a = input ('Enter the value of amplitude');

plot (t,a,'r');

xlabel ('time');

ylabel ('amplitude');

title ( 'Complex Wave')

grid on;

**Output:**

**Note: (Amplitude = 1)**

**Exponential Singals:**

clc;

clear all;

close all;

a=0.5;

t=0:0.1:10;

disp('EXPONENTIAL DECAYING SIGNAL');

x=a.^t;

subplot(2,1,1);

plot(t,x);

xlabel('Time');

ylabel('Amplitude');

title('Exponential Decaying Signal Response');

grid on;

disp('EXPONENTIAL GROWING SIGNAL');

x=a.^-t;

subplot(2,1,2);

plot(t,x);

xlabel('Time');

ylabel('Amplitude');

title('Exponential Growing Signal Response');

grid on;

**Output:**

**Discrete Time Signals**

**Sinusoidal DT waveform:**

clc;

clear all;

close all;

N = input('Enter Number of Samples : ');

n = 0:0.1:N;

x = sin(n);

stem (n,x);

xlabel ('Time');

ylabel ('Amplitude');

title ('Discrete Time Sine Signal');

grid on;

**Output:**

**Note: (Number of Samples = 15)**

**Square DT Waveform:**

clc;

clear all;

close all;

N = input('Enter the number of Samples: ');

n = 0:0.1:N;

s = square(2\*n);

stem (n,s);

xlabel ('time');

ylabel ('amplitude');

title ('square wave')

grid on;

**Output:**

**Note: (Number of Samples = 10)**

**Complex DT Waveform:**

clc;

clear all;

close all;

n = 0 : 0.1 : 15;

y = [exp((-0.2+2\*i)\*n)];

stem (n,y,'r');

xlabel ('time');

ylabel ('amplitude');

title ('Discrete Time Complex wave')

grid on;

**Output:**

**Unit Step DT Waveform:**

clc;

clear all;

close all;

N = input(' Enter the Number of Samples : ');

n = -N:1:N;

x = [zeros(1,N) 1 ones(1,N)];

subplot (2,1,1);

stem (n,x,'r');

xlabel ('Time');

ylabel ('Amplitude');

title ('Unit Step Response');

grid on;

**Output:**

**Note: (Number of Samples = 10)**

**Exponential DT Signals:**

clc;

clear all;

close all;

N = input('Enter Number of Samples : ');

% EXPONENTIAL DECAYING SIGNAL

a = 0.5;

n = 0:.1:N;

x = a.^n;

subplot (2,1,1);

stem (n,x,'r');

xlabel ('Time');

ylabel ('Amplitude');

title ('Exponential Decaying Signal Response');

grid on;

%EXPONENTIAL GROWING SIGNAL

subplot (2,1,2);

x = a.^-n;

stem (n,x,'r');

xlabel ('Time');

ylabel ('Amplitude');

title ('Exponential Growing Signal Response');

grid on;

**Output:**

**Note: (Number of Samples = 10)**

**Exercise:**

1. Using above mentioned knowledge, write a MATLAB program to find the sum of sinusoidal signals and understand the concept of harmonics. , generate five signals from to , such that etc.
2. Plot the Unit Impulse Discrete Time Signal using MATLAB, for N = -10 to 10.

# Lab 03

## Convolution

This lab is to familiarize the students with MATLAB commands for convolution.

**Write a program to determine the discrete time convolution of x[n]=[1 2], h[n]=[2 3 4]:**

clc;

clear all;

close all;

n = 0:1:1;

n1 = 0:1:2;

n2 = 0:1:3;

x = input('Enter x[n]: ' );

h = input ('Enter h[n]: ' );

y = conv(x,h);

subplot(3,1,1);

stem(n, x, 'r' , 'c\*' );

title('x[n]' );

grid on;

subplot(3,1,2);

stem(n1, h, 'g' );

title('h[n]' );

grid on;

subplot(3,1,3);

stem(n2,y);

title('y[n]' );

grid on;

**Output:**

Q. **Confirm you answer by mathematically solving and plotting the out of y[n] = x[n]\*h[n], when:**

**x[n]=[1 2], h[n]=[2 3 4]**

**Write a program to determine the discrete time convolution of**

**Verify the answer obtained using table view method as well.**

**Write a program to determine the discrete time convolution of**

****

**Verify the answer obtained using table view method as well.**

**Write a program to determine the discrete time convolution of**

****

# Lab 04

## Sampling

This lab is to familiarize the students with the concept of sampling.

**Write a program to plot**

y=cos(2\*pi\*f1\*t)+ cos(2\*pi\*f2\*t);

where

f1=100

f2=200

**Output:**

**Write a program to determine the highest value between f1 and f2, in continuation of the above problem.**

**Output:**

**Write a program to verify the Nyquist criteria when sampling frequency is less than 2fm, with original signal, sampled discrete time signal and recovered signal in the plot.**

**Output:**

**Write a program to verify the Nyquist criteria when sampling frequency is equal to 2fm, with original signal, sampled discrete time signal and recovered signal in the plot.**

**Output:**

**Write a program to verify the Nyquist criteria when sampling frequency is greater than 2fm, with original signal, sampled discrete time signal and recovered signal in the plot.**

**Output:**

Post Lab Task:

1. **Explain the process of sampling (fs=2fm) and aliasing using frequency domain representation of signals.**
2. **Explain the drawbacks of under sampling and over sampling.**

# Lab 05

## Z Transform

This lab is to familiarize the students with MATLAB commands z-transform.

**Description**

As analog filters are designed using the Laplace transform, recursive digital filters are developed with a parallel technique called the z-transform. The overall strategy of these two transforms is the same: probe the impulse response with sinusoids and exponentials to find the system's poles and zeros. The Laplace transforms deals with differential equations, the s-domain, and the s-plane. Correspondingly, the z-transform deals with difference equations, the z-domain, and the z-plane. However, the two techniques are not a mirror image of each other; the s-plane is arranged in a rectangular coordinate system, while the z-plane uses a polar format. Recursive digital filters are often designed by starting with one of the classic analog filters, such, Chebyshev, or elliptic. A series of mathematical conversions are then used to obtain the desired digital filter. The Z transform of a discrete time system X[n] is defined as Power Series.

**Mathematically**

As Z Transform is the infinite Power Series; it exits only for the region for which the series converges (Region of convergence). Inverse Z Transform is the method of inverting the Z Transform of a signal to obtain the time domain representation

**Determine the Z Transform such that**

*(Note: Solve to confirm your answer)*

**Determine the poles and zero and plot the Poles and Zeros for the above two tasks.**

*(Note: Solve to confirm your answer)*

**Determine the Inverse Z-Transform of the system**

**Define vectors and create LTI systems in MATLAB for expressing the following transfer function**

* **Determine the poles and zero**
* **Plot the Poles and Zeros**

**Define vectors and create LTI systems in MATLAB for expressing the following transfer function**

* **Determine the poles and zero**
* **Plot the Poles and Zeros**

**Determine the output response of the system when**

**Define vectors and create LTI system in MATLAB for expressing the following transfer function**

k = 4;

z = [0.5 2];

p = [-.03 0.4 -0.6];

sys2 = zpk(z,p,k,1)

# Lab 06

## Frequency and Phase Response of LTI Systems

This lab is to familiarize the students with MATLAB commands for the magnitude and phase response of LTI systems.

**Determine the magnitude and phase response of LTI system**

%prog for finding the magnitude and phase response of LTI system by

%h(n) =(0.9)^n\*u(n)

w =[0:1:500]\*pi/500; %[0,pi] axis divided into 501 points

h =exp(j\*w)./(exp(j\*w)-0.9\*ones(1,501));

magh =abs(h); angh =angle(h);

subplot(2,1,1);plot(w/pi,magh);grid;

ylabel('|h|');

xlabel('(Frequency in pi units');

title('Magnitude Response');

subplot(2,1,2);plot(w/pi,angh/pi);grid;

ylabel('Phase in pi Radians');

xlabel('(Frequency in pi units');

title('Phase Response');

**Output**

**Determine the magnitude and phase response of the given LTI System.**

b = [1, 4]; % Numerator coefficients

a = [1, -5]; % Denominator coefficients

w = -2\*pi: pi/256: 2\*pi; % Frequency Response of the LTI system

[h] = freqz(b, a, w);

subplot(2, 1, 1),

plot(w, abs(h));

xlabel ('Frequency \omega'),

ylabel ('Magnitude');

grid on; % Phase Response of the LTI system

subplot(2, 1, 2),

plot(w, angle(h));

xlabel('Frequency \omega'),

ylabel('Phase - Radians');

grid on;

**Output**

**Determine the magnitude and phase response of the given LTI System.**

**Output**

**Determine the magnitude and phase response of the given LTI System.**

**Output**

# Lab 07

##  Discrete Fourier Transform

This lab is to familiarize the students with MATLAB commands for Fourier Transform.

**Description**

Fourier analysis is a family of mathematical techniques, all based on decomposing singals into sinusoids. The discrete Fourier transform (DFT) is the family member used with digitized signals. A signal can be either continuous or discrete, and it can be wither periodic or Non-periodic. The combination of these two features generates the four categories given below:

* **Period Continuous**

This includes the sine wave, square waves, and any waveform that repeats itself in a regular pattern from negative to positive infinity. This version of the Fourier transform is called Fourier series.

* **Non-periodic Continuous**

This includes decaying exponentials and Gaussian curves. These signals extend to both positive and negative infinity without repeating in a periodic pattern. The Fourier Transform for this type of signal is simply called the Fourier Transform.

* **Periodic Discrete**

These discrete signals repeat themselves in a periodic fashion from negative to positive infinity. This class of Fourier transform is sometimes called Discrete Fourier Series, but is most often called the Discrete Fourier Transform. The Fourier Transform for this type of signal is simply called the Discrete Time Fourier Transform.

* **Non-periodic Discrete**

These signals are only defined at the discrete points between positive and negative infinity, and do not repeat themselves in a periodic fashion.

**Determine the Fourier Transform of a simple sine wave:**

fs = 150;

t = 0:1/fs:1;

f = 5;

x = sin(2\*pi\*f\*t);

nfft = 1024;

X = fft(x,nfft);

X = X(1:nfft/2);

mx = abs(X);

%f\_bins = 0:nfft/2-1;

f = (0:nfft/2-1)\*fs/nfft;

subplot(2,1,1)

plot(t,x,'linewidth',2)

grid on

subplot(2,1,2)

plot(f,mx,'linewidth',2)

grid on

**Output**

**Determine the Fourier Transform of the following signal:**

x = cos(2\*pi\*f1\*t)+2\*cos(2\*pi\*f2\*t);

let f1 = 10

 f2 = 20

**Output**

**Determine the Fourier Transform of a square wave with F=5:**

**Output**

**Determine the Fourier Transform of a given pulse:**

****

**Output**

**Determine the DTFT of [0 1 2 3 4 5 6 7]:**

clc;

clear all;

close all;

x =input('Enter the sequence '); %x =[0 1 2 3 4 5 6 7]

n = input('Enter the length of Fourier Transform ') %n =8 has to be same as

%the length of sequence

x =fft(x,n);

stem(x);

ylabel('imaginary axis');

xlabel('real axis');

title('Exponential sequence');

disp('DFT is');

**Output**

# Lab 08

## Implementation of FIR Filters

This lab is to familiarize the students with MATLAB commands for FIR Filter Design.

**Description**

Digital filters refers to the hard ware and software implementation of the mathematical algorithm which accepts a digital signal as input and produces another digital signal as output whose wave shape, amplitude and phase response has been modified in a specified manner. Digital filter play very important role in DSP. Compare with analog filters they are preferred in number of application due to following advantages:

* Truly linear phase response
* Better frequency response Filtered
* Unfiltered data remains saved for further use.

There are two types of digital filters.

1. FIR (finite impulse response) filter
2. IIR (infinite impulse response) filter

**FIR1 Command**

FIR filters design using the window method. designs an order low pass FIR digital filter and returns the filter coefficients in length vector B. The cutoff frequency must be between , with 1.0 corresponding to half the sample rate. The filter B is real and has linear phase. The normalized gain of the filter at is .

 designs an *N'th* order highpass filter. You can also use *B = FIR1(N,Wn,'low')* to design a lowpass filter. If *Wn* is a two-element vector, *Wn = [W1 W2]*, *FIR1* returns an order *N* bandpass filter with passband *W1 < W < W2*. *B = FIR1(N,Wn,'stop')* is a bandstop filter if *Wn = [W1 W2]*. You can also specify If Wn is a multi-element vector, *Wn = [W1 W2 W3 W4 W5 ... WN]*, *FIR1* returns an order *N* multiband filter with bands *0 < W < W1, W1 < W < W2, ..., WN < W < 1*.

*B = FIR1(N,Wn,'DC-1')* makes the first band a pass band.

*B = FIR1(N,Wn,'DC-0')* makes the first band a stop band.

By default, FIR1 uses a Hamming window. Other available windows, including Boxcar, Hann, Bartlett, Blackman, Kaiser and Chebwin can be specified with an optional trailing argument. For example, *B = FIR1(N,Wn,kaiser(N+1,4))* uses a Kaiser window with beta=4. *B = FIR1(N,Wn,'high',chebwin(N+1,R))* uses a Chebyshev window. For filters with a gain other than zero at *Fs/2*, e.g., high pass and bands top filters, N must be even. Otherwise, N will be incremented by one. In this case the window length should be specified as *N+2*.

**Differentiate between Low Pass, High Pass, Band Pass and Band Stop Filters via proper diagrams.**

**Design a Low Pass FIR Filter:**

N = input('Filter Order ');

wc = input('Cutoff Frequency ');

h = fir1(N,wc);

fvtool(h)

**Output (N = 5, 25, 50) (wc = 0.5)**

**Q. What happens when you increase order of the filter and why?**

**Implement the FIR low Pass filter with N = 50 using Rectangular Window and Hamming Window commands**

**Output**

**Design a High Pass FIR Filter:**

**Output (N = 5, 25, 50) (wc = 0.5)**

**Design a Band Pass FIR Filter:**

**Output (N = 50) (wc = 0.3 and 0.7)**

**Output**

**Design a Band Stop FIR Filter:**

**Output (N = 50) (wc = 0.3 and 0.7)**

**Output**

**Design a Low Pass FIR Filter Using Rectangular Window Method:**

N = 50;

wc = 0.5;

a = fir1(N,wc);

b = fir1(N,wc,rectwin(51));

fvtool(a)

fvtool(b)

**Output**

**Demonstrate the filtering operation using a 10-point averaging low pass filter.**

fs = 100;

t = 0:1/fs:1;

x = sin(2\*pi\*t\*3)+.25\*sin(2\*pi\*t\*40);

b = ones(1,10)/10; % 10 point averaging filter

y = filter(b,1,x);

plot(t,x,'b',t,y,'r')

**Output**

**Decrease the order of the filter and compare your result. What is the effect when we decrease the order of the filter?**

# Lab 09

## Implementation of IIR Filters

This lab is to familiarize the students with MATLAB commands for IIR Filter Design.

**Description**

Matlab contains various routines for design and analyzing digital filter IIR. Most of these are part of the signal processing toolbox. A selection of these filters is listed below.

* Buttord ( for calculating the order of filter)
* Butter ( creates an IIR filter)
* Ellipord ( for calculating the order of filter)
* Ellip (creates an IIR filter)
* Cheb1ord (for calculating the order of filter)
* Cheyb1 (creates an IIR filter)

**Buttord**

*[N, Wn] = BUTTORD(Wp, Ws, Rp, Rs)* returns the order *N* of the lowest order digital Butterworth filter that loses no more than *Rp dB* in the pass band and has at least *Rs dB* of attenuation in the stop band. *Wp* and *Ws* are the pass band and stop band edge frequencies, normalized from 0 to 1 (where 1 corresponds to pi radians/sample). For example

* Low pass: Wp = .1, Ws = .2
* High pass: Wp = .2, Ws = .1
* Band pass: Wp = [.2 .7], Ws = [.1 .8]
* Band stop: Wp = [.1 .8], Ws = [.2 .7]

BUTTORD also returns Wn, the Butterworth natural frequency (or, the "3 dB frequency") to use with BUTTER to achieve the specifications.

*[N, Wn] = BUTTORD(Wp, Ws, Rp, Rs, 's')* does the computation for an analog filter, in which case Wp and Ws are in radians/second. When *Rp* is chosen as 3 dB, the *Wn* in BUTTER is equal to *Wp* in BUTTORD.

**Design a Low Pass Butterworth IIR Filter which pass frequencies below 1200 Hz with pass band ripples = 1 dB and minimum stop band attenuation of 50 dB at 1500 Hz:**

fs=8000;

[n,w]=buttord(1200/4000,1500/4000,1,50); % finding the order of the filter

[b,a]=butter(n,w); % finding zeros and poles for filter

figure(1)

[h,q] = freqz(b,a,512,8000);

plot(q,abs(h)); % Normalized Magnitude plot

grid on

**Output**

**Design a High Pass Butterworth IIR Filter which passes frequencies above 1200 Hz:**

**Output**

**Design a Band Pass filter to pass all frequencies between 1200 Hz and 2800 Hz with pass band ripples = 1 dB and minimum stop band attenuation of 50 dB. Sampling frequency is 8000 Hz.**

**Output**

**Design a Band Stop IIR Filter:**

**Output**

**Design an IIR Filter to suppress frequencies of 50 Hz from given signal:**

x = sin(2\*pi\*5\*t) + sin(2\*pi\*10\*t) + sin(2\*pi\*30\*t) + sin(2\*pi\*50\*t);

# Lab 10

## Demonstration of Down Sampling and Up Sampling

This lab is to familiarize the students with MATLAB commands for Down sampling and Up sampling.

**Determine the down sampling operation**

x = 0:1:20;

y = downsample(x',n)'

**Output (n = 2, 4, 6)**

**Plot the outputs**

**Determine the up sampling operation**

x = 0:1:20;

y = upsample (x',n)'

**Output (n = 2, 4)**

**Plot the outputs**

**Plot a signal** x = cos(2\*pi\*100\*t) **and apply the down sampling operation by reducing the number of intervals in t.**

# Lab 11

## Signal Plotting Using Simulink

This lab is to familiarize the students with Simulink Basics.

**Plot a sine wave with amplitude 1 and frequency 1, 2 and 4.**

****

****

**Outputs**

**Plot a cosine wave with amplitude 1 and frequency 2.**

**Output**

**Plot a Unit Step Function with time starting from 0 to 10.**

**Output**

**Plot a simple Ramp Function with time starting from 0 to 10.**

**Output**

**Plot the sum of first 4 sinusoids.**

**Output**

# Lab 12

## Fourier Transform and Filters Demonstration Using Simulink

This lab is to familiarize the students with Simulink Basics for frequency representation and filter design.

**Plot a simple sine wave with 5 Hz frequency and determine its frequency component.**

**Plot a composite signal and determine its frequency components.**

**Design and Low pass filter and determine its output.**

**Design and High pass filter and determine its output.**

# Lab 13

## Introduction to TMS320C5505 eZDSP Stick

This lab is designed to familiarize the students with the installation, interconnection and use of DSP kits.

1. Use the file in accompanied USB to install and configure eZDSP stick.
2. The C5505 eZDSP USB Stick must be connected in CCS before proceeding. If the board is not connected, please refer to section 4.0 of this guide.
3. Click “File->Import”.
4. When the new window appears, expand “CCS” and select “Existing CCS/CCE Eclipse Project”. Click “Next”.
5. Select “Select Root Directory” and click the “Browse” button. Browse to directory path “<Install\_Dir>\ccsv4\emulation\boards\usbstk5505\_v2\tests\led”. The default <Install\_Dir> is “C:\programfiles\Texas Instruments”
6. Highlight the “LED” folder and click “OK”. Then click “finish”.
7. Click “View->C/C++Projects”. A new tab will appear in the CCS v4 window and there should be a LED project visible.
8. Click “Project->Build Active Project”.
9. When the build is complete, a message will print in the Console Window.
10. Look in the “C\C++Projects” tab again and expand the “binaries” folder. There now should be a “led.out” file listed.
11. Right click on “led.out” and click “Load Program”.
12. Click “Target->Run”. The C5505 eZDSP USB Stick “XF” LED should now blink.