**LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING**

**L.B REDDY NAGAR, MYLAVARAM-521230**

# Signals and Systems LAB

**B. Tech IV SEM (ECE)**

**COURSE CODE: 23EC54**

**DEPARTMENT OF ECE**

**A.Y:2024-2025**

**Prepared by**

**Mr.M.K.Linga Murthy**

**Verified by**

**Dr.G.L.N.Murthy** **Dr.P.James Vijay**

## LIST OF EXPERIMENTS

# EXPERIMENT-1

## Generation of Basic Signals (Analog and Discrete)

**AIM:** To generate and plot the following continuous time signals using MATLAB Software:

a) Unit step
b) Unit Ramp
c) Unit Impulse
d) Sinusoidal
e) Signum

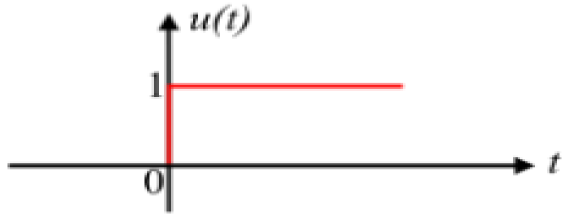
**Software Used:** MATLAB R2016a

**Theory:**

A **signal** is defined as a single-valued function of one or more independent variables which contain some information. Examples of signals are electric current and voltage, human speech, etc.

**Unit Step Signal:** A signal x(t) is said to be step signal if and only if it exists for t > 0 and zero for t < 0. The step signal is an important signal used for analysis of many systems. If a step signal has unity magnitude, then it is known as unit step signal or unit step function. It is denoted by u(t).

$$u(t) = 1 \text{ for } t \geq 0 \text{ and } u(t) = 0 \text{ ; for } t < 0$$

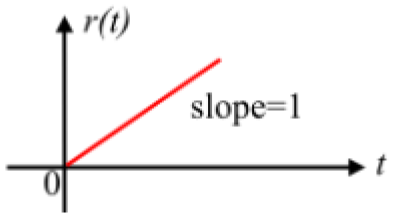


**Unit ramp Signal:** The continuous-time unit ramp signal is that function which starts at $t = 0$ and increases linearly with time. It is denoted by r(t). The mathematical expression of a continuous-time ramp signal is

$$r(t) = 1 \quad \text{for} \quad t \geq 0$$

$$r(t) = 0 \quad \text{for} \quad t < 0$$

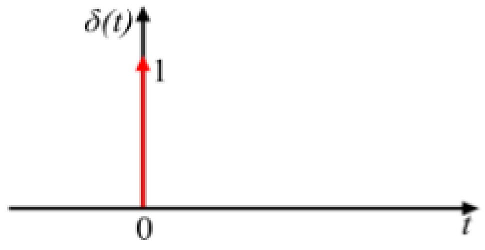
Ramp signal can also be expressed as r(t)= tu(t).



**Unit Impulse Signal:** The continuous-time unit impulse signal is denoted by δ(t).
The ideal impulse signal is zero everywhere but infinitely high at the instant zero. But the area of the ideal impulse signal is a finite value only.

$$\int_{-\infty}^{\infty} \boldsymbol{\delta}(\mathbf{x})\mathbf{dx} = \mathbf{1}$$

The unit impulse signal finds extensive use in the analysis of signals and systems.



**Signum function:**

Signum function is denoted as sgn(t). It is defined as

$$\mathbf{sgn(t)} \quad = -1 \text{ if } t < 0$$

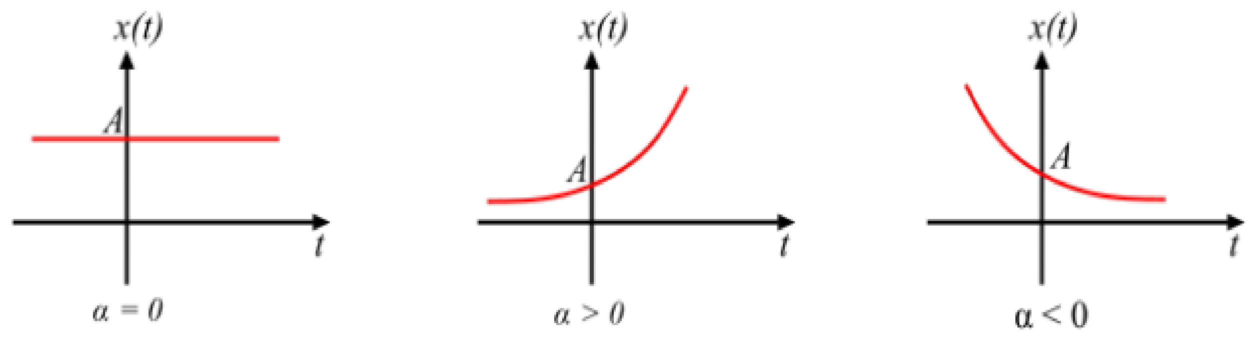
$$= +1 \text{ if } t > 0$$

$$= 0 \;\; \text{If } t = 0$$

$$\mathbf{sgn(t) = 2u(t) - 1}$$

**Exponential Signal:** A real exponential signal which is defined for every instant of time is called **continuous time real exponential signal**. A continuous time real exponential signal is defined as

$$\mathbf{x(t) = Ae^{\alpha t}}$$



**Sinusoidal Signal:** The mathematical expression for the sinusoidal signal is given by x(t)=Asin(ωt+Φ) Where A is amplitude and Φ is phase difference.

## Procedure:

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window

## Program:

```
clc
close all
clear all
% Define time vector
t = -5:0.001:5; % Time range from -5 to 5 seconds with step size of 0.001
 % Unit step signal (u(t))
u = double(t >= 0);
 % Impulse signal (delta(t))
```

```
impulse = (t == 0); % Impulse is 1 at t = 0, otherwise 0
 % Ramp signal (r(t))
r = t .* (t >= 0);
 % Exponential signal (e^(-t))
exp_signal = exp(-t) .* (t >= 0);
 signumoft =2.*(t>=0)-1;
sin_t=sin((pi/2)*t);
% Plot all signals together
figure;
subplot(6, 1, 1);
plot(t, u, 'LineWidth', 2);
title('Unit Step Signal: u(t)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
 subplot(6, 1, 2);
plot(t, impulse, 'LineWidth', 2);
title('Impulse Signal: delta(t)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
 subplot(6, 1, 3);
plot(t, r, 'LineWidth', 2);
title('Ramp Signal: r(t)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
 subplot(6, 1, 4);
plot(t, exp_signal, 'LineWidth', 2);
title('Exponential Signal: e^{-t}');
xlabel('Time (s)');
```

```
ylabel('Amplitude');
grid on;
subplot(6, 1, 5);
plot(t, signumoft, 'LineWidth', 2);
title('Signum function: sgn(t)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
 subplot(6, 1, 6);
plot(t, sin_t, 'LineWidth', 2);
title('Sinusoidal Signal: Sin(t)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
 % Define discrete time index (n)
n = -5:5; % Time index from -10 to 10
 % Unit step signal (u[n])
u_discrete = double(n >= 0);
 % Impulse signal (delta[n])
impulse_discrete = (n == 0);
 % Ramp signal (r[n])
r_discrete = n .* (n >= 0);
 % Exponential signal (e^(-n))
exp_discrete = exp(-n) .* (n >= 0);
sin_discrete=sin((pi/2)*n);
 % Plot all discrete signals together
figure;
subplot(5, 1, 1);
stem(n, u_discrete, 'filled', 'LineWidth', 2);
title('Discrete-Time Unit Step Signal: u[n]');
xlabel('n (Discrete Time Index)');
```

```
ylabel('Amplitude');
grid on;
 subplot(5, 1, 2);
stem(n, impulse_discrete, 'filled', 'LineWidth', 2);
title('Discrete-Time Impulse Signal: delta[n]');
xlabel('n (Discrete Time Index)');
ylabel('Amplitude');
grid on;
 subplot(5, 1, 3);
stem(n, r_discrete, 'filled', 'LineWidth', 2);
title('Discrete-Time Ramp Signal: r[n]');
xlabel('n (Discrete Time Index)');
ylabel('Amplitude');
grid on;
 subplot(5, 1, 4);
stem(n, exp_discrete, 'filled', 'LineWidth', 2);
title('Discrete-Time Exponential Signal: e^{-n}');
xlabel('n (Discrete Time Index)');
ylabel('Amplitude');
grid on;
 subplot(5, 1, 5);
stem(n, sin_discrete, 'filled', 'LineWidth', 2);
title('Discrete-sinusoidal Signal: sin(n)');
xlabel('n (Discrete Time Index)');
ylabel('Amplitude');
grid on;
```

## Precautions:

1. Check out source file is with '.m' extension or not.

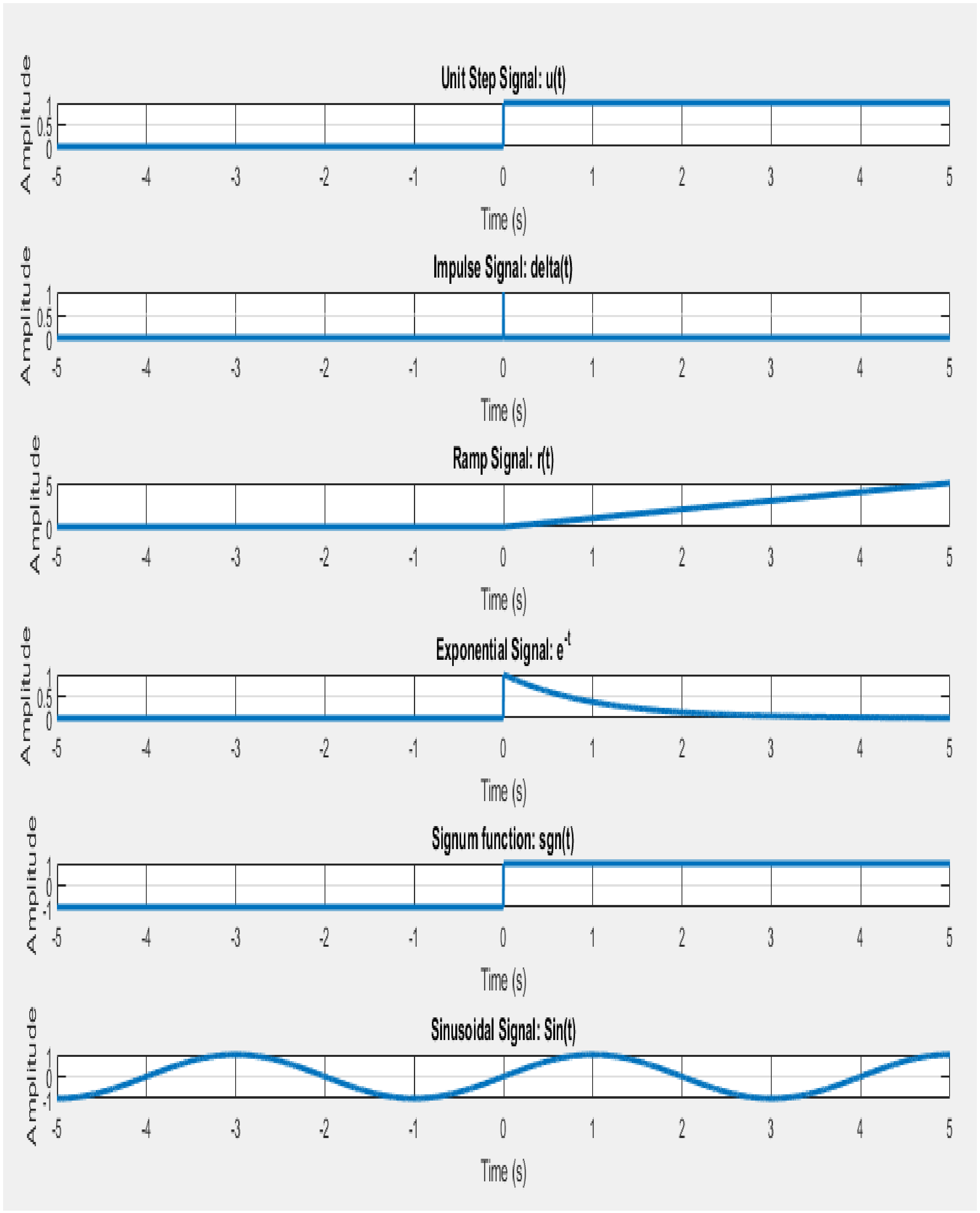
2. The file name should begin with character and should not contain any punctuation marks.

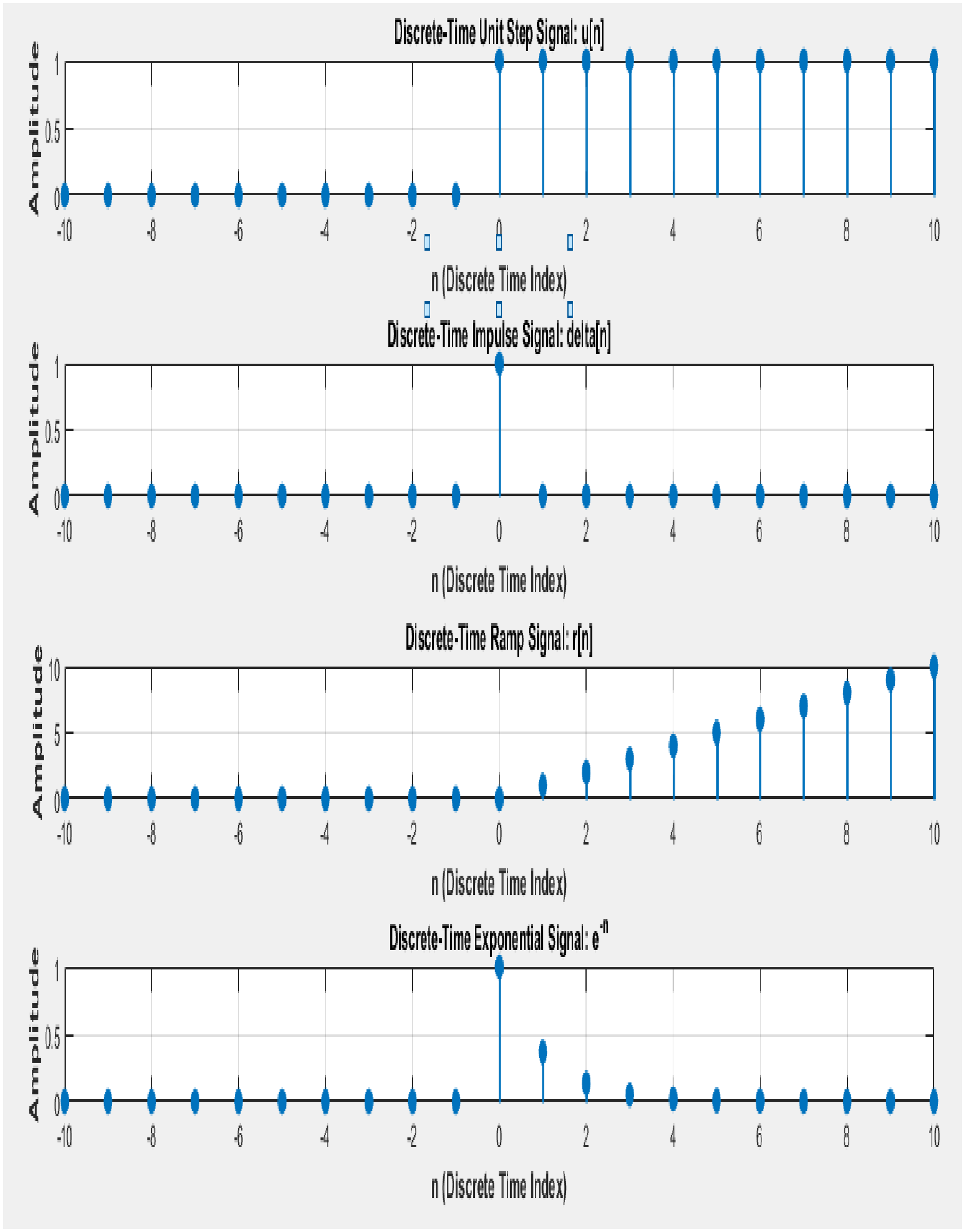
3. File name should not be any in built in function name or any keyword

4. Save the .m files preferably in work folder of MATLAB.

5. Don't delete any file or folder without informing the system administrator or lab in-charge.

**Output:**





Discrete-Time Unit Step Signal: u[n]
Discrete-Time Impulse Signal: delta[n]
Discrete-Time Ramp Signal: r[n]
Discrete-Time Exponential Signal: $e^{-n}$
Amplitude
n (Discrete Time Index)

**Result:** Fundamental Continuous time and Discrete time signals were generated.

## Viva Questions:

1. Define Signal?
2. Define continuous and discrete Signals?
3. State the relation between step, ramp and Delta Functions?
4. Differentiate saw tooth and triangular signals?
5. Define Periodic and aperiodic Signal?

# EXPERIMENT-2
## Operations on signals

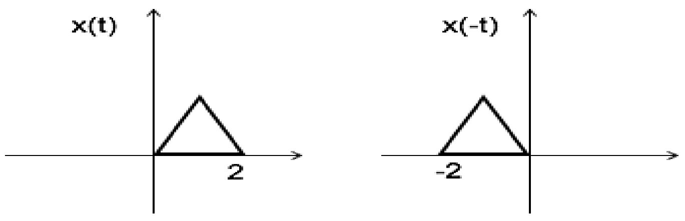
**AIM:** To perform the following operations on Signals

a) Time Shifting
b) Time reversal
c) Addition
d) Subtraction

**Software Used:** MATLAB R2016a

## Theory:

**Time Reversal:** Also called reflection of the signal. It is a special case of time scaling. x(-t) is time reversal version of x(t)



**Time Shifting**: x(t - T ) is the time shifting version of x(t) ,where Tis a constant. If T is positive, the signal said to be delayed i.e., right shift operation .If T is negative, the signal is said to be advanced i.e., left shift operation

**Addition:** If two continuous-time signals are added, $x_1(t)$ and $x_2(t)$, the resultant signal will have an amplitude equal to the sum of their amplitudes.

**Subtraction:** The subtraction operation of two signals is similar to that of addition, as the amplitude of the resultant signal is the value obtained from the subtraction of the amplitudes of the two individual signals in their respective intervals.

**Procedure:**

1. Open the MATLAB software by double clicking the icon on desktop.

2. Open the new M-file by using file menu.

3. Write the program in new file.

4. Click on save and run the icon.

5. Perform error check which displayed on command window.

6. Plot the waveforms displayed on figure window.

7. Note down the values, which are displayed on the command window

## Program:

```
% MATLAB Code for Time Shifting and Time Reversal of Unit Step Signal,
% addition and subtraction of two given unit step signals
clc; clear; close all;

t = -5:0.001:5; % Time range from -5 to 5 seconds with step size of 0.001

% Unit step signal (u(t))
u = double(t >= 0);

% Plot the original unit step signal
figure;
subplot(3, 1, 1);
plot(t, u, 'b', 'LineWidth', 1.5);
title('Original Unit Step Signal');
xlabel('Time (t)');
ylabel('Amplitude');
grid on;

% Time shifting
shift_amount = -4; % Shift by 3 units
t_shifted = t - shift_amount; % Negative shift moves the signal right
```

```
u_shifted = t_shifted >= 0; % Generate shifted unit step signal

subplot(3, 1, 2);
plot(t, u_shifted, 'r', 'LineWidth', 1.5);
title(['Time-Shifted Signal (Shift by ', num2str(shift_amount), ' units)']);
xlabel('Time (t)');
ylabel('Amplitude');
grid on;

% Time reversal
t_reversed = -t; % Reverse the time axis
u_reversed = t_reversed >= 0; % Generate time-reversed unit step signal

subplot(3, 1, 3);
plot(t, u_reversed);
title('Time-Reversed Signal');
xlabel('Time (t)');
ylabel('Amplitude');
grid on;

t1 =t;% Time range from -10 to 10 seconds with step size of 0.001
t2=t1;
% Unit step signal (u(t))
u1 = double(t1 >= 0);
u2=u1;

% Perform convolution
add_signals = u1+u2; % addition of u1 and u2

sub_signals = u1-u2; % subtraction of u1 and u2
% Plot the signals
```

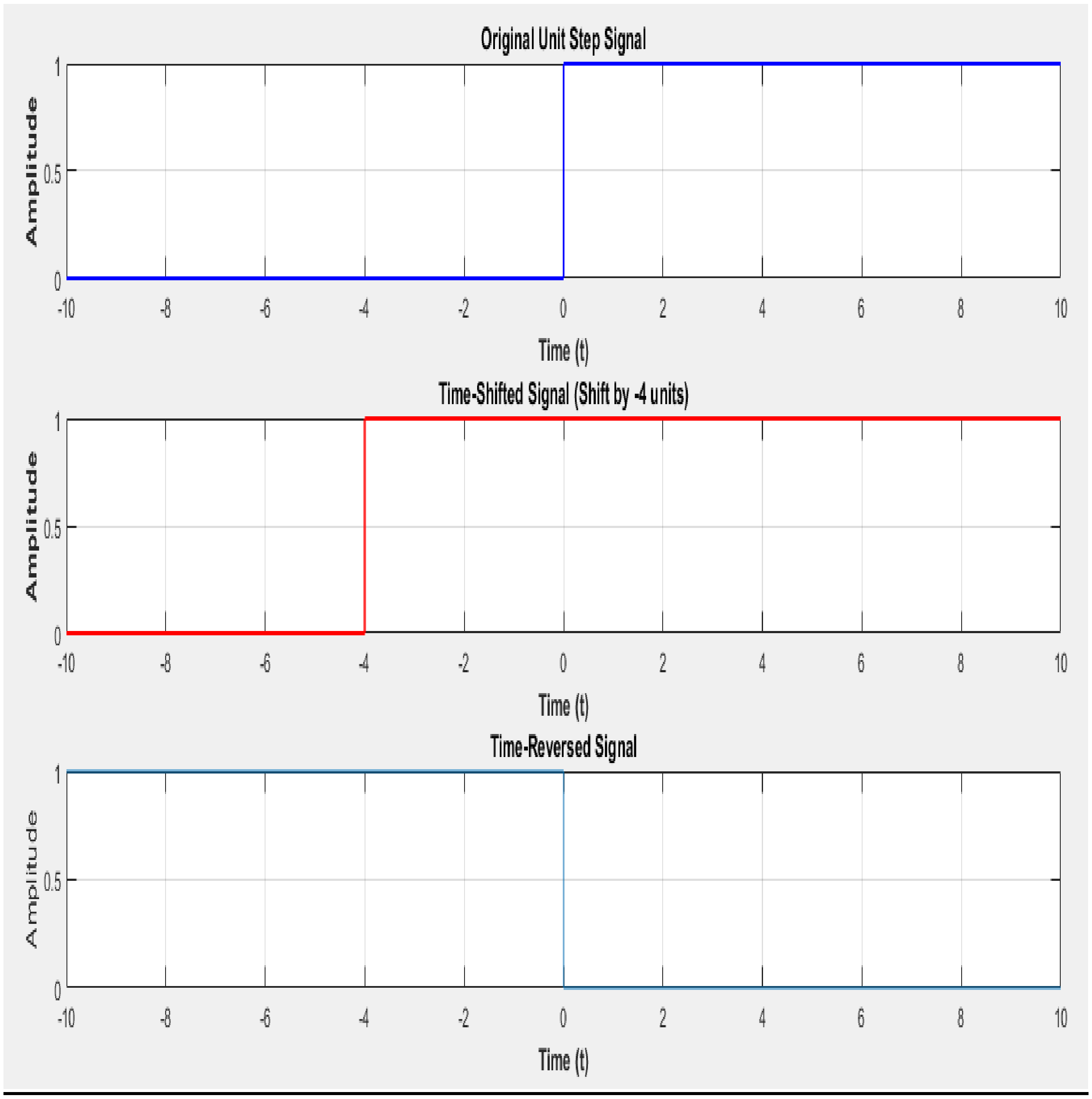
```
figure;

% Plot the first unit step signal
subplot(4, 1, 1);
plot(t1, u1, 'b', 'LineWidth', 1.5);
title('Unit Step Signal 1');
xlabel('Time (t)');
ylabel('Amplitude');
grid on;

% Plot the second unit step signal
subplot(4, 1, 2);
plot(t2, u2, 'r', 'LineWidth', 1.5);
title('Unit Step Signal 2');
xlabel('Time (t)');
ylabel('Amplitude');
grid on;

% Plot the added signal
subplot(4, 1, 3);
plot(t1, add_signals, 'g', 'LineWidth', 1.5);
title('Addition of Unit Step Signal 1 and Unit Step Signal 2');
xlabel('Time (t)');
ylabel('Amplitude');
grid on;
% Plot the subtracted  signal
subplot(4, 1, 4);
plot(t1, sub_signals, 'g', 'LineWidth', 1.5);
title('Subtratcion of Unit Step Signal 1 and Unit Step Signal 2');
xlabel('Time (t)');
ylabel('Amplitude');
```

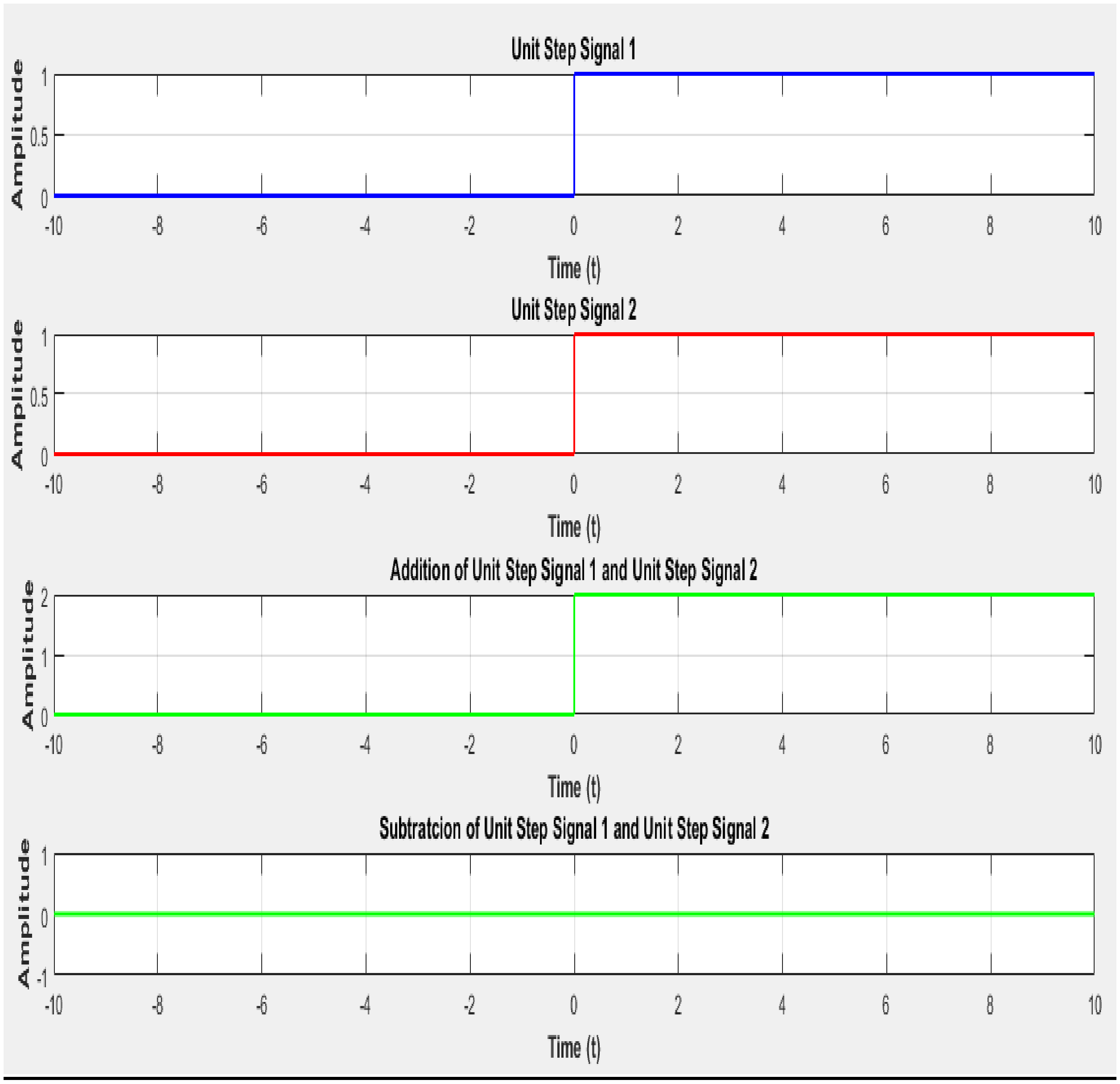
```
grid on;
```

## Precautions:

1. Check out source file is with '.m' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Save the .m files preferably in work folder of MATLAB.
5. Don't delete any file or folder without informing the system administrator or lab in-charge.

**Output:**


Original Unit Step Signal
Amplitude
Time (t)
Time-Shifted Signal (Shift by -4 units)
Amplitude
Time (t)
Time-Reversed Signal
Amplitude
Time (t)
-10
-8
-6
-4
-2
0
2
4
6
8
10
0
0.5
1



**Result:** Time shifting, time reversal as well as addition and subtraction operations were performed on signals

**Viva Questions:**

1. Sketch the signal 5+ 5 Sinwt
2. Draw the signal x(t)= 2u(t)-1
3. Sketch the signal x(t)=r(t)-2r(t-1)+r(t-2)
4. Plot the signal x(t)=u(t+2)-u(t-3).
5. Draw the signal x(t)=3r(t-1)

# EXPERIMENT-3
## Estimation of energy and power of signal

**AIM:** To estimate the energy and power of given signal

**Software Used:** MATLAB R2016a

**Theory:**

In signal processing, understanding the **energy** and **power** of a signal is essential for analyzing and characterizing the behavior of signals in different contexts, such as communications, control systems, and more. These two concepts help quantify the strength or magnitude of a signal over time.

**1. Energy of a Signal**

The **energy** of a signal provides a measure of how much "work" the signal does over time. It quantifies the total amount of energy contained in the signal. The energy of a signal is defined as the square of its amplitude integrated over time (or summed over discrete samples in the case of discrete signals).

**Continuous-Time Signal Energy:**

For a continuous-time signal x(t)x(t)x(t), the energy is defined as:

$$E = \int_{-\infty}^{\infty} |x(t)|^2 \, dt$$

**Discrete-Time Signal Energy:**

For a discrete-time signal x[n], the energy is defined as:

$$E = \sum_{n=-\infty}^{\infty} |x[n]|^2$$

**2.Power of a Signal**

The power of a signal is a measure of how much energy the signal delivers per unit of time. Unlike energy, which is a measure of the total work over a duration, power focuses on how much energy is being transmitted or consumed over time.

For a continuous-time signal x(t), the average power  is defined as:

$$P = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} |x(t)|^2 \, dt$$

Discrete-Time Signal Power:

For a discrete-time signal x[n], the average power is defined as:

$$P = \lim_{N\to\infty} \frac{1}{N} \sum_{n=-N}^{N} |x[n]|^2$$

## Procedure:

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window

## Program:

```
% MATLAB Code for Energy and Power of Signals
clc; clear; close all;
 %% Define the time vector and signals
t = 0:0.01:10; % Time vector for continuous signal
```

```
n = 0:1:50;    % Index vector for discrete signal
 % Define a continuous-time signal (e.g., sine wave)
x_continuous = 5*sin(t)+3*cos(t); % 1 Hz sine wave
 % Define a discrete-time signal (e.g., exponential decay)
x_discrete = (0.9).^n; % Exponential signal
 %% Calculate energy and power of continuous signal
energy_continuous = trapz(t, abs(x_continuous).^2); % Energy using numerical integration
power_continuous = energy_continuous / (t(end) - t(1)); % Power = Energy / Time period
 %% Calculate energy and power of discrete signal
energy_discrete = sum(abs(x_discrete).^2); % Energy for discrete signal
power_discrete = energy_discrete / length(x_discrete); % Power = Energy / Length
 %% Display results
disp('--- Continuous-Time Signal ---');
disp(['Energy: ', num2str(energy_continuous)]);
disp(['Power: ', num2str(power_continuous)]);
 disp('--- Discrete-Time Signal ---');
disp(['Energy: ', num2str(energy_discrete)]);
disp(['Power: ', num2str(power_discrete)]);
 %% Plot signals
figure;
 % Plot continuous-time signal
subplot(2, 1, 1);
plot(t, x_continuous, 'b', 'LineWidth', 1.5);
title('Continuous-Time Signal (Sine Wave)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
 % Plot discrete-time signal
subplot(2, 1, 2);
stem(n, x_discrete, 'r', 'LineWidth', 1.5);
title('Discrete-Time Signal (Exponential Decay)');
```

```
xlabel('Index (n)');
ylabel('Amplitude');
grid on;
```

## Precautions:

1. Check out source file is with '.m' extension or not.

2. The file name should begin with character and should not contain any punctuation marks.

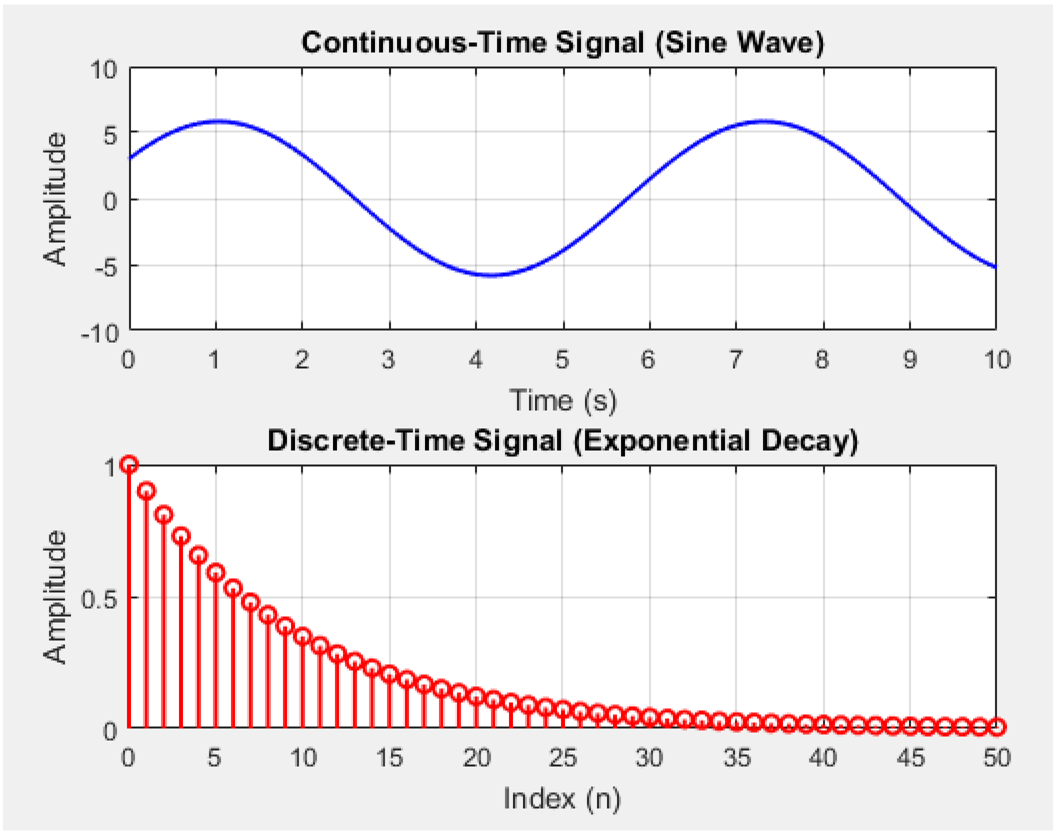
3. File name should not be any in built in function name or any keyword

4. Save the .m files preferably in work folder of MATLAB.

5. Don't delete any file or folder without informing the system administrator or lab in-charge.

## Output:

```
--- Continuous-Time Signal ---
Energy: 170.7876
Power: 17.0788
--- Discrete-Time Signal ---
Energy: 5.263
Power: 0.1032
```



**Result:** The Energy and average power of a given signal are estimated**.**

## Viva Questions:

1. Define energy and power signals.
2. Define even and odd signals.
3. Mention the condition for a signal to be called as periodic.
4. What type of signal is a sinusoidal signal in terms of energy and power?
5. State Parseval's theorem in relation to energy and power?

# EXPERIMENT-4

## Obtain the response of a system using convolution

**AIM:** To obtain the response of a system using Convolution and extract back the original signals.

**Software Used:** MATLAB R2016a

**Theory:** Convolution is a mathematical operation that combines two functions to produce a third function. It expresses how the shape of one function is modified by the other. In simple terms, it measures the overlap between one function and a reversed, shifted version of another function. In various fields, convolution is used to describe the process of applying a filter, transformation, or response function.

If the impulse signal [δ(t)] is input to the system, then output of the system is called the impulse response h(t) of the system and is given by

$$\mathrm{h(t)} = \mathrm{T}[\delta(\mathrm{t})]$$

As any arbitrary signal x(t) can be represented as

$$\mathrm{x(t)} = \int_{-\infty}^{\infty} \mathrm{x}(\tau)\,\delta(\mathrm{t}-\tau)\mathrm{d}\tau$$

Then the output can be given as

$$\mathrm{y(t)} = \mathrm{T}[\mathrm{x(t)}]$$

$$\Rightarrow \mathrm{y(t)} = \mathrm{T}\left[\int_{-\infty}^{\infty} \mathrm{x}(\tau)\,\delta(\mathrm{t}-\tau)\mathrm{d}\tau\right]$$

For a continuous-time linear system, the output is given by,

$$\mathrm{y(t)} = \int_{-\infty}^{\infty} \mathrm{x}(\tau)\,\mathrm{T}\left[\delta(\mathrm{t}-\tau)\right]\mathrm{d}\tau$$

$$\mathrm{y(t)} = \mathrm{x(t)} * \mathrm{h(t)} = \int_{-\infty}^{\infty} \mathrm{x}(\tau)\,\mathrm{h}(\mathrm{t}-\tau)\mathrm{d}\tau$$

**Procedure:**

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window

**Program:**

```
close all;
clear;
 clc;
% Program for convolution and Deconvolution
 T = 2;
tmin = 0;
tmax = 2*T; d
t = 0.01;
t = tmin:dt:tmax;
x1 = exp(-0.7*t).*(t>=0 & t<T);
x2 = 1.*(t>=0 & t<T);
x3 = conv(x1,x2); n3 = length(x3); tl = 0:1:n3-1; subplot(3,2,1);
plot(t, x1);
xlabel('t');
```

```
ylabel('xl(t)');
title('First Signal xl(t)');
subplot(3,2,2);
plot(t, x2);
xlabel('t');
ylabel('x2(t)');
title('Second Signal x2(t)');
subplot(3,2,3);
plot(tl, x3);
xlabel('t'); ylabel('x3(t)');
title('Convolution of xl(t) and x2(t)');

x1d = deconv(x3,x2);
x2d = deconv(x3,x1);
ymin = min([min(x1d), min(x2d)]);
 ymax = max([max(x1d), max(x2d)]);
subplot(3,2,5);
plot(t,x1d);
axis([tmin tmax ymin ymax]);
xlabel('t');
ylabel('xl(t)');
title('Extracted First Signal')
subplot(3,2,6);
plot(t, x2d);
axis([tmin tmax ymin ymax]);
xlabel('t');
ylabel('x2(t)');
```

```
title('Extracted second Signal')
```

**Precautions:**

1. Check out source file is with '.m' extension or not.

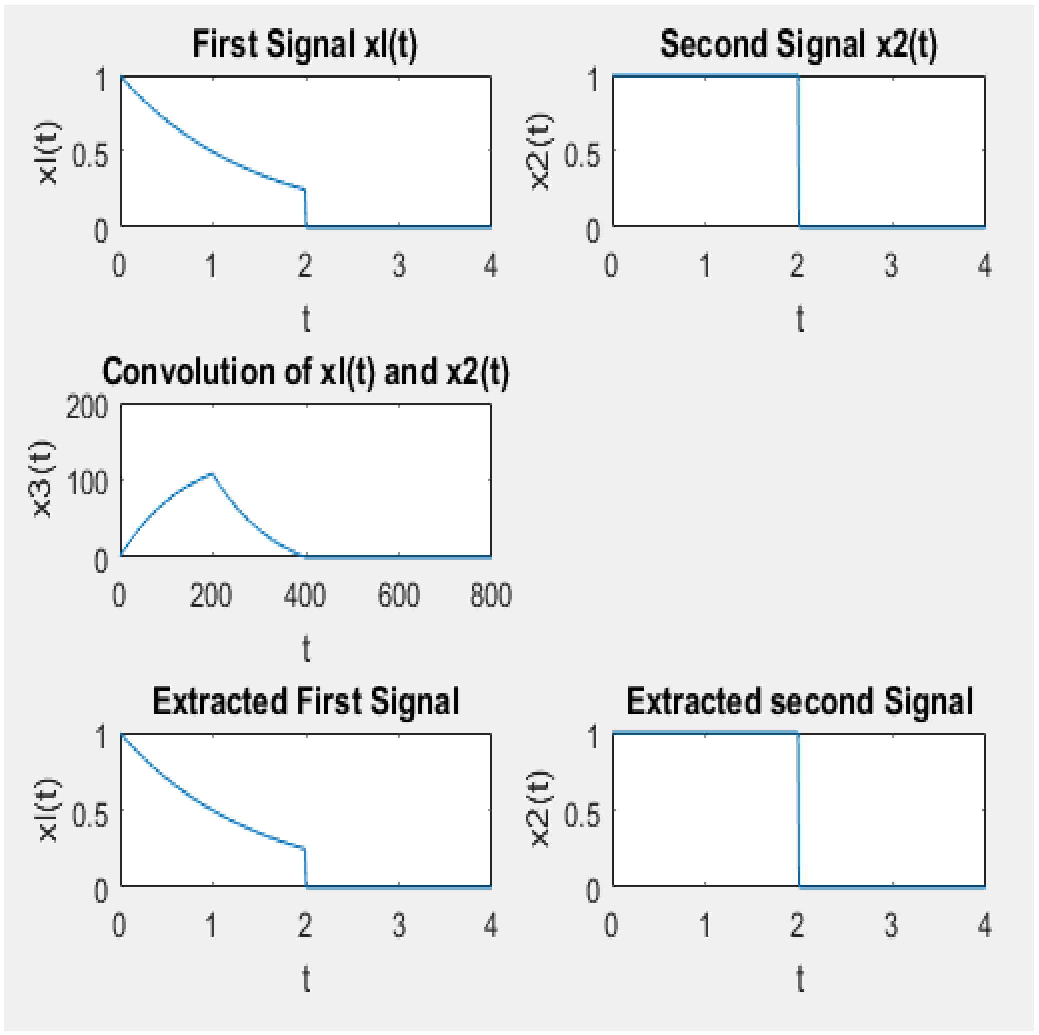
2. The file name should begin with character and should not contain any punctuation marks.

3. File name should not be any in built in function name or any keyword

4. Save the .m files preferably in work folder of MATLAB.

5. Don't delete any file or folder without informing the system administrator or lab in-charge.

## Output:



**Result:** The response of a system using Convolution and extract back the original signals is performed.

**Viva Questions:**

1. Define convolution
2. What are the operations in convolution?
3. What is the convolution of u(t) with u(t)?
4. State time Convolution theorem.
5. Mention the areas of applications of convolution.

# EXPERIMENT-5
## Perform correlation between signals

**AIM:** To perform the correlation between given signals.

**Software Used:** MATLAB R2016a

**Theory:**

Correlation is a statistical measure that quantifies the degree to which two variables are related. It provides insight into how one variable changes in response to changes in another. Correlation is widely used in statistics, data analysis, signal processing, and machine learning to identify relationships and patterns.

**There are two types of correlations –**

- Cross-correlation
- Autocorrelation

**Cross-correlation**

The cross-correlation between two different signals or functions or waveforms is defined as the measure of similarity or coherence between one signal and the time-delayed version of another signal. The cross-correlation between two different signals indicates the degree of relatedness between one signal and the time-delayed version of another signal.

The cross-correlation of energy (or aperiodic) signals and power (or periodic) signals is defined separately.

Consider two complex signals $x_1(t)$ and $x_2(t)$

$$R_{12}(\tau)=\frac{1}{T}\int_{-(T/2)}^{(T/2)} x_1(t)x_2^*(t-\tau)\,dt$$

**Autocorrelation**

The autocorrelation function is defined as the measure of similarity or coherence between a signal and its time delayed version. Therefore, the autocorrelation is the correlation of a signal with itself.

Like cross-correlation, autocorrelation is also defined separately for energy (or aperiodic) signals and power (periodic) signals.

The autocorrelation a signal x(t) is defined as

$$R(\tau) = \lim_{T \to \infty} \frac{1}{T} \int_{-(T/2)}^{(T/2)} x(t)x^*(t-\tau)\, dt$$

**Procedure:**

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window

## Program:

```
% Define time parameters
T = 10;            % Duration of the signals (seconds)
dt = 0.01;         % Time step for discretization
%t = -T:dt:T;       % Time vector

% Define the continuous-time signals (replace with your own)
t = -10:0.01:10; % Time range from -5 to 5 seconds with step size of 0.001

% Unit step signal (u(t))
x = double(t >= 0);

y = double(t >= 0);

% Initialize the correlation result
tau = -2*T:dt:2*T; % Range of lags (tau)
```

```
r = zeros(size(tau)); % Correlation result
% Compute correlation manually
for i = 1:length(tau)
    shift = tau(i); % Current lag
    y_shifted = interp1(t, y, t - shift, 'linear', 0); % Shift y(t) by tau
    r(i) = sum(x .* y_shifted) * dt; % Compute the integral using Riemann sum
end
% Plot the original signals
figure;
subplot(3,1,1);
plot(t, x);
title('Signal x(t)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
subplot(3,1,2);
plot(t, y);
title('Signal y(t)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
% Plot the correlation
subplot(3,1,3);
plot(tau, r);
title('Correlation r(\tau)');
xlabel('Lag (\tau)');
ylabel('Correlation');
grid on;
```

**Precautions:**

1. Check out source file is with '.m' extension or not.

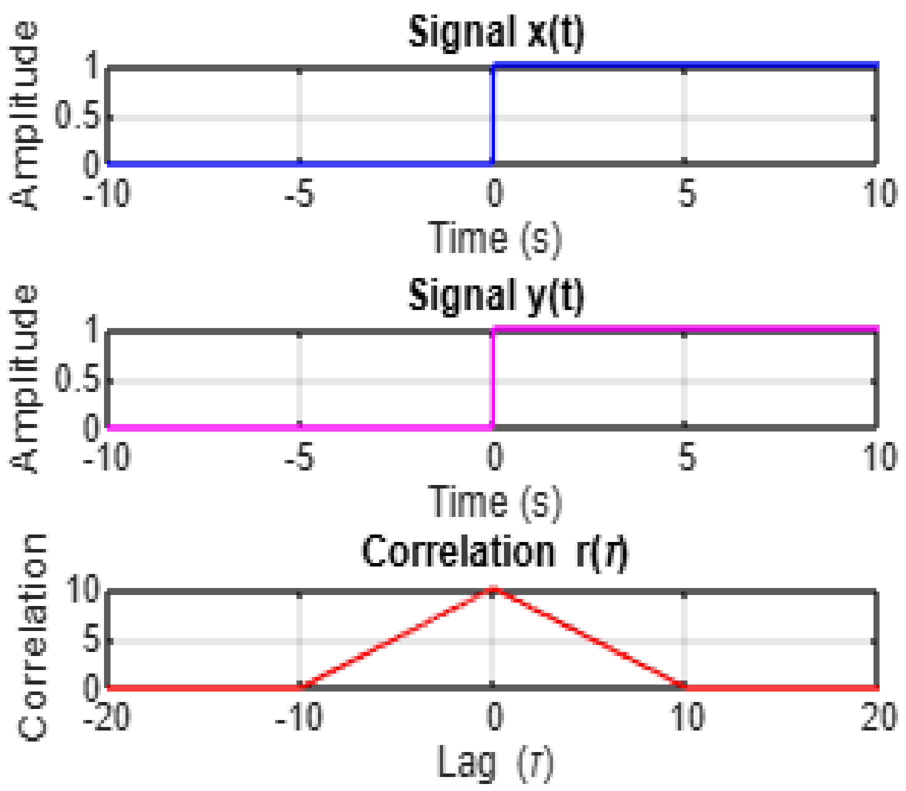
2. The file name should begin with character and should not contain any punctuation marks.

3. File name should not be any in built in function name or any keyword

4. Save the .m files preferably in work folder of MATLAB.

5. Don't delete any file or folder without informing the system administrator or lab in-charge.

**Output:**



**Result:** Correlation between given signals is performed.

1. Define correlation
2. What is auto correlation and cross correlation?
3. State Winer -Khnichine relation.
4. State the applications where auto correlation is used.
5. What is the difference between convolution and correlation?

## EXPERIMENT-6

## Estimation of Fourier coefficients of a given periodic signal

**AIM:** To obtain the trigonometric and exponential Fourier series coefficients of a given periodic signal

**Software Used:** MATLAB R2016a

**Theory:**

The Fourier Series is a mathematical tool used to represent a periodic signal as a sum of sine and cosine functions (or equivalently, complex exponentials). It is a cornerstone of signal processing, allowing A periodic function f(t) with period T can be expressed as periodic functions to be analyzed in terms of their frequency components.

$$f(t) = a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos\left(\frac{2\pi n t}{T}\right) + b_n \sin\left(\frac{2\pi n t}{T}\right) \right]$$

**Procedure:**

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window

**Program:**

```
close all
```

```
clear all

clc

% Define the function

f = @(t) t.^2;

% Define the period

 T = 2 * pi;

 w0 = 2 * pi / T; % Fundamental angular frequency

% Define the number of harmonics

N = 10;

% Initialize the coefficients

a0 = (1 / T) * integral(@(t) f(t), 0, T); an = zeros(N, 1);

bn = zeros(N, 1);

% Compute the coefficients

for n = 1:N

an(n) = (2 / T) * integral(@(t) f(t) .* cos(n * w0 * t), 0, T);

bn(n) = (2 / T) * integral(@(t) f(t) .* sin(n * w0 * t), 0, T);

end

subplot(2,1,1)

stem(an,'r')

xlabel('nw');

ylabel('amplitude');

title('an');

subplot(2,1,2)
```

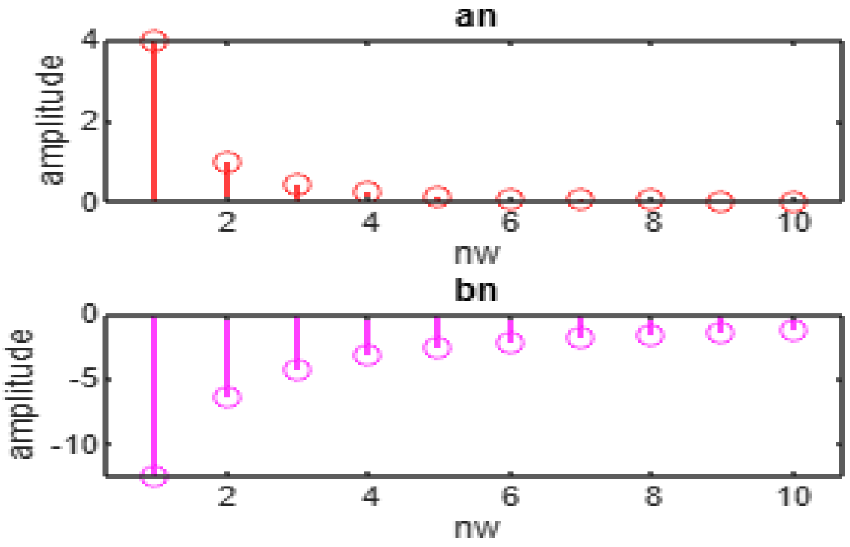
```
stem(bn,'m')

xlabel('nw');

ylabel('amplitude');

title('bn');

%Exponential Fourier Series

% Define the function

f = @(t) t.^2;

% Define the period

T = 2 * pi;

w0 = 2 * pi / T; % Fundamental angular frequency

% Define the number of harmonics

N = 10;

% Initialize the coefficients

c = zeros(2*N+1, 1); % c_{-N} to c_{N}

% Compute the coefficients

for n = -N:N

integrand = @(t) f(t) .* exp(-1j * n * w0 * t);

c(n+N+1) = (1 / T) * integral(@(t) f(t) .* exp(-1j * n * w0 * t), 0, T);

end

subplot(2,1,1);

stem(abs(c))

xlabel('nw');

ylabel('amplitude');
```

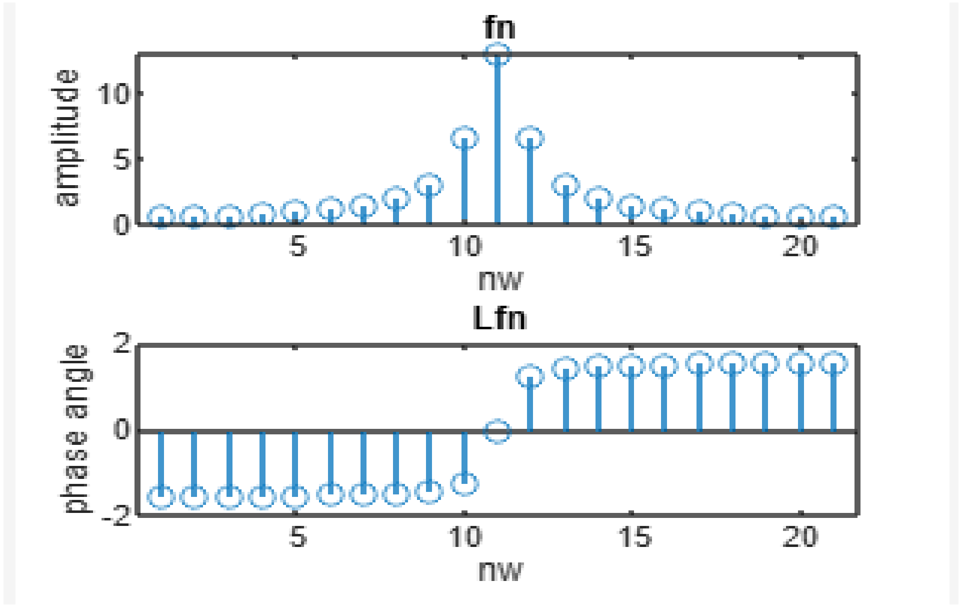
```
title('fn');

subplot(2,1,2);

stem(angle(c))

xlabel('nw');

ylabel('phase angle');

title('Lfn');
```

**Precautions:**

1. Check out source file is with '.m' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Save the .m files preferably in work folder of MATLAB.
5. Don't delete any file or folder without informing the system administrator or lab in-charge.

**Outputs:**





**Result:** Spectrum of trigonometric and exponential Fourier series of given continuous time signal are plotted.

**Viva Questions:**

1.why Fourier series is needed?
2.What are the conditions needed to be satisfied for existence of Fourier Series?
3.What are the different types of Fourier Series?
4.State the relation between Trigonometric and Exponential Fourier series?
5.What is line spectrum?

# EXPERIMENT-7
# Analysis of Fourier spectrum using Fourier Transform

**AIM:** To analyze the Fourier spectrum using Fourier transform and obtain the dominant frequencies

**Software Used:** MATLAB R2016a

**Theory:**

The main drawback of Fourier series is, it is only applicable to periodic signals. There are some naturally produced signals such as nonperiodic or aperiodic, which we cannot represent using Fourier series. To overcome this shortcoming, Fourier developed a mathematical model to transform signals between time (or spatial) domain to frequency domain & vice versa, which is called 'Fourier transform

The Fourier Transform decomposes any function into a sum of sinusoidal basis functions. Each of these basis functions is a complex exponential of a different frequency. The Fourier Transform therefore gives us a unique way of viewing any function as the sum of simple sinusoids. It converts a signal from the time domain to the frequency domain. Fourier transform of a signal is given as

$$f(t) = F[\omega] = [\int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt]$$

**Procedure:**

1. Open the MATLAB software by double clicking the icon on desktop.

2. Open the new M-file by using file menu.

3. Write the program in new file.

4. Click on save and run the icon.

5. Perform error check which displayed on command window.

6. Plot the waveforms displayed on figure window.

7. Note down the values, which are displayed on the command window

**Program:**

```
% Define parameters
Fs = 1000;          % Sampling frequency (Hz)
T = 1/Fs;           % Sampling period (s)
L = 1000;           % Length of the signal (number of samples)
t = (0:L-1)*T;      % Time vector

% Define the signal (replace with your own signal)
% Example: a signal with two sine waves of different frequencies
f1 = 50;            % Frequency of first sine wave (Hz)
f2 = 120;           % Frequency of second sine wave (Hz)
signal = 0.7*sin(2*pi*f1*t) + 0.5*sin(2*pi*f2*t);

% Add noise to the signal
noise = 0.2 * randn(size(t)); % Random noise
signal_noisy = signal + noise;

% Plot the time-domain signal
figure;
subplot(2,1,1);
plot(t, signal_noisy);
title('Time-Domain Signal');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

% Perform Fourier Transform
Y = fft(signal_noisy);

% Compute the two-sided spectrum and then the single-sided spectrum
P2 = abs(Y/L);      % Two-sided spectrum
P1 = P2(1:L/2+1);   % Single-sided spectrum
P1(2:end-1) = 2*P1(2:end-1);

% Frequency vector for plotting
f = Fs*(0:(L/2))/L;

% Plot the frequency-domain spectrum
subplot(2,1,2);
plot(f, P1);
title('Frequency-Domain Spectrum (Fourier Transform)');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
grid on;

% Highlight the dominant frequencies
```

```
[~, locs] = findpeaks(P1, 'MinPeakHeight', 0.1); % Adjust threshold if needed
disp('Dominant frequencies (Hz):');
disp(f(locs));
```

**Precautions:**

1. Check out source file is with '.m' extension or not.

2. The file name should begin with character and should not contain any punctuation marks.

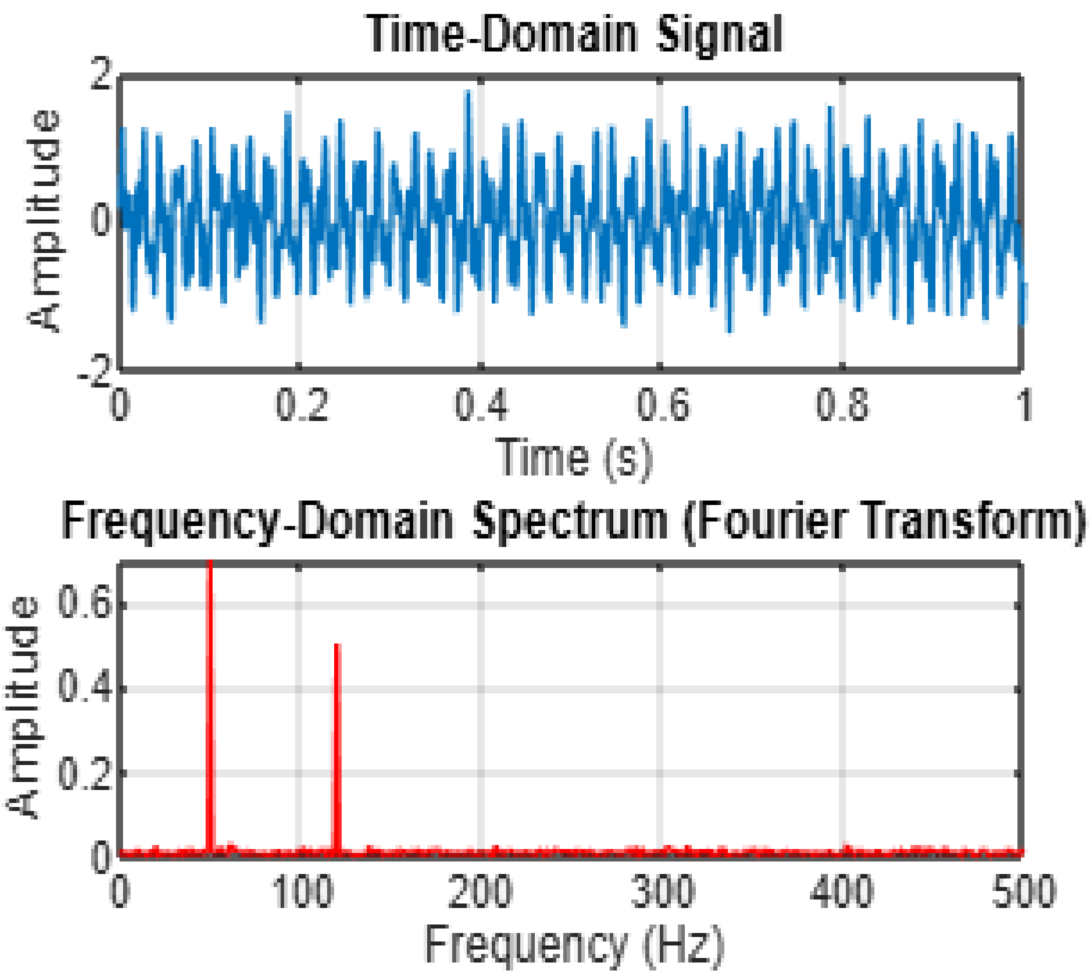
3. File name should not be any in built in function name or any keyword

4. Save the .m files preferably in work folder of MATLAB.

5. Don't delete any file or folder without informing the system administrator or lab in-charge.

**Output:**

```
Dominant frequencies (Hz):
    50   120
```



**Result:** The Fourier spectrum is analyzed using Fourier transform and the dominant frequencies were obtained.

**Viva Questions:**

**1.** Define Fourier Transform

2. What is the difference between Fourier transform and Fourier Series?

3. State the condition for existence of Fourier transform.

4. What is the difference between Laplace Transform and Fourier transform?

5. Mention the Fourier Transform of standard functions.

# EXPERIMENT-8

## Estimation of Laplace transform of an arbitrary function

**AIM:** To estimate the Laplace transform of given arbitrary function

**Software Used:** MATLAB R2016a

**Theory:**

The Laplace Transform is a mathematical technique widely used in engineering, physics, and applied mathematics to transform a time-domain function f(t) into a complex frequency-domain representation F(s). It is especially useful for solving linear differential equations and analyzing systems.

The Laplace Transform of a function f(t) is given as

$$F(s) = \mathcal{L}\{f(t)\} = \int_0^\infty e^{-st} f(t)\, dt$$

**Procedure:**

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window

**Program:**

```
% Define the arbitrary function as a function handle
syms t s;
f = input('Enter the function f(t): '); % Example: sin(t), exp(-t), t^2, etc.
% Compute the Laplace Transform
```

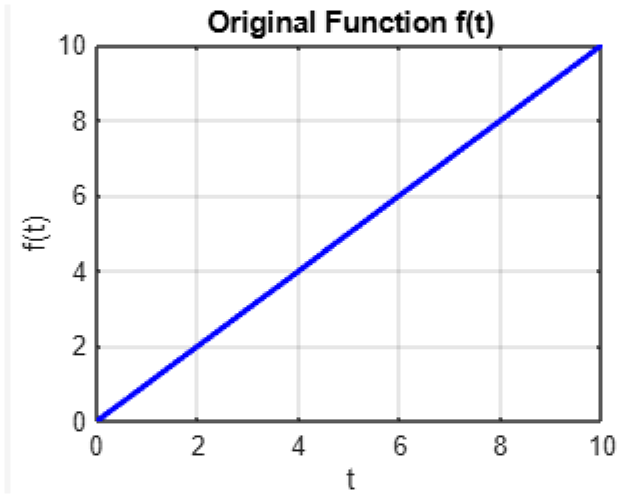
```
laplace_transform = laplace(f, t, s);
% Display the result
disp('The Laplace Transform of the function is:');
disp(laplace_transform);
% Plot the original function
f_handle = matlabFunction(f); % Convert symbolic function to MATLAB function handle
t_vals = linspace(0, 10, 100); % Time values for plotting
f_vals = f_handle(t_vals);
figure;plot(t_vals, f_vals, 'b', 'LineWidth', 2);
xlabel('t');
ylabel('f(t)');
title('Original Function f(t)');
grid on;
% Displaying the symbolic Laplace Transform as a 2D function
laplace_handle = matlabFunction(laplace_transform);
s_vals = linspace(1, 10, 100);
laplace_vals = laplace_handle(s_vals);
figure; plot(s_vals, laplace_vals, 'r', 'LineWidth', 2);
xlabel('s');
ylabel('F(s)');
title('Laplace Transform F(s)');
grid on;
```

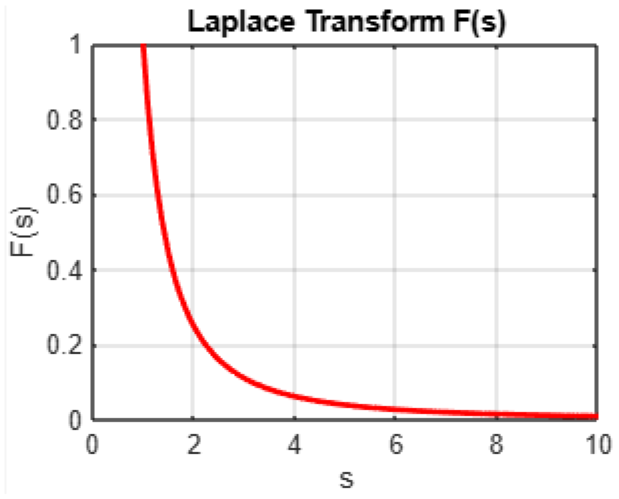
**Precautions:**

1. Check out source file is with '.m' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Save the .m files preferably in work folder of MATLAB.
5. Don't delete any file or folder without informing the system administrator or lab in-charge.

**Outputs:**

```
Enter the function f(t):
t
The Laplace Transform of the function is:
1/s^2
```





**Result:** Laplace Transform of given arbitrary function was obtained.

**Viva Questions:**

1. Define Laplace Transform?
2. What is Region of Convergence?
3. What are the properties of RoC?
4. Why do we need Laplace transform if Fourier transform is existing?
5. Estimate the Laplace transform of standard functions?

## EXPERIMENT-9
## Estimation of Z- transform of an arbitrary function

**AIM:** To estimate the Z-transform a given arbitrary sequence

**Software Used:** MATLAB R2016a

**Theory:**

The Z-transform (ZT) is a mathematical tool which is used to convert the difference equations in time domain into the algebraic equations in z-domain.
The Z-transform is a very useful tool in the analysis of a linear shift invariant (LSI) system. An LSI discrete time system is represented by difference equations. To solve these difference equations which are in time domain, they are converted first into algebraic equations in z-domain using the Z-transform, then the algebraic equations are manipulated in z-domain and the result obtained is converted back into time domain using the inverse Z-transform.
The **Z-Transform** of a discrete-time signal x[n] is defined as

$$X(z) = \mathcal{Z}\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n] z^{-n}$$

**Procedure:**

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window

**Program:**

```
% Z-Transform Estimation in MATLAB

clc; clear; close all;

% Define the arbitrary sequence as a symbolic function

syms n z;

sequence = input('Enter the sequence x[n] (as a function of n): '); % Example: n, 2^n, sin(n), etc.

% Compute the Z-Transform

z_transform = ztrans(sequence, n, z);

% Display the Z-Transform

disp('The Z-Transform of the sequence is:');

disp(z_transform);

% Plot the sequence x[n]

n_vals = 0:10; % Sample values for n (discrete points)

x_n = subs(sequence, n, n_vals); % Evaluate x[n] for specific n values

figure;stem(n_vals, double(x_n), 'b', 'LineWidth', 1.5, 'MarkerFaceColor', 'b');

xlabel('n');

ylabel('x[n]');

title('Original Sequence x[n]');

grid on;

% Plot the Z-Transform magnitude in the Z-plane

f = matlabFunction(z_transform); % Convert symbolic Z-transform to a function handle

z_vals = linspace(0.5, 2, 100) + 1j * linspace(-1, 1, 100); % Complex Z-plane values

[z_real, z_imag] = meshgrid(real(z_vals), imag(z_vals));

z_complex = z_real + 1j * z_imag;

Z = arrayfun(@(z) f(z), z_complex); % Evaluate Z-transform

figure;surf(real(z_complex), imag(z_complex), abs(Z), 'EdgeColor', 'none');

xlabel('Real(Z)');

ylabel('Imag(Z)');

zlabel('|X(Z)|');

title('Magnitude of Z-Transform in Z-Plane');
```

colorbar;

view(3);

grid on;

**Precautions:**

1. Check out source file is with '.m' extension or not.

2. The file name should begin with character and should not contain any punctuation marks.

3. File name should not be any in built in function name or any keyword

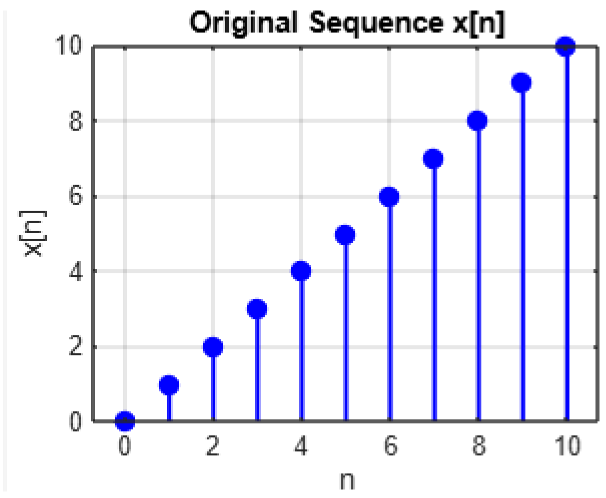
4. Save the .m files preferably in work folder of MATLAB.

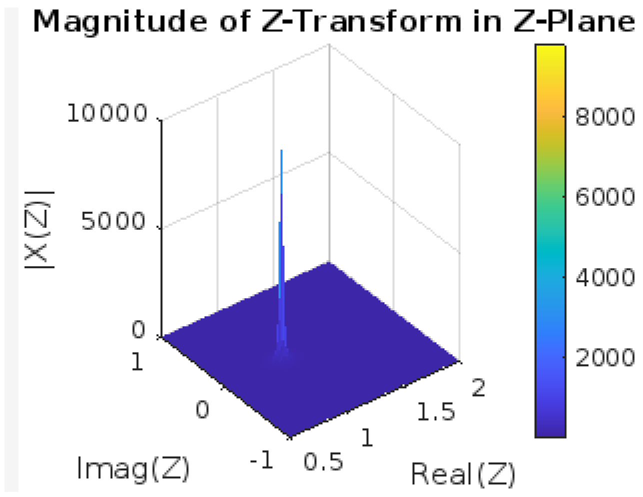
5. Don't delete any file or folder without informing the system administrator or lab in-

charge.

**Outputs:**

```
Enter the sequence x[n] (as a function of n):
n
The Z-Transform of the sequence is:
z/(z - 1)^2
```





**Result**: Z-transform of a given arbitrary sequence was obtained**.**

**Viva Questions:**

1. Define Z transform
2. What is the RoC of Z transform?
3. State the applications of Z transform
4. Mention the Z transform of Standard functions.
5. State the time convolution property of Z transform.

# EXPERIMENT-10

## Estimation of power Spectral density for noise signals

**AIM:** To Estimate the power Spectral density for noise signals

**Software Used:** MATLAB R2016a

**Theory:**

Power Spectral Density is the distribution of average power of a signal x(t) in the frequency domain is called the power spectral density (PSD) or power density (PD) or power density spectrum. The PSD function is denoted by S(ω) and is given by

$$S(\omega) = \lim_{\tau \to \infty} \frac{|X(\omega)|^2}{\tau}$$

**Procedure:**

1. Open the MATLAB software by double clicking the icon on desktop.
2. Open the new M-file by using file menu.
3. Write the program in new file.
4. Click on save and run the icon.
5. Perform error check which displayed on command window.
6. Plot the waveforms displayed on figure window.
7. Note down the values, which are displayed on the command window

**Program:**

```
clc;
clear;
close all;
% Parameters
Fs = 1000;            % Sampling frequency (Hz)
```

```
T = 1;                  % Signal duration (seconds)
N = Fs * T;             % Number of samples
t = (0:N-1) / Fs;       % Time vector

% Generate a noise signal
noise_signal = randn(1, N); % White Gaussian noise

% Plot the noise signal
figure;
plot(t, noise_signal);
title('Noise Signal');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

% Compute the FFT of the signal
X = fft(noise_signal);          % Compute the FFT
X_mag = abs(X).^2;              % Compute the magnitude squared (power)

% Normalize the power spectrum
PSD = X_mag / (Fs * N);         % Scale by sampling frequency and length
frequencies = (0:N-1) * (Fs / N); % Frequency vector

% Use only the positive half of the spectrum
half_idx = 1:N/2+1;
PSD_half = PSD(half_idx);
frequencies_half = frequencies(half_idx);

% Plot the PSD
figure;
plot(frequencies_half, 10*log10(PSD_half));
```

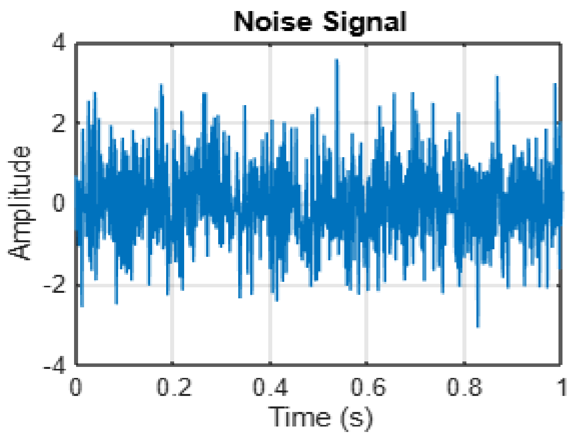
```
title('Power Spectral Density Estimate (Manual Calculation)');
xlabel('Frequency (Hz)');
ylabel('Power/Frequency (dB/Hz)');
grid on;

% Segment-based averaging for better PSD estimation
segment_length = 256;           % Length of each segment
overlap = 128;                  % Overlap between segments
step = segment_length - overlap;    % Step size for sliding window
num_segments = floor((N - overlap) / step); % Number of segments
PSD_avg = zeros(1, segment_length/2+1); % Initialize PSD accumulator

% Loop through segments
for k = 1:num_segments
    start_idx = (k-1) * step + 1;
    end_idx = start_idx + segment_length - 1;
    segment = noise_signal(start_idx:end_idx); % Extract segment

    % Compute FFT for the segment
    X_segment = fft(segment .* hamming(segment_length)'); % Apply window
    X_mag_segment = abs(X_segment).^2;             % Power spectrum
    PSD_segment = X_mag_segment / (Fs * segment_length); % Normalize

    % Accumulate the positive half of the PSD
    PSD_avg = PSD_avg + PSD_segment(1:segment_length/2+1);
end

% Average the PSD
PSD_avg = PSD_avg / num_segments;
frequencies_avg = (0:segment_length/2) * (Fs / segment_length);
```

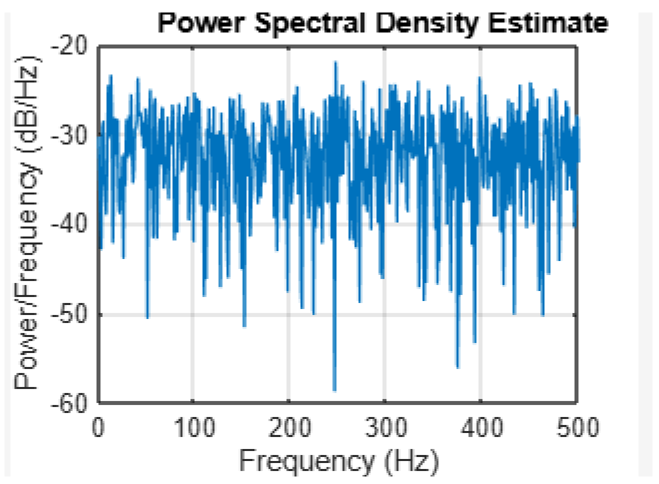
```
% Plot the averaged PSD
figure;
plot(frequencies_avg, 10*log10(PSD_avg));
title('Averaged Power Spectral Density Estimate');
xlabel('Frequency (Hz)');
ylabel('Power/Frequency (dB/Hz)');
grid on;
```

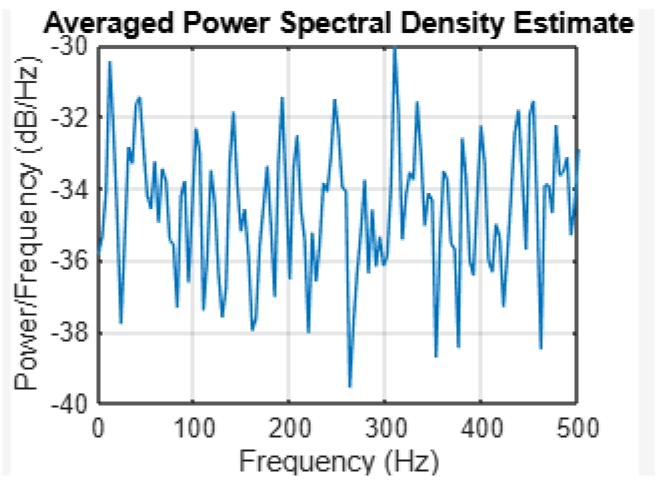
**Precautions:**

1. Check out source file is with '.m' extension or not.
2. The file name should begin with character and should not contain any punctuation marks.
3. File name should not be any in built in function name or any keyword
4. Save the .m files preferably in work folder of MATLAB.
5. Don't delete any file or folder without informing the system administrator or lab in-charge.

**Output:**







**Result:** The power Spectral density for noise signals is estimated.

**Viva Questions:**

1. Define power Spectral density
2. What is the relation between Power spectral density and auto correlation?
3. How to estimate power using power spectral density?
4. Define average power.
5. Define cross power.