NITTE (Deemed to be University) | NMAM INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

SIGNAL ANALYSIS AND PROCESSING (IPCC) LABORATORY MANUAL

(COURSE CODE: EE3003-1)

NAME: _____

USN: _____

BATCH: _____

Contents

Experiments: **Date:**

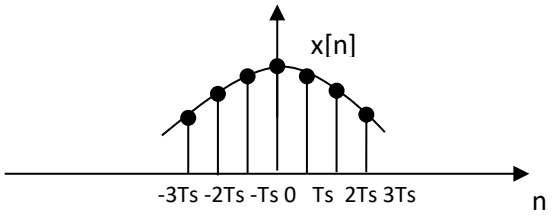
Experiment 1 Realization of Sampling theorem and Aliasing and Quantization

AIM: To Verify the Sampling Theorem in Time Domain and Frequency Domain and show the effect of aliasing.

Theory:

Concept of Sampling:

Sampling converts a continuous time signal into a discrete time signal. A discrete time signal is often derived from a continuous time signal by sampling at uniform rate.



Sampling period be Ts.

'n' is an integer may be positive or negative.

x[n]= x[nTs] where n=0, ±1, ±2,…..

Sampling theorem states that **if a continuous time signal is sampled for all time at a rate f_s, which is more than twice the band limit f_m of the signal, the original continuous time signal can be recovered exactly from the samples.**

If the highest frequency present in a signal is f_m , the minimum rate at which a signal can be sampled and still be reconstructed from its samples is above $2f_m$ and the frequency $2f_m$ is called ***Nyquist rate****. When a signal is sampled at a rate f_s, the frequency $f_s/2$ is called the* ***Nyquist frequency.***

Concept of Aliasing and process of Reconstruction:

The aliasing concept can be explained as follows

Consider a sinusoidal signal of 50 Hz (T=20 msec) and 100 Hz (T=10msec) Samples are taken at a rate of 100 Hz.

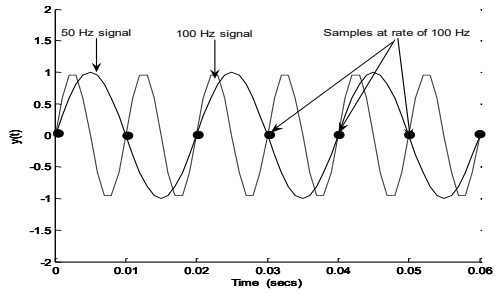


Figure 1: Signals with 50Hz and 100 Hz frequencies

The Discrete Time signal obtained for both the signals are same. If the signals are sampled at 200 kHz to meet the Nyquist rate, then obtained waveforms are as shown

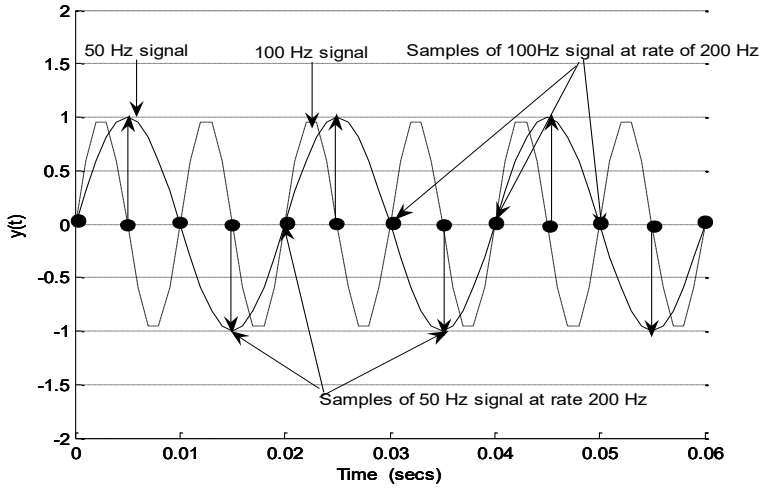


Figure 2: Sampled Signal at 200 Hz

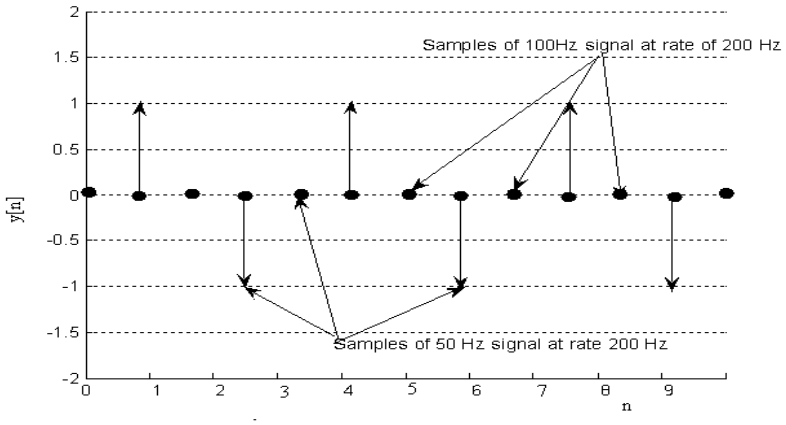


Figure 2: Discretized 100 Hz signal with sampling frequency of 200 Hz

From the samples obtained, it can be inferred that they are distinct and there is no aliasing effect. Hence original signals can be easily recovered knowing the sampling frequency.

Very often, the information can be extracted only in the lower frequency components of a signal. Therefore, the actual signal can be passed through a low pass filter to attenuate high frequency components in the signal. Then the signal can be sampled a rate less than twice the highest frequency that is actually present in the signal. The continuous time low pass filter is termed as ***antialiasing filter***.

Experiment 1a: Illustrate the effect of choosing sampling frequency of 15kHz , 20kHz & 50kHz on 10kHz sinusoidal signal

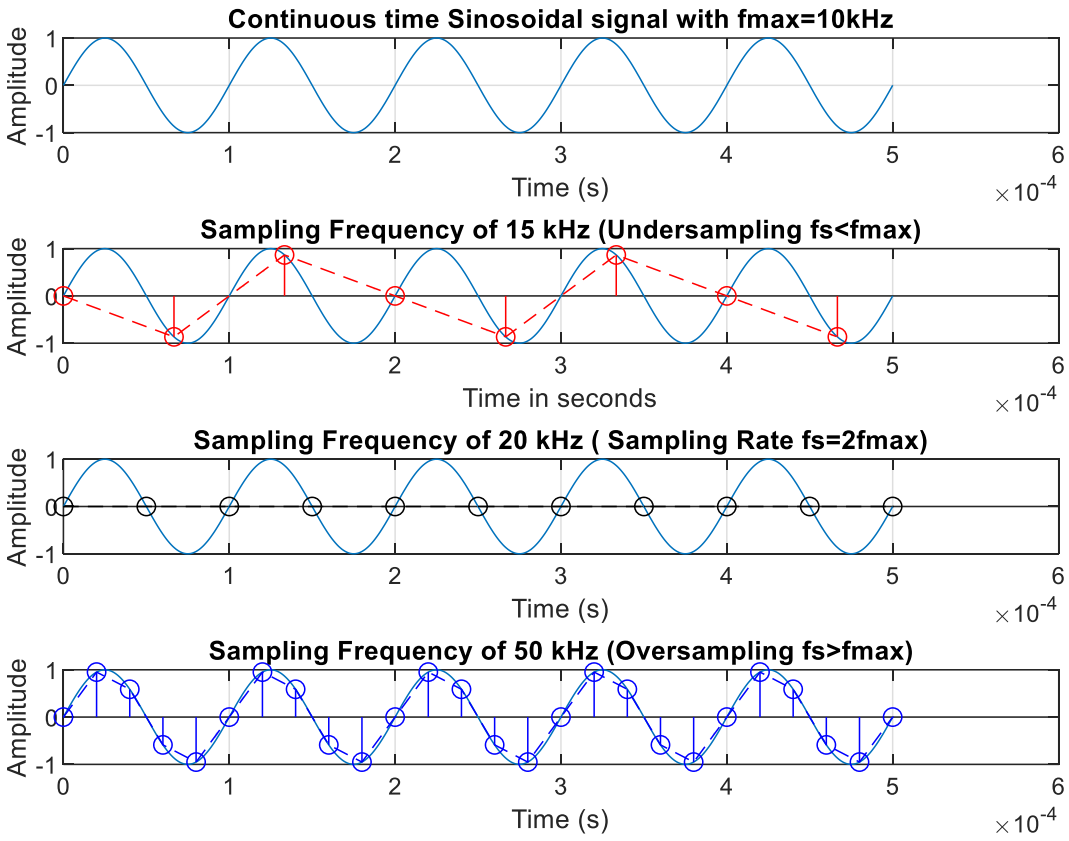
PROGRAM:

//Verification of sampling theorem in Time Domain

```
% MATLAB Program: Effect of Sampling Frequency on a 10kHz Sinusoidal Signal
clc;
clear;
close all;

% Parameters
f = 10e3; % Frequency of sinusoid (10 kHz)
ncyl = 5; % Generate five cycles of sinusoid
% Time index for continuous time signal
T_cont = 1/f/100; % Very small time step for continuous signal
% Continuous Time Signal
t = 0: T_cont: ncyl*(1/f); % Time vector for continuous signal
x = sin(2*pi*f*t);
figure;
subplot(4,1,1);
plot(t,x);
title(' Continuous time Sinosoidal signal with fmax=10kHz');

xlabel('Time (s)');
ylabel('Amplitude');
grid on;

% Sampling at 15 kHz
fs1 = 15e3; % Sampling frequency 15 kHz
t1 = 0:1/fs1:ncyl*(1/f); % Time vector for sampled signal
x1 = sin(2*pi*f*t1);
subplot(4,1,2);
plot(t,x);
hold on;
stem(t1,x1,'r');
hold on
% Interpolate
x1_interp = interp1(t1,x1,t,'linear');
plot(t, x1_interp,'r--');
title('Sampling Frequency of 15 kHz (Undersampling fs<fmax)');
xlabel('Time in seconds');
ylabel('Amplitude');
```

```
grid on;

% Sampling at 20 kHz
fs2 = 20e3; % Sampling frequency 20 kHz
t2 = 0:1/fs2:ncyl*(1/f);
x2 = sin(2*pi*f*t2);
subplot(4,1,3);
plot(t,x);
hold on;
stem(t2,x2,'k');
% Interpolate
x2_interp = interp1(t2,x2,t,'linear');
plot(t, x2_interp, 'k--');
title('Sampling Frequency of 20 kHz ( Sampling Rate fs=2fmax)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

% Sampling at 50 kHz
fs3 = 50e3; % Sampling frequency 50 kHz
t3 = 0:1/fs3:ncyl*(1/f);
x3 = sin(2*pi*f*t3);
subplot(4,1,4);
plot(t,x);
hold on;
stem(t3,x3,'b');

% Interpolate
x3_interp = interp1(t3,x3,t,'linear');
plot(t, x3_interp, 'b--');
title('Sampling Frequency of 50 kHz (Oversampling fs>fmax)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
hold off
```

Results:



Task:

1).Find the discrete time signals for the $x(t)$ given in the equation (1) for sampling frequencies of a) fs=20kHz b) Fs=50kHz

$$x(t) = \sin(2\pi 50t) \quad (1)$$

2) Find the discrete time signals for the $x(t)$ given in the equation (1) for sampling frequencies of a) fs=20kHz b) Fs=50kHz

$x(t) = \sin(480\pi t) + 3\sin(720\pi t)$ is sampled at 600 times/second

a) Determine Nyquist sampling rate for $x(t)$
b) Determine folding frequency and check for aliasing
c) Determine the Discrete time signal x[n]

Activity With Audio Signal:
Activity 1:

```
% MATLAB Code to Load an Audio Clip, Check Frequency Content, and Apply Sampling

% Load an audio file (Replace 'audio_sample.wav' with your actual file)
[audio_signal, fs_record] = audioread('audio_sample.wav');  % Read audio file
duration = length(audio_signal) / fs_record;  % Calculate the duration of the audio

% Check the frequency content of the audio using FFT
n = length(audio_signal);  % Length of the signal
f = (0:n-1)*(fs_record/n);  % Frequency axis
Y = fft(audio_signal);     % Perform FFT

% Plot the frequency spectrum of the audio signal
figure;
subplot(2,1,1);
plot(f(1:n/2), abs(Y(1:n/2)));  % Plot magnitude of the first half of the FFT (positive frequencies)
title('Frequency Content of the Audio');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Define low and high sampling rates for resampling
fs_low = 1000;   % Low sampling rate (causes aliasing)
fs_high = 10000; % High sampling rate (no aliasing)

% Resample audio signal
audio_low = resample(audio_signal, fs_low, fs_record);   % Low rate
audio_high = resample(audio_signal, fs_high, fs_record); % High rate

% Play the original, low rate, and high rate audio signals
disp('Playing original audio...');
sound(audio_signal, fs_record);  % Play original audio
pause(duration);  % Wait for original audio to finish

disp('Playing low rate audio (Aliasing)...');
sound(audio_low, fs_low);  % Play audio at low rate (with aliasing)
pause(duration);  % Wait for low rate audio to finish
```

```
disp('Playing high rate audio (No Aliasing)...');
sound(audio_high, fs_high);  % Play audio at high rate (no aliasing)
```

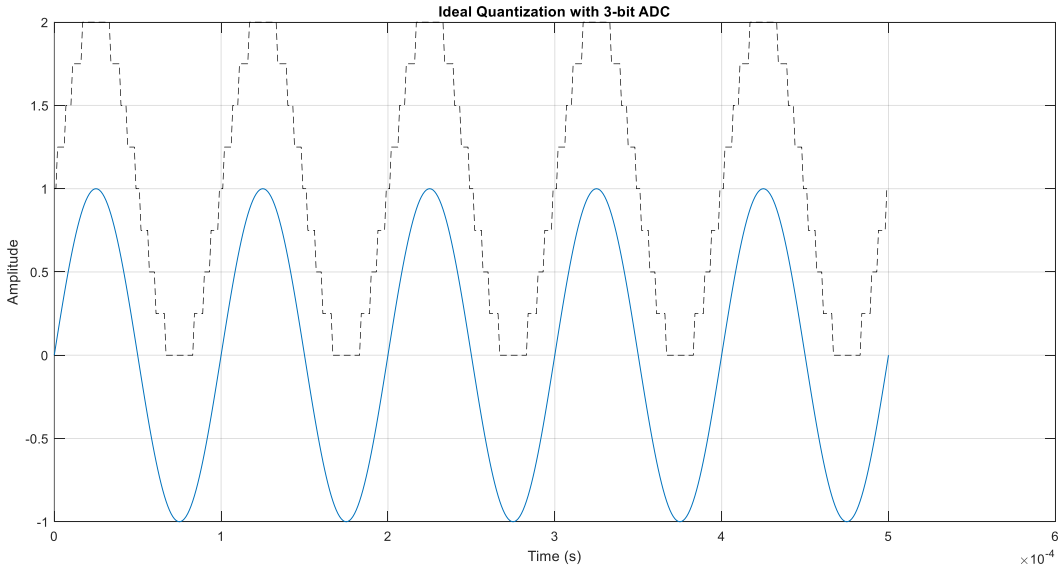
Activity 2:

```
% MATLAB Code to Capture Audio from Microphone and Demonstrate Aliasing

% Clear previous data
clear;
clc;

% Record audio from microphone (5 seconds)
fs_record = 44100;  % Sampling rate for recording
duration = 5;       % Duration in seconds
recObj = audiorecorder(fs_record, 16, 1);  % 16-bit mono
disp('Recording audio...');
recordblocking(recObj, duration);  % Record for specified duration
audio_signal = getaudiodata(recObj);  % Get recorded audio data

% Define low and high sampling rates for resampling
fs_low = 2000;   % Low sampling rate (causes aliasing)
fs_high = 10000; % High sampling rate (no aliasing)

% Resample audio signal
audio_low = resample(audio_signal, fs_low, fs_record);   % Low rate
audio_high = resample(audio_signal, fs_high, fs_record); % High rate

% Play the original, low rate, and high rate audio signals
disp('Playing original audio...');
sound(audio_signal, fs_record);  % Play original audio
pause(duration);  % Wait for original audio to finish

disp('Playing low rate audio (Aliasing)...');
sound(audio_low, fs_low);  % Play audio at low rate (with aliasing)
pause(duration);  % Wait for low rate audio to finish

disp('Playing high rate audio (No Aliasing)...');
sound(audio_high, fs_high);  % Play audio at high rate (no aliasing)
```

1b. **Quantization of 10 kHz signal with 3 bit ADC.**

```
fs = 500000; % Sampling frequency
f = 10e3; % Frequency of the signal
t = 0:1/fs:(2*1/f); % Time vector
x = sin(2*pi*f*t); % Sine wave signal
n_bits = 3; % Number of bits for quantization
L = 2^n_bits; % Number of quantization levels
x_max = max(abs(x)); % Maximum amplitude of the signal
step_size = 2*x_max/L; % Step size for quantization

% Quantization process
x_quantized = round(x/step_size)*step_size;

% Generate quantization levels for visualization
quant_levels = (-x_max:step_size:x_max);
% Levels of quantization
level_lines = repmat(quant_levels, length(t), 1);

% Plot the original and quantized signals with levels
figure;
hold on;
plot(t, x, 'b', 'LineWidth', 1.2); % Original signal
plot(t, x_quantized, 'r--', 'LineWidth', 1.2); % Quantized signal
for k = 1:length(quant_levels) % Plot quantization levels
    yline(quant_levels(k), 'k--', 'LineWidth', 0.8);
end
xlabel('Time (s)');
ylabel('Amplitude');
title('Quantization Levels and Signal');
legend('Original Signal', 'Quantized Signal', 'Quantization Levels');
grid on;
hold off;

quantization_error = x - x_quantized;
figure;
plot(t, quantization_error);
xlabel('Time (s)');
ylabel('Amplitude');
title('Quantization Error');
grid on
```

Expected Graph



VIVA QUESTIONS

1) What is Nyquist rate?
2) State sampling theorem.
3) What do you understand by the term 'Aliasing'
4) What is Quantization?

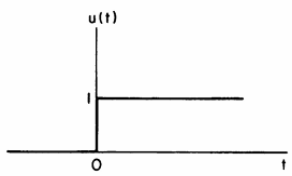
Experiment 2. Generation of the standard test signals & verification of the properties and operations

Aim: To generate various standard test signals and Verification of the signal properties like, time shift, time folding and amplitude scaling.

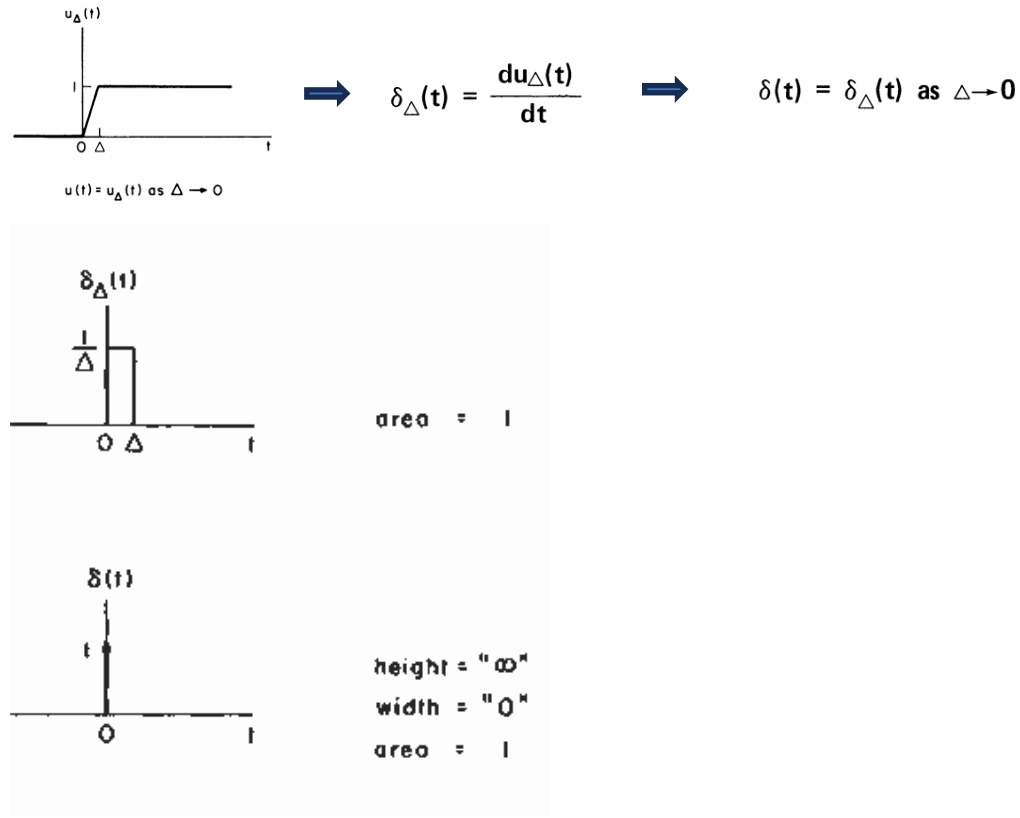
Theory:

Generation of Various Continuous time signals

i. Unit Step Signal: $u(t) = 1; \quad for\ t > 0;$
$= 0\ ; \quad for\ t < 0;$



ii. Unit Impulse Signal : $\delta(t)\ has\ unit\ area\ at\ t = 0$



iii. Comple Exponential Signals:

$$x(t) = Ke^{at} \; ; \qquad (1)$$

a) Real Exponential: K and a are real numbers in equation (1)

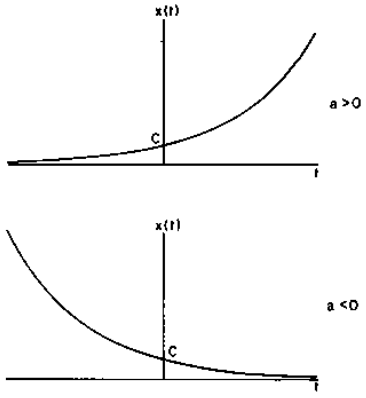
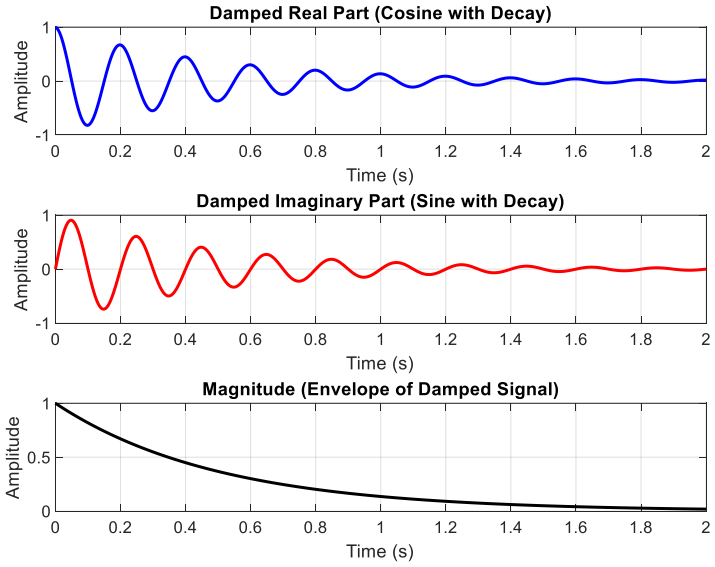
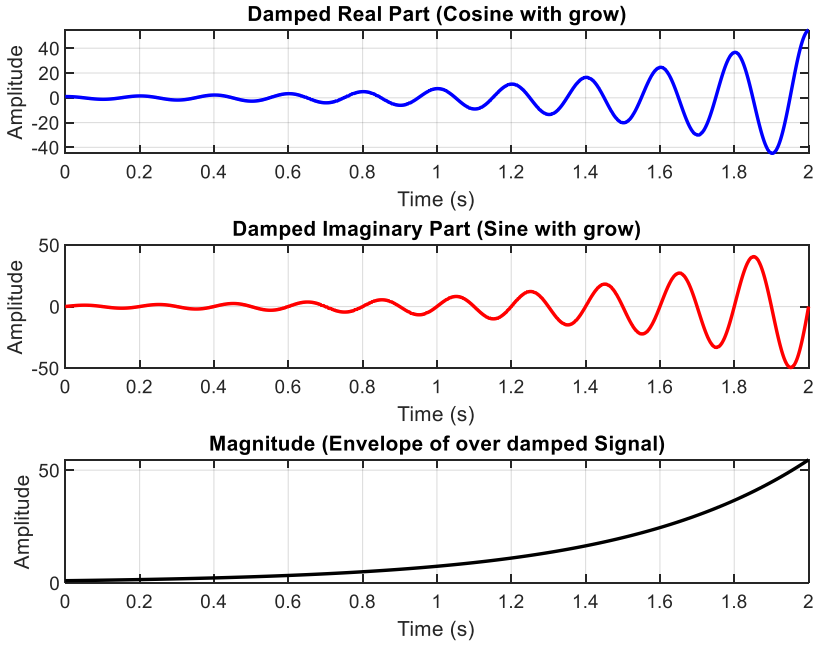


Figure 1:

b) If **a** is complex number

$$\mathbf{x(t) = Ke^{(\alpha \mp j\omega)t} = \; Ke^{(\alpha t)}e^{(\alpha t)}e^{(\mp j\omega)t} = Ke^{(\alpha t)}\{\cos\omega t \mp j\sin\omega t\}}$$




Damped Real Part (Cosine with grow)
Amplitude
40
20
0
-20
-40
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
Time (s)
Damped Imaginary Part (Sine with grow)
Amplitude
50
0
-50
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
Time (s)
Magnitude (Envelope of over damped Signal)
Amplitude
50
0
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2
Time (s)

```
clc;  clear all;  close all;

%generation of unit impulse signal
t1=-1:0.01:1;
syms delta(ta);
delta(ta)=(t1==0);
subplot(2,2,1);
plot(t1,delta(ta));
xlabel('time');
ylabel('amplitude');
title('unit impulse signal');
%generation of impulse sequence
subplot(2,2,2);
stem(t1,delta(ta));
xlabel('n');
ylabel('amplitude');
title('unit impulse sequence');



%generation of unit step signal
t2=-10:0.01:10;
syms u(ta);
u(ta)=(t2>=0);
subplot(2,2,3);
plot(t2,u(ta));
xlabel('time');
ylabel('amplitude');
title('unit step signal');
%generation of unit step sequence
subplot(2,2,4);
stem(t2,u(ta));
xlabel('n');
ylabel('amplitude');
title('unit step sequence');

f = 50; % Frequency in Hz
square_wave = square(2*pi*f*t);
figure;
plot(t, square_wave);
xlabel('Time (s)');
ylabel('Amplitude');
title('Square Wave');
```

```
grid on;

f = 50; % Frequency in Hz
sawtooth_wave = sawtooth(2*pi*f*t);
figure;
plot(t, sawtooth_wave);
xlabel('Time (s)');
ylabel('Amplitude');
title('Sawtooth Wave');
grid on;


% generating two input signals
t=0:.0001:0.1;
x1=sin(2*pi*50*t);
x2=sin(2*pi*100*t);
subplot(2,2,1);
plot(t,x1);

xlabel('time');
ylabel('amplitude');
title('input signal 1');
subplot(2,2,2);
plot(t,x2);
xlabel('time');
ylabel('amplitude');
title('input signal 2');
% addition of signals
y1=x1+x2;
subplot(2,2,3);
plot(t,y1);
xlabel('time');
ylabel('amplitude');
title('addition of two signals');
% multiplication of signals

y2=x1.*x2;
subplot(2,2,4);
plot(t,y2);
xlabel('time');
ylabel('amplitude');
title('multiplication of two signals');
```

% Operation on Independent variable for unit step signal

```
figure;
t2=-10:0.01:10;
syms u(ta);
u(ta)=(t2>=0);
subplot(3,1,1);
plot(t2,u(ta));
%plot(t,x1);
ylim([-2 2]);
xlabel('time t');
ylabel('amplitude');
title('input signal');
u(ta)=(t2>=2);
subplot(3,1,2);
plot(t2, u(ta));
ylim([-2 2]);
xlabel('time ');
ylabel('amplitude');
title('right shifted signal');
subplot(3,1,3);
u(ta)=(t2>=-2);
subplot(3,1,3);
plot(t2, u(ta));
ylim([-2 2]);
xlabel('time');
ylabel('amplitude');
title('left shifted signal');
```

% scaling of a signal1

```
figure;
A=2;
t2=-10:0.01:10;
u(ta)=(t2>=0);
y3=A* u(ta);
subplot(2,2,1);
plot(t2, u(ta));
xlabel('time');
ylabel('amplitude');
title('input signal')
```

```
subplot(2,2,2);
plot(t2,y3);
xlabel('time');
ylabel('amplitude');
title('amplified input signal');

% folding of a signal1
h=length(u(ta));
nx=0:h-1;
subplot(2,2,3);
plot(nx, y3);
xlabel('nx');
ylabel('amplitude');
title('input signal')
y4=fliplr(y3);
nf=-fliplr(nx);
subplot(2,2,4);
plot(nf,y4);
xlabel('nf');
ylabel('amplitude');
title('folded signal');



% Generation of Complex exponential signals
% 1). Real exponential
f = 50;              % Frequency in Hz
omega = 0;   % Angular frequency in rad/s
alpha = 2;           % Damping factor 2 for decaying and -2 for growing
t = 0:0.0001:2;       % Time vector (0 to 2 seconds, 1 ms resolution)
x = exp(-alpha * t) .* exp(1j * omega * t);


% Parameters
f = 50;              % Frequency in Hz
omega = 2 * pi * f;   % Angular frequency in rad/s
alpha = -2;           % Damping factor
t = 0:0.0001:2;       % Time vector (0 to 2 seconds, 1 ms resolution)
figure;
% Real part
subplot(4, 1, 1);
plot(t, real_part, 'b', 'LineWidth', 1.5);
title('Damped Real Part (Cosine with Decay');
```

```
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

% Damped complex exponential signal
x = exp(-alpha * t) .* exp(1j * omega * t);

% Extract real, imaginary parts and magnitude
real_part = real(x);          % Cosine component with damping
imag_part = imag(x);          % Sine component with damping
magnitude = abs(x);           % Magnitude (envelope)

% Plotting the damped complex exponential signal
figure;
% Real part
subplot(4, 1, 2);
plot(t, real_part, 'b', 'LineWidth', 1.5);
title('Damped Real Part (Cosine with Decay');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

% Imaginary part
subplot(4, 1, 3);
plot(t, imag_part, 'r', 'LineWidth', 1.5);
title('Damped Imaginary Part (Sine with Decay)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

% Magnitude (Envelope of the signal)
subplot(4, 1, 4);
plot(t, magnitude, 'k', 'LineWidth', 1.5);
title('Magnitude (Envelope of damped Signal)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```

Task:

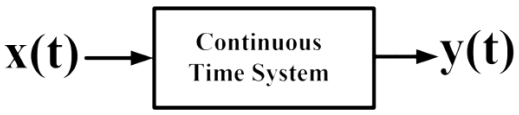
1). Write a program to generate unit impulse, unit step, complex exponential discrete time

Signals.

2. Write a program to verify the operations of independent variable for discrete time unit step signal

Experiment 3: Verification of the system responses and System Properties.

Aim: Analyze the response of RC circuit for various standard test inputs and verify the linearity, time invariant and casualty of RC systems.



Mathematical Model of RC circuit:

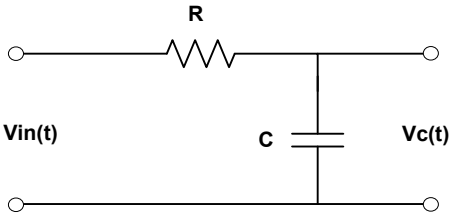


Figure 1:

Transfer Function: (Students are expected to derive the transfer function for the circuit manually)

$$H(s) = \frac{V_c(s)}{V_{in}(s)} = \frac{\frac{1}{RC}}{\left(S+\frac{1}{RC}\right)} \qquad (1)$$

Case 1: Impulse response

Vin(t)= Impulse signal

Therefore Vin(s)=1

Therefore equation (1) can be written as

$$V_c(s) = \frac{\frac{1}{RC}}{\left(S+\frac{1}{RC}\right)}$$

Therefore the response of system $V_c(t) = \frac{1}{RC}e^{-t/RC} = \frac{1}{RC}e^{-t/\tau}$ where $\tau = RC$ is the time constant

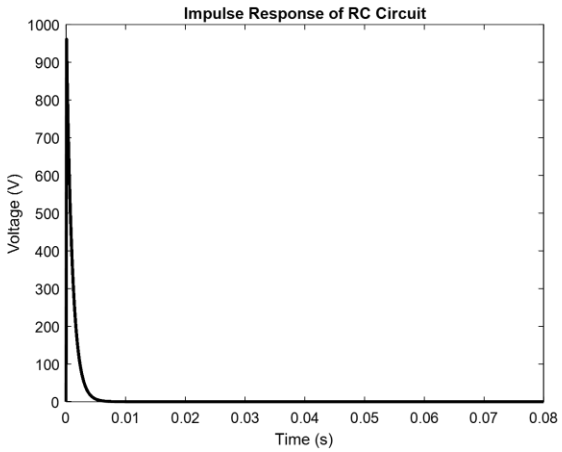


Figure 2

Case 2: Step Response of RC system:

Vin(t)= Unit Step Signal; Therefore Vin(s)=1/s

Therefore equation (1) can be written as

$$V_c(s) = \frac{\frac{1}{RC}}{S\left(S + \frac{1}{RC}\right)}$$

Taking inverse Laplace transform of $V_c(s)$ is $V_c(t) = \left(1 - e^{-\frac{t}{RC}}\right) = \left(1 - e^{-\frac{t}{\tau}}\right)$

Expected Graph:

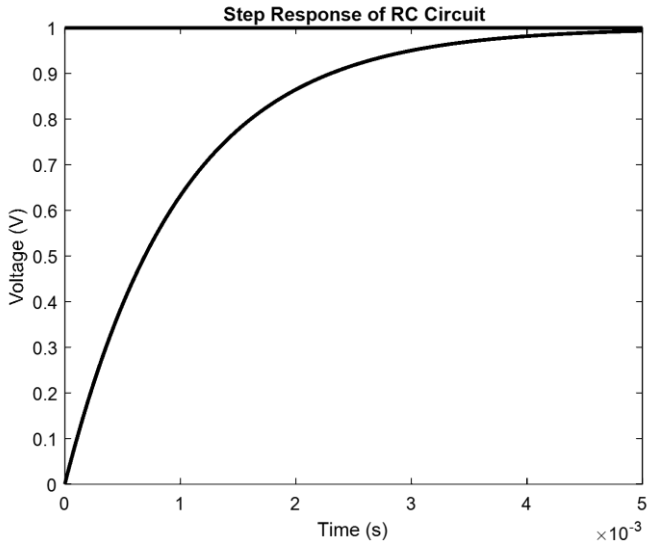


Figure 3

Task: 1). Obtain the Reponses for the sinusoidal input

2) Obtain the responses for Discrete time signals

Impulse response:

```
clc;
clear all;
close all;
% Circuit parameters
R = 1e3;  % Resistance in ohms
C = 1e-6; % Capacitance in farads
% Time vector
tau=R*C;% time constant 1msec
% Unit impulse
t=linspace(0,5*tau,1000);
dt=t(2)-t(1);
Vin=  zeros(size(t))
Vin(1)=1/dt;
% RC system transfer function
s = tf('s');
H = (1/(R*C)) / (1/(R*C) + s );
% Compute impulse response
Vcout = lsim(H, Vin,t);
% Plot results
figure;
plot(t, Vin, 'r', 'LineWidth', 2);
hold on;
plot(t, Vcout, 'b', 'LineWidth', 2);
grid on;
xlabel('Time (s)');
ylabel('Voltage (V)');
title(' Impulse Response of RC Circuit');
%legend('Input: delta(t)', 'Output: V_{out}(t)');
```

Step response:

```
clc;
clear all;
close all;

% Circuit parameters
R = 1e3;  % Resistance in ohms
C = 1e-6; % Capacitance in farads
% Time vector
tau=R*C;% time constant 1msec
% Unit Step
t=linspace(0,5*tau,1000);
Vin=  ones(size(t))
% RC system transfer function
s = tf('s');
H = (1/(R*C)) / (1/(R*C) + s );
% Compute step response
Vcout = lsim(H, Vin,t);
% Plot results
figure;
plot(t, Vin, 'r', 'LineWidth', 2);
hold on;
plot(t, Vcout, 'b', 'LineWidth', 2);
grid on;
xlabel('Time (s)');
ylabel('Voltage (V)');
title(' Step Response of RC Circuit');
legend('Input: u(t)', 'Output: V_{out}(t)');
```

Verification of System Properties

Aim: To verify the linearity, time invariant and casualty property of the system

Theory:

1. Linearity:
   To prove the linearity property of an RC circuit, we need to verify the principles of superposition and homogeneity. A system is linear if it satisfies the following conditions:
   1. Additivity (Superposition Property): If the input is the sum of two signals, the output must be the sum of the outputs for each individual input.
   $x_1(t) + x_2(t) \Longrightarrow y_1(t) + y_2(t)$
   2. Homogeneity (Scaling Property): If the input is scaled by a constant factor, the output must also be scaled by the same factor.
   a x(t) $\Longrightarrow$ a y(t)

Consider the RC Circuit
A simple RC circuit consists of a resistor (R) and a capacitor (C) in series. The voltage across the capacitor $V_c(t)$ is governed by the first-order linear differential equation:

$$RC\frac{dV_c(t)}{dt} + V_c(t) = V_{in}(t)$$

where:
-$V_c(t)$ is the capacitor voltage (output),
- $V_{in}(t)$ is the input voltage,
- R is the resistance,
- C is the capacitance.

- Check Additivity (Superposition Property)

If we apply two different inputs, $V_{in1}(t)$ and $V_{in2}(t)$the corresponding outputs will be $V_{c1}(t)$ and $V_{c2}(t)$.

Since the system follows a linear differential equation, applying V_in1(t) + V_in2(t) as the input results in:

$$RC\frac{d\{V_{c1}(t)+V_{c2}(t)}{dt} + V_{c1}(t) + V_{c2}(t) = V_{in1}(t) + V_{in2}(t)$$

This equation confirms that the total response is the sum of the individual responses, proving additivity.

- Check Homogeneity (Scaling Property)

If we apply a scaled input a $V_{in}(t)$, the output equation becomes:

$$\left\{RC\frac{daV_c(t)}{dt} + aV_c(t)\right\} = aV_{in}(t)$$

which simplifies to:

$$a\left\{RC\,\frac{dV_c(t)}{dt} + V_c(t)\right\} = aV_{in}(t)$$

This confirms that the output is also scaled by the same factor a, proving homogeneity.

2. Time invariant:

A system is time-invariant if a time shift in the input results in the same time shift in the output. Mathematically, if an input signal x(t) produces an output y(t), then for any time shift t_0, the system is time-invariant if:

If x(t) → y(t), then x(t - t_0) → y(t - t_0)

The differential equation governing the voltage across the capacitor in an RC circuit is:

$$RC\,\frac{dV_c(t)}{dt} + V_c(t) = V_{in}(t)$$

where:

- $V_c(t)$ is the capacitor voltage (output),

- $V_{in}(t)$ is the input voltage,

- R is the resistance,

- C is the capacitance.

2. Apply a Time-Shifted Input
Let's apply a shifted input Vin(t - t_0) to the system. The new differential equation becomes:

$$RC\,\frac{dV_c(t-t_0)}{dt} + V_c(t-t_0) = V_{in}(t-t_0)$$

Comparing this with the original equation, we observe that the form of the equation remains unchanged. This means that the output also shifts by t_0, i.e.,

$V_c(t-t_0)$is the output for input $V_{in}(t-t_0)$.

Program:

```
clc;
clear all;
close all;
% Circuit parameters
R = 1e3;  % Resistance in ohms
C = 1e-6; % Capacitance in farads
tau = R * C; % Time constant
% Time vector
```

```
t = 0:1e-4:0.1; % 0 to 0.1 sec with 0.1 ms steps
% Input signals
Vin1 = cos(2 * pi * 50 * t);  % Sinusoidal input 1 (50 Hz) %x1(t)
Vin2 = cos(2 * pi * 100 * t); % Sinusoidal input 2 (100 Hz)%x2(t)
Vin_sum = Vin1 + Vin2;        % Superposition input%x(t)
s = tf('s');
H = 1 / (1 + s * R * C);
% Compute outputs using lsim (Laplace response to input)
Vout1 = lsim(H, Vin1, t); %y1(t)
Vout2 = lsim(H, Vin2, t);%y2(t)
Vout_sum = lsim(H, Vin_sum, t);%y(t)
% Verify Superposition Property
Vout_check = Vout1 + Vout2;
% Homogeneity Test
k = 2;
Vin_scaled = k * Vin1;
Vout_scaled = lsim(H, Vin_scaled, t);
Vout_homogeneity_check = k * Vout1;
% Plot Results
figure;
% Superposition Test
subplot(3,1,1);
plot(t, Vout_sum, 'b', 'LineWidth', 2);
hold on;
plot(t, Vout_check, 'r--', 'LineWidth', 2);
grid on;
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Superposition Test: Vout1 + Vout2 vs. Output of (Vin1 + Vin2)');
legend('Vout(Vin1 + Vin2)', 'Vout1 + Vout2');
% Homogeneity Test
subplot(3,1,2);
plot(t, Vout_scaled, 'b', 'LineWidth', 2);
hold on;
plot(t, Vout1, 'r--', 'LineWidth', 2);
grid on;
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Homogeneity Test: k * Vout(Vin) vs. Vout(k * Vin)');
legend('Vout(k * Vin)', 'k * Vout(Vin)');
% Time invariant

% Circuit parameters
R = 1e3;  % Resistance in ohms
```

```
C = 1e-6; % Capacitance in farads
% Time vector
t = 0:1e-4:0.1; % 0 to 0.1 sec
% Input signal: Sinusoidal wave
Vin = sin(2 * pi * 50 * t);
% Delayed input signal
T = 0.01; % Time shift
Vin_shifted = sin(2 * pi * 50 * (t - T));
% RC system transfer function
s = tf('s');
H = 1 / (1 + s * R * C);
% Compute outputs
Vout = lsim(H, Vin, t);
Vout_shifted = lsim(H, Vin_shifted, t);
% Plot results
figure;
subplot(2,1,1);
plot(t, Vin, 'r', 'LineWidth', 2);
hold on;
plot(t, Vin_shifted, 'b--', 'LineWidth', 2);
grid on;
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Input Signal and Shifted Input');
legend('Original Vin(t)', 'Shifted Vin(t - T)');
subplot(2,1,2);
plot(t, Vout, 'r', 'LineWidth', 2);
hold on;
plot(t, Vout_shifted, 'b--', 'LineWidth', 2);
grid on;
xlabel('Time (s)');
ylabel('Voltage (V)');
title('Output Signal and Shifted Output');
legend('Original Vout(t)', 'Shifted Vout(t - T)');
```

Task:

1. Verify the Causal and Memory for RC system .

Experiment 4:

Solution of difference equation

AIM: To perform solution of difference equation using iterative approach for various inputs

Theory:

A linear constant-coefficient difference equation **has the general form:**

$$\boldsymbol{a_0 y[n] + a_1 y[n-1] + a_2 y[n-2] + \cdots a_n y[n-N] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + \cdots b_n x[n-M]} \quad \textbf{(1)}$$

equation (1) can be written as

$$\boldsymbol{\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]} \quad (2)$$

The solution y[n] is summation of particular solution to equation (2) and homogenous solution

$$\boldsymbol{\sum_{k=0}^{N} a_k y[n-k] = 0} \quad (3)$$

The iterative method of solving difference equation solution can be written as

$$y[n] = \frac{1}{a_0}\{\boldsymbol{\sum_{k=0}^{M} b_k x[n-k] - \sum_{k=1}^{N} a_k y[n-k]}\} \quad (4)$$

Problem Definition: For a given difference equation find

i. Homogenous solution
ii. Impulse response
iii. Step response

$$\boldsymbol{y[n] + \frac{1}{2} y[n-1] + \frac{1}{5} y[n-2] = x[n]}$$

$$\boldsymbol{y[n] = -\frac{1}{2} y[n-1] - \frac{1}{5} y[n-2] + x[n]}$$

Let initial conditions y[-1]=1 and y[-2]= 0

Program:

```
% Define coefficients  as per equation (4)
b = input('Enter the coefficient of input terms b=');        % Numerator b0=1
a = input('Enter the coefficient of output terms a=') ; % Denominator a0=1, a1=0.5 and a2=0.2
% Case 1;
% Define input signal (Homogenous solution)
n=input('Enter the range of data indices required ')% Number  of sequences required for the
computation

x =  zeros(1, length(n)-1);   % x(1)=1 for case 2
% Define initial conditions
y_init = input("Enter the initial conditions for output")  ; % y[-1] = 1, y[-2] = 0

% Compute initial conditions for filter
zi = filtic(b, a, y_init);

% Solve using filter
y = filter(b, a, x, zi);

% Plot
stem(n, y, 'filled'); grid on;
xlabel('n'); ylabel('y[n]'); title('Difference Equation Solution with Initial Conditions');
```

Result:

Enter the coefficients of input terms b= [1]

Enter the coefficients of output terms a= [1 0.5 0.2]

Enter the range of data indices require 0:10;

Output:

y[n]=[ ]

plot of y[n]:

Program:

Case 2:

```
% Define coefficients  as per equation (4)
b = input('Enter the coefficient of input terms b=');        % Numerator b0=1
a = input('Enter the coefficient of output terms a=') ; % Denominator a0=1, a1=0.5 and a2=0.2
% Case 1;
% Define input signal (Impuse reponse)
n=input('Enter the range of data indices required ')% Number  of sequences required for the
computation

x =  zeros(1, length(n)-1);
x(1)=1 % which modifies vector x to impulse signal
% Define initial conditions
y_init = input("Enter the initial conditions for output")  ; % y[-1] = 1, y[-2] = 0

% Compute initial conditions for filter
zi = filtic(b, a, y_init);

% Solve using filter
y = filter(b, a, x, zi);

% Plot
stem(n, y, 'filled'); grid on;
xlabel('n'); ylabel('y[n]'); title('Difference Equation Solution with Initial Conditions');
```

Result:

Enter the coefficients of input terms b= [1]

Enter the coefficients of output terms a= [1 0.5 0.2]

Enter the range of data indices require 0:10;

Output:

y[n]=[ ]

plot of y[n]=

Case 3:

```
% Define coefficients  as per equation (4)
b = input('Enter the coefficient of input terms b=');        % Numerator b0=1
a = input('Enter the coefficient of output terms a=') ; % Denominator a0=1, a1=0.5 and a2=0.2
% Case 1;
% Define input signal (Step resonse)
n=input('Enter the range of data indices required ')% Number  of sequences required for the
computation

x =  ones(1, length(n)-1);
% Define initial conditions
y_init = input("Enter the initial conditions for output")  ; % y[-1] = 1, y[-2] = 0

% Compute initial conditions for filter
zi = filtic(b, a, y_init);

% Solve using filter
y = filter(b, a, x, zi);

% Plot
stem(n, y, 'filled'); grid on;
xlabel('n'); ylabel('y[n]'); title('Difference Equation Solution with Initial Conditions');
```

Result:

Enter the coefficients of input terms b= [1]

Enter the coefficients of output terms a= [1 0.5 0.2]

Enter the range of data indices require 0:10;

Output:

y[n]=[ ]

plot of y[n]:

Data index **n**	Case 1: x[n]=0 y[n]	Case 2: $x[n] = \delta[n]$ y[n]	Case 3: $x[n] = u[n]$ y[n]
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			

Task: 1). Find step response of RC circuit. R=1kΩ and C=1uF take sampling period of 1msecond.

Experiment 4b:

TO PERFORM LINEAR CONVOLUTION OF GIVEN SEQUENCES

AIM: To perform linear convolution of given two casual sequences

Theory:

Program:

```
% Convolution Sum for Any Discrete Sequences
clc; clear; close all;

% Define first sequence x[n] and its indices
x = input('Enter sequence x[n]: ');
nx = input('Enter time indices of x[n] (e.g., -3:3): ');

% Define second sequence h[n] and its indices
h = input('Enter sequence h[n]: ');
nh = input('Enter time indices of h[n] (e.g., -2:2): ');

% Compute convolution
y = conv(x, h);

% Compute the time indices for the convolution result
n_min = nx(1) + nh(1);
n_max = nx(end) + nh(end);
ny = n_min:n_max;  % Time indices for y[n]

% Plot input sequences
figure;

subplot(3,1,1);
stem(nx, x, 'filled', 'r'); grid on;
xlabel('n'); ylabel('x[n]'); title('Input Sequence x[n]');

subplot(3,1,2);
stem(nh, h, 'filled', 'b'); grid on;
xlabel('n'); ylabel('h[n]'); title('Input Sequence h[n]');

% Plot Convolution Result
subplot(3,1,3);
stem(ny, y, 'filled', 'k'); grid on;
xlabel('n'); ylabel('y[n]'); title('Convolution Result y[n] = x[n] * h[n]');

% Display the computed convolution result
disp('Convolution result y[n]: ');
disp(y);
disp('Corresponding time indices: ');
disp(ny);
```

Result:

Input

x[n] = [1 2 1 1]

Enter the indices of x[n]= -2:1;

Input h[n]= [1 1 1 1]

Enter the indices of h[n]= -1:2;

Output

Y[n]=[1 3 4 5 4 2 1]

Output indices : -3:3;

Task:

1). Perform Convolution sum for step response RC circuit. R=1kΩ and C=1uF.

- Consider $h[n] = \alpha^n$ and $x[n] = u[n]$ for n=20 point.
- Where Discrete-time decay factor $\alpha = e^{-T/RC}$, T= sampling period as 1msec

Experiment 5: **Fourier series Analysis for standard signals**

Aim: To synthesize the standard periodic test signals using sum of harmonically related sinusoids

Theory:

Synthesis equation :

$$x(t) = \sum_{k=-\infty}^{\infty} a_k e^{jk\omega_0 t} = \sum_{k=-\infty}^{\infty} a_k e^{jk(2\pi/T)t}$$

Where T is fundamental period and ω_0 is fundamental frequency of the signal x(t).

a_k is the Fourier coefficients.

Analysis Equation:

$$a_k = \frac{1}{T}\int_T x(t)\, e^{-jk(2\pi/T)t}\ dt$$

Example:

Define a square wave signal $x(t) = A \quad for \quad 0 \leq t < \frac{T}{2}$;

$$= -A \; for \quad \frac{T}{2} \leq t < T$$

It is periodic for the period $T = \frac{1}{f_0}$ and $\omega_0 = \frac{2\pi}{T}$

Fourier coefficient $\mathbf{a_k = \frac{2A}{jk\pi} \; for \; k = odd}$

= 0 for n =even

Use synthesis equation for $-10 \leq k \leq 10$ find the approximation

$$x(t) = \sum_{k=-10}^{10} \frac{2A}{jk\pi} e^{jk\omega_0 t}$$

Program 1: To compute the Fourier Coefficients for $-10 \leq k \leq 10$

```
clc; clear; close all;
% Square wave parameters
A = 1;              % Amplitude
T = 2*pi;           % Period
w0 = 2*pi/T;        % Fundamental angular frequency  is 1 rad /sec
N = 10;             % Number of harmonics to compute

% Define the square wave function over one period

x = @(t) A * (t < T/2) - A * (t >= T/2);


% @(t) indicates that 't' is the independent variable for function 'x'
%(t < T/2)  evaluated as true value =1 for all the values of 'x' for t < T/2  and zero otherwise
% Compute Fourier Coefficients ak
ak = zeros(1, 2*N+1);   % Store coefficients for n = -N to N
k_values = -N:N;        % Range of harmonics

for index = 1:length(k_values)
   k = k_values(index);
```

```
    % Fourier coefficient integral calculation
    ak(index) = (1/T) * integral(@(t) x(t) .* exp(-1j * k * w0 * t), 0, T);
end

% Plot Fourier Coefficients

figure;
subplot(2,1,1);
stem(k_values, abs(ak), 'r', 'LineWidth', 1.5);
xlabel('Harmonic Index n'); ylabel('|a_k|');
title('Magnitude Spectrum of Fourier Coefficients');
grid on;

subplot(2,1,2);
stem(k_values, angle(ak), 'b', 'LineWidth', 1.5);
xlabel('Harmonic Index n'); ylabel('Phase of a_k (radians)');
title('Phase Spectrum of Fourier Coefficients');
grid on;
```

Theoretical Calculation: for N=10 and to be compared with the plot obtained after simulation

K	Magnitude $\mathbf{\lvert a_k \rvert = \frac{\lvert 2A \rvert}{\lvert jk\pi \rvert} = \frac{2A}{k\pi}}$	$\mathbf{angle(a_k) = \tan^{-1}\left(\frac{0}{2A}\right) - \tan^{-1}(k\pi/0)}$
-10	0	-
-9	0.07	90
-8	0	0
-7	0.09	90
-6	0	0
-5	0.127	90
-4	0	0
-3	0.21	90
-2	0	0
-1	0.63	90
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Program 2: Synthesis of x(t) using the relevant Fourier coefficient and harmonically related components of its fundamental frequency ω_0 for $-10 \leq k \leq 10$

% Fourier Series Approximation of a Square Wave using Exponential Form

```
clc; clear; close all;
% Parameters
A = 1;              % Amplitude of the square wave
T = 2*pi;           % Period (arbitrary choice for visualization)
w0 = 2*pi/T;        % Fundamental angular frequency
t = linspace(-T, T, 1000); % Time vector
% Number of harmonics to include
N = 10;  % Increase this for a better approximation
% Construct the Fourier series
x_approx = zeros(size(t));
for k = -N:N
   if mod(k,2) ~= 0  % Only odd harmonics contribute,( reminder of k/2  is not zero)
      ak = (2*A) / (1j * k * pi); % Fourier coefficient
      x_approx = x_approx + ak * exp(1j * k * w0 * t);
     % plot(t, real(x_approx), 'r', 'LineWidth', 2); hold on;
    %  pause(1)
   end
end
% Plot the results
figure;
plot(t, real(x_approx), 'r', 'LineWidth', 2); hold on;
plot(t, A * sign(sin(w0 * t)), 'k--', 'LineWidth', 1.5);
% Ideal Square Wave
% sign function returns  1  is sin(w0 * t)>0 and it returns  -1 for sin(w0 * t)<0.
grid on;
xlabel('Time (t)');
ylabel('Amplitude');
title(['Exponential Fourier Series Approximation (N = ', num2str(N), ')']);
legend('Fourier Series Approximation', 'Ideal Square Wave');
ylim([-1.5 1.5]);
```

Experiment 6:

Frequency Response of RC -Low pass filter using Fourier Transform

Aim: To analyse the frequency response of RC low pass filter using Fourier Transform

Theory:

$$T_{rise} = \frac{2.1972}{\omega_o}$$

Cutoff frequency $\omega_o = \frac{1}{RC}\ rad/sec$

Program 1:

```
clc;
clear;
close all;

% Define circuit parameters
R = 1e3;  % Resistance in ohms
C = 1e-6; % Capacitance in farads
% Define frequency range
w = logspace(0, 5, 1000);  % Frequency in rad/s (1 Hz to 1 MHz)
% Compute Transfer Function H(jw)
Hjw = 1 ./ (1 + 1j * w * R * C);

% Magnitude and Phase Response
magnitude = abs(Hjw);
phase = angle(Hjw);

figure;
% Plot Magnitude Response (Linear Frequency Scale)
subplot(2,1,1);
plot(w, magnitude, 'b', 'LineWidth', 2);
grid on;
xlabel('Frequency (rad/sec)');
ylabel('Magnitude');
title('Magnitude Response of RC Circuit');
xlim([1 10^4])
```

```
% Plot Phase Response
subplot(2,1,2);
semilogx(w, phase * (180/pi), 'r', 'LineWidth', 2);
grid on;
xlabel('Frequency (rad/s)');
ylabel('Phase (degrees)');
title('Phase Response of RC Circuit');
%
figure;
subplot(2,1,1);
semilogx(w, 20*log10(magnitude), 'b', 'LineWidth', 2);
grid on;
xlabel('Frequency (rad/s)');
ylabel('Magnitude (dB)');
title('Magnitude Response of RC Circuit');

% Plot Phase Response
subplot(2,1,2);
semilogx(w, phase * (180/pi), 'r', 'LineWidth', 2);
grid on;
xlabel('Frequency (rad/s)');
ylabel('Phase (degrees)');
title('Phase Response of RC Circuit');
ylim([-40 5]);
fc = 1 / (2 * pi * R * C); % Cutoff frequency in Hz
wc=2*pi*fc;
disp(['Cutoff Frequency (Hz): ', num2str(wc)]);
```

$\boldsymbol{\omega}$	Magnitude $\mathbf{\lvert H(j\omega)\rvert = \frac{1}{\lvert 1 + j\frac{\omega}{\omega_0}\rvert}}$	$\mathbf{angle(H(j\omega) = -\tan^{-1}\left(\frac{\omega}{\omega_0}\right)}$
0	1	0
$\boldsymbol{\omega_0}$		
$\boldsymbol{2\omega_0}$		
$\boldsymbol{3\omega_0}$		
$\boldsymbol{4\omega_0}$		
$\boldsymbol{5\omega_0}$		
$\boldsymbol{6\omega_0}$		
$\boldsymbol{7\omega_0}$		
$\boldsymbol{8\omega_0}$		
$\boldsymbol{9\omega_0}$		
$\boldsymbol{10\omega_0}$		

Experiment 7: **Computation of DFT and FFT for given sequence**

Aim: Compute DFT and FFT, compare the computation requirements

Theory:

Program:

```
clc; clear; close all;

% Define the input sequence
x =input('Enter the input sequence of the length N')% [0 1 2  3 Example input signal
N = length(x);  % Number of points
%% Compute DFT using definition
tic;
X_dft = zeros(1, N);  % Preallocate for efficiency
for k = 0:N-1  % MATLAB indexing correction
   for n = 0:N-1
      X_dft(k+1) = X_dft(k+1) + x(n+1) * exp(-1j * 2 * pi * k * n / N);
   end
end
t_dft = toc;

%% Compute FFT using MATLAB built-in function
tic;
X_fft = fft(x);  % FFT computation
t_fft = toc;

%% Display computation times
fprintf('Time taken by DFT: %f seconds\n', t_dft);
fprintf('Time taken by FFT: %f seconds\n', t_fft);
fprintf('Speed-up factor (DFT/FFT): %.2f\n', t_dft / t_fft);

%% Ensure identical magnitude and phase for both DFT and FFT
% Magnitude should match directly
mag_dft = abs(X_dft);
mag_fft = abs(X_fft);

% Phase unwrapping to ensure continuity
phase_dft = unwrap(angle(X_dft));
phase_fft = unwrap(angle(X_fft));
```

```
%% Plot DFT results
figure;
subplot(2,1,1);
stem(0:N-1, mag_dft, 'b', 'Marker', 'o');
title('Magnitude Spectrum of DFT');
xlabel('Frequency index k');
ylabel('|X(k)|');
grid on;

subplot(2,1,2);
stem(0:N-1, phase_dft, 'r', 'Marker', 'o');
title('Unwrapped Phase Spectrum of DFT');
xlabel('Frequency index k');
ylabel('Phase(X(k)) [radians]');
grid on;

%% Plot FFT results
figure;
subplot(2,1,1);
stem(0:N-1, mag_fft, 'b', 'Marker', 'o');
title('Magnitude Spectrum of FFT');
xlabel('Frequency index k');
ylabel('|X(k)|');
grid on;

subplot(2,1,2);
stem(0:N-1, phase_fft, 'r', 'Marker', 'o');
title('Unwrapped Phase Spectrum of FFT');
xlabel('Frequency index k');
ylabel('Phase(X(k)) [radians]');
grid on;

%% Check if DFT and FFT are identical
if max(abs(mag_dft - mag_fft)) < 1e-10 && max(abs(phase_dft - phase_fft)) < 1e-10
   disp('DFT and FFT Magnitude & Phase are identical!');
else
   disp('There is a slight difference due to numerical precision.');
end
```

Result:

Input: x=[1 2 3 4];

Experiment 8: FFT on RC circuit to identify the frequency content in input and output

Aim: Apply FFT to RC circuit as low pass filter and identify the frequency information

Program:

```
% Parameters
R = 1e3;       % Resistance in Ohms
C = 1e-6;      % Capacitance in Farads
fc = 1 / (2 * pi * R * C);  % Cutoff frequency

fs = 100e3;    % Sampling frequency
T = 1;         % Signal duration (s)
n= 0:1/fs:T;  % Time vector( Discrete)

% Input Signal (Fundamental + Harmonics)
f0 = 50;      % Fundamental frequency (Hz)
x =sin(2 * pi * f0 * n) + 0.5 * sin(2 * pi * 3 * f0 * n) + 0.3 * sin(2 * pi * 5 * f0 * n);
% Composite signal with 3rd and 5th harmonics

% RC Circuit Response (Convolution with Impulse Response)
h = (1 / (R * C)) * exp(-n/ (R * C)); % Impulse response of RC
y = conv(x, h, 'same'); % Output of RC circuit

% FFT of Input and Output
N = length(n);% Length
X = fft(x, N);
Y = fft(y, N);
f = (0:N-1) * (fs / N); % Frequency axis

% Magnitude Spectrum
figure;
subplot(2,1,1);
plot(f, abs(X) / max(abs(X)));
title('Input Signal Spectrum');
xlabel('Frequency (Hz)');
ylabel('Magnitude (Normalized)');
xlim([0, 10e3]);
```

```
subplot(2,1,2);
plot(f, abs(Y) / max(abs(Y)));
title('Output Signal Spectrum (After RC Circuit)');
xlabel('Frequency (Hz)');
ylabel('Magnitude (Normalized)');
xlim([0, 10e3]);

% Display Cutoff Frequency
disp(['RC Circuit Cutoff Frequency: ', num2str(fc), ' Hz
```