DIGITAL SIGNAL PROCESSING

LAB PRACTITAL FILE (ICT391)

Submitted as part of the requirements for awarding the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATIONS ENGINEERING

SUBMITTED BY :

Ketan Kumar (03116412822)

B. Tech. (ECE), 5th Semester

SUBMITTED TO :

Dr. Chandra Shekhar Rai,

Prof. at USICT

Guru Gobind Singh
Indraprastha University

UNIVERSITY SCHOOL OF INFORMATION, COMMUNICATION AND TECHNOLOGY

SECTOR 16C, DWARKA, NEW DELHI 110078

INDEX

PRACTICAL No. 1

AIM

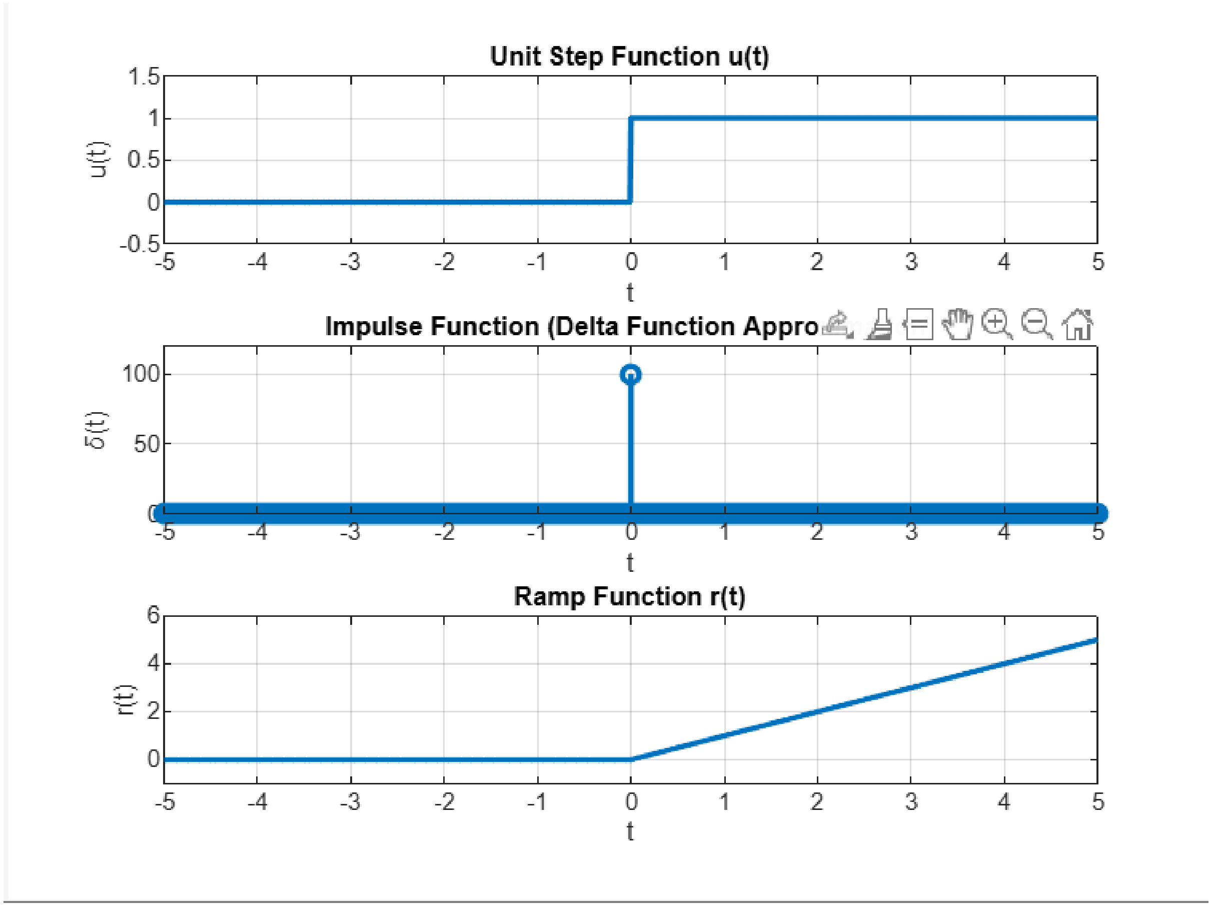
To plot various functions using matlab.

PROGRAM

```
% Define the time vector
t = -5:0.01:5;  % Time from -5 to 5 with a step size of 0.01

% Unit Step Function (u(t))
u = double(t >= 0);  % u(t) = 0 for t < 0 and u(t) = 1 for t >= 0

% Impulse Function (delta(t)) - Approximation using a very high value at t =
0
delta = zeros(size(t));
delta(t == 0) = 100;  % Approximate Dirac delta function by setting a large
value at t = 0

% Ramp Function (r(t))
r = max(0, t);  % r(t) = 0 for t < 0 and r(t) = t for t >= 0

% Plotting all functions in one figure
figure;

% Subplot for Unit Step Function
subplot(3, 1, 1);  % 3 rows, 1 column, first plot
plot(t, u, 'LineWidth', 2);
title('Unit Step Function u(t)');
xlabel('t');
ylabel('u(t)');
axis([-5 5 -0.5 1.5]);
grid on;

% Subplot for Impulse Function (Delta Function Approximation)
subplot(3, 1, 2);  % 3 rows, 1 column, second plot
stem(t, delta, 'LineWidth', 2);  % Use stem for discrete impulse
approximation
title('Impulse Function (Delta Function Approximation)');
xlabel('t');
ylabel('\delta(t)');
axis([-5 5 0 120]);
```

```
grid on;

% Subplot for Ramp Function
subplot(3, 1, 3);  % 3 rows, 1 column, third plot
plot(t, r, 'LineWidth', 2);
title('Ramp Function r(t)');
xlabel('t');
ylabel('r(t)');
axis([-5 5 -1 6]);
grid on;
```

OUTPUT



PRACTICAL No. 2

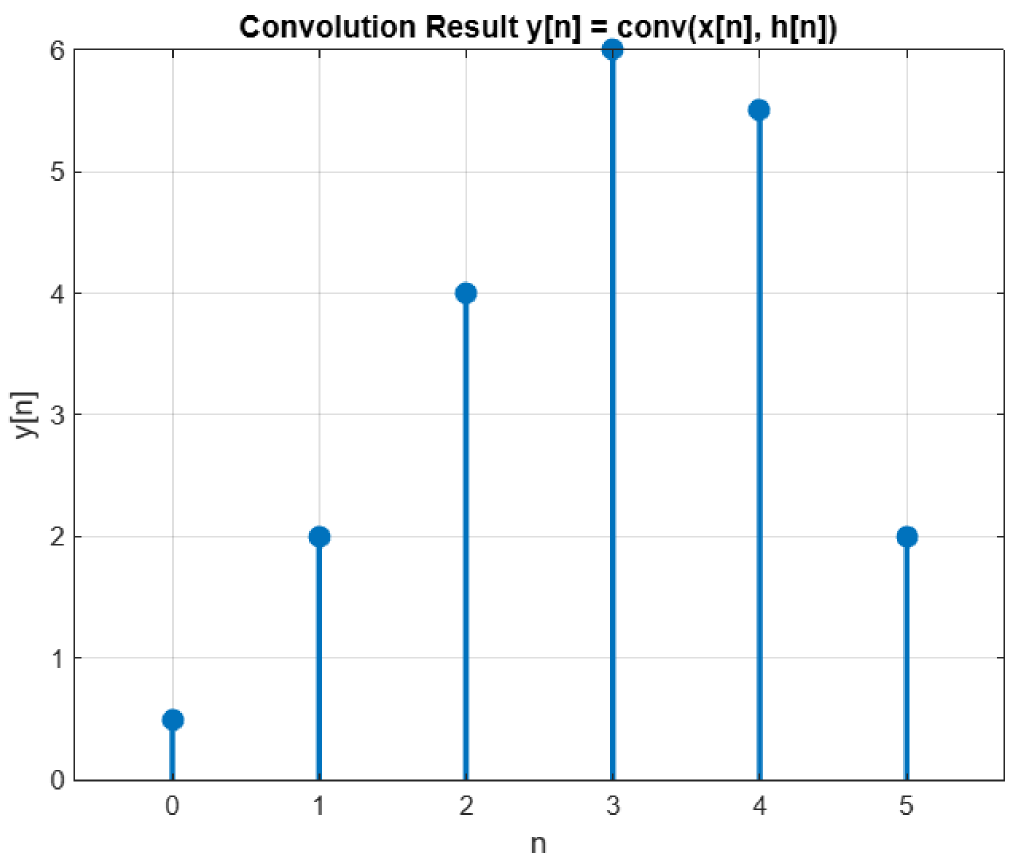
AIM

To implement convolution with and without matlab function.

PROGRAM

```
% Define two signals (example: x and h)
x = [1, 2, 3, 4];     % Example signal x[n]
h = [0.5, 1, 0.5];    % Example signal h[n]

% Perform the convolution using MATLAB's conv function
y = conv(x, h);

% Plot the results
figure;
subplot(2,1,1);
stem(0:length(x)-1, x, 'filled', 'LineWidth', 2);
title('Signal x[n]');
xlabel('n');
ylabel('x[n]');
grid on;

subplot(2,1,2);
stem(0:length(h)-1, h, 'filled', 'LineWidth', 2);
title('Signal h[n]');
xlabel('n');
ylabel('h[n]');
grid on;

% Plot the convolution result
figure;
stem(0:length(y)-1, y, 'filled', 'LineWidth', 2);
title('Convolution Result y[n] = conv(x[n], h[n])');
xlabel('n');
ylabel('y[n]');
grid on;
```

OUTPUT



PRACTICAL No. 3

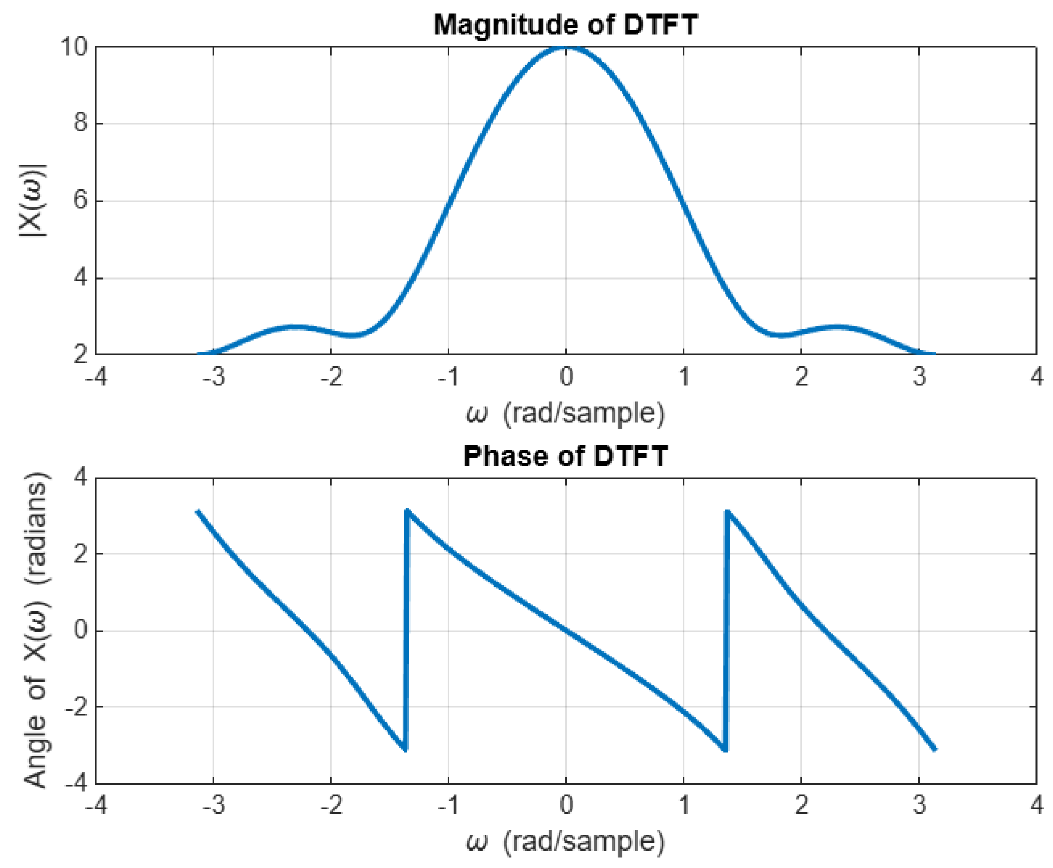
AIM

To find DTFT of given sequence.

PROGRAM

```
% Define the sequence x[n] (replace with your sequence of numbers)
x = [1, 2, 3, 4];  % Example sequence

% Define the frequency range for DTFT (from -pi to pi)
omega = linspace(-pi, pi, 500);  % 500 points between -pi and pi

% Initialize the DTFT result
X_omega = zeros(size(omega));

% Compute the DTFT using the summation formula
for k = 1:length(omega)
    X_omega(k) = sum(x .* exp(-1j * omega(k) * (0:length(x)-1)));  % DTFT
formula
end

% Plot the magnitude and phase of the DTFT
figure;

% Magnitude plot
subplot(2,1,1);
plot(omega, abs(X_omega), 'LineWidth', 2);
title('Magnitude of DTFT');
xlabel('\omega (rad/sample)');
ylabel('|X(\omega)|');
grid on;

% Phase plot
subplot(2,1,2);
plot(omega, angle(X_omega), 'LineWidth', 2);
title('Phase of DTFT');
xlabel('\omega (rad/sample)');
ylabel('Angle of X(\omega) (radians)');
grid on;
```

OUTPUT



Magnitude of DTFT
|X(ω)|
ω (rad/sample)
Phase of DTFT
Angle of X(ω) (radians)
ω (rad/sample)

PRACTICAL No. 4

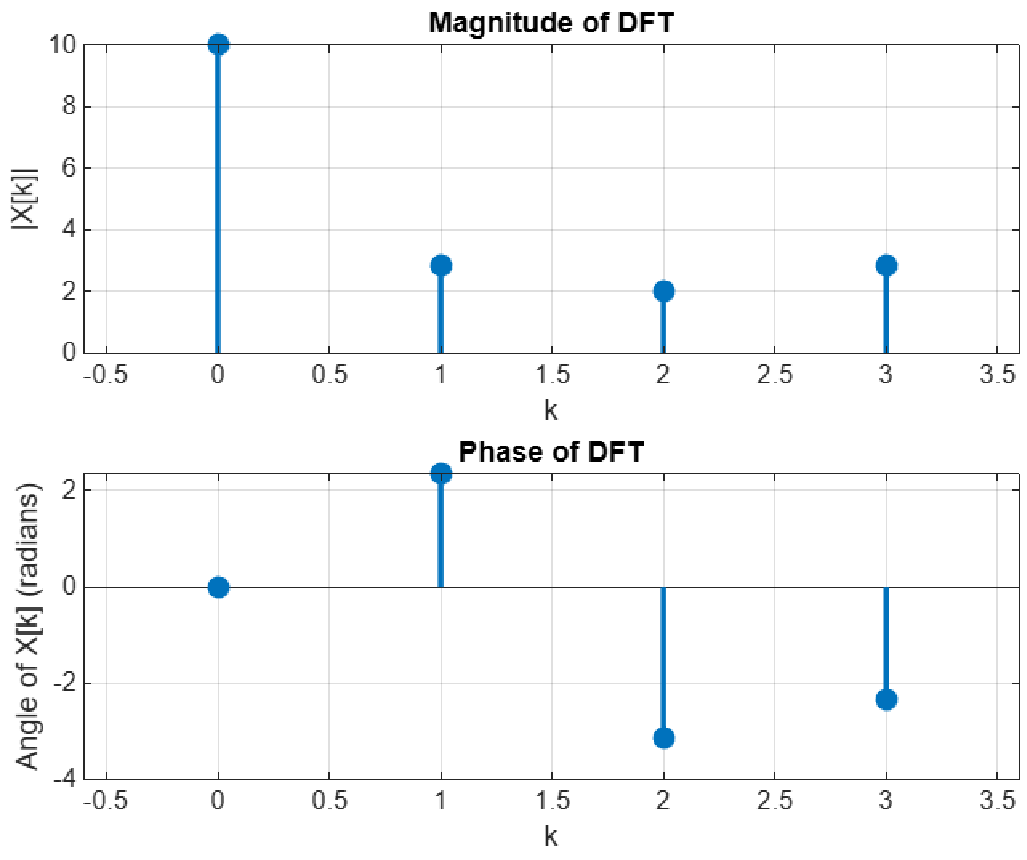
AIM

To implement DFT of given sequence.

PROGRAM

```
% Define the sequence x[n] (replace with your sequence)
x = [1, 2, 3, 4];  % Example sequence (can be any given sequence)

% Length of the sequence
N = length(x);

% Initialize the DFT result
X = zeros(1, N);  % This will store the DFT values

% Compute the DFT using the DFT formula
for k = 1:N
    X(k) = sum(x .* exp(-1j * 2 * pi * (k-1) * (0:N-1) / N));  % DFT formula
end

% Plot the magnitude and phase of the DFT
figure;

% Magnitude plot
subplot(2,1,1);
stem(0:N-1, abs(X), 'filled', 'LineWidth', 2);
title('Magnitude of DFT');
xlabel('k');
ylabel('|X[k]|');
grid on;

% Phase plot
subplot(2,1,2);
stem(0:N-1, angle(X), 'filled', 'LineWidth', 2);
title('Phase of DFT');
xlabel('k');
ylabel('Angle of X[k] (radians)');
grid on;
```

OUTPUT



PRACTICAL No. 5

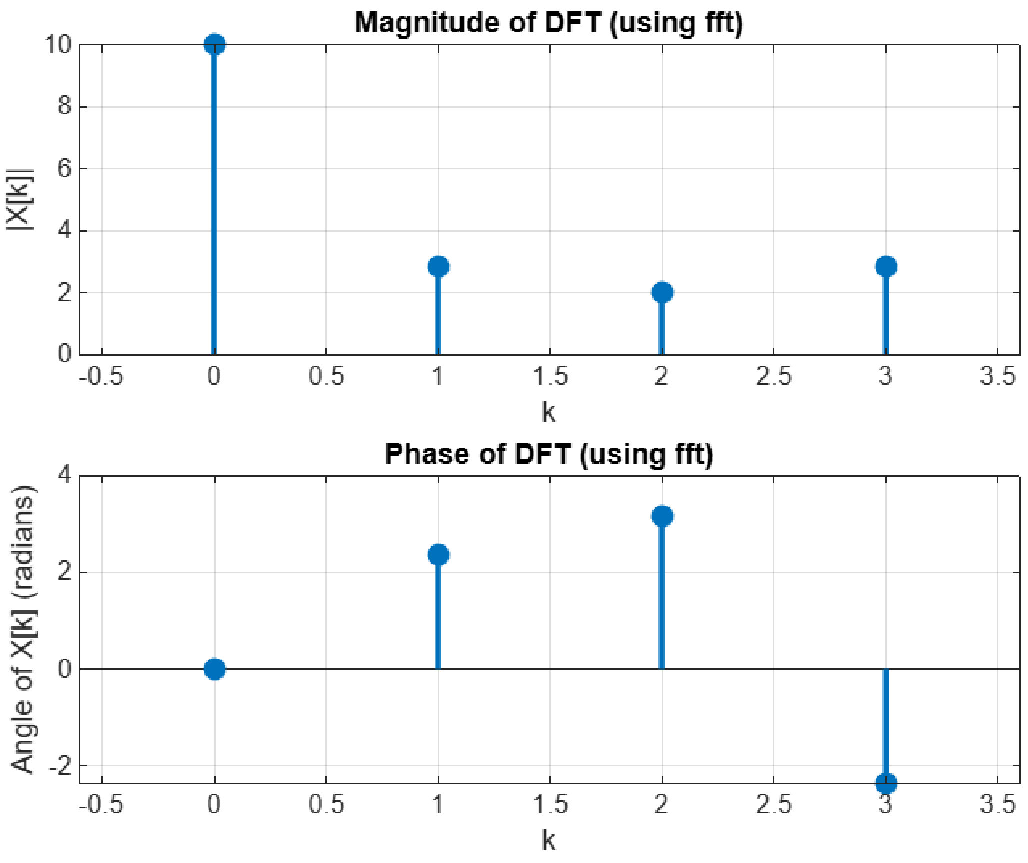
AIM

To find DFT of given sequence with and without FFT.

PROGRAM

```
% Main Script to Use fft for DFT
x = [1, 2, 3, 4];  % Example sequence (can be replaced with your own)

% Compute DFT using the fft function
X_fft = fft(x);

% Plot the magnitude and phase of the FFT
N = length(X_fft);  % Length of the sequence

figure;

% Magnitude plot (using fft)
subplot(2,1,1);
stem(0:N-1, abs(X_fft), 'filled', 'LineWidth', 2);
title('Magnitude of DFT (using fft)');
xlabel('k');
ylabel('|X[k]|');
grid on;

% Phase plot (using fft)
subplot(2,1,2);
stem(0:N-1, angle(X_fft), 'filled', 'LineWidth', 2);
title('Phase of DFT (using fft)');
xlabel('k');
ylabel('Angle of X[k] (radians)');
grid on;
```

OUTPUT



Magnitude of DFT (using fft)
|X[k]|
k
Phase of DFT (using fft)
Angle of X[k] (radians)
k

PRACTICAL No. 6

AIM

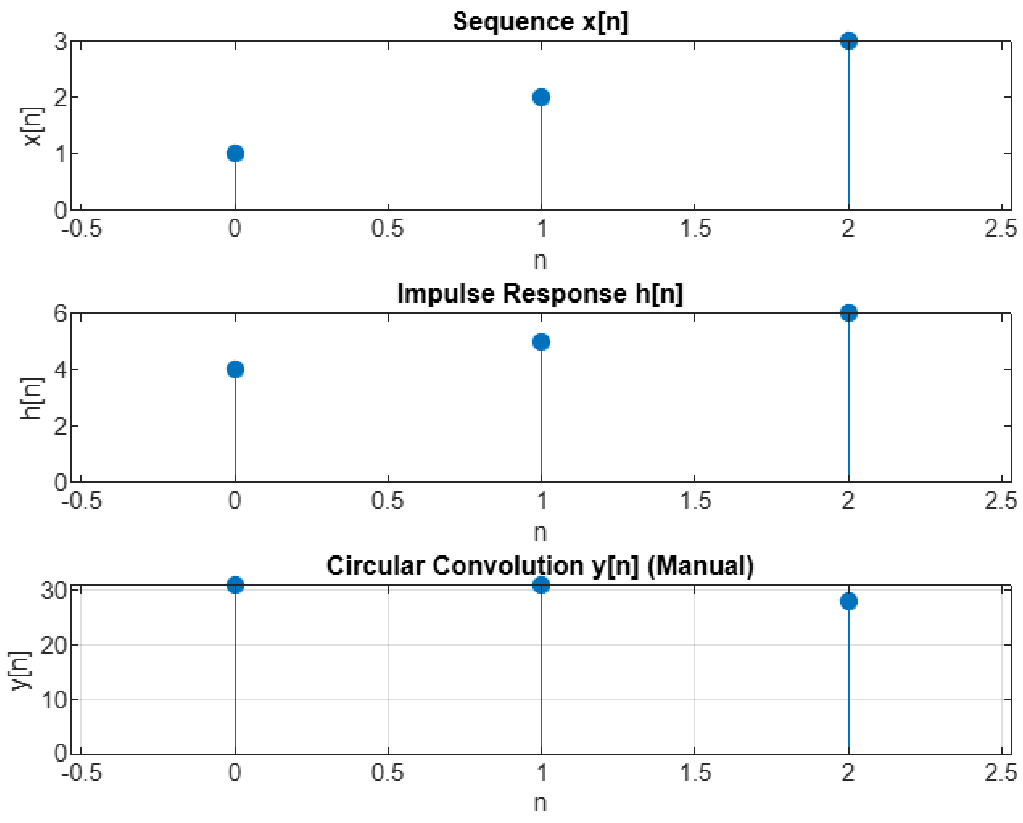
To find circular convolution with and without matlab's inbuilt functions.

PROGRAM (without 'cconv' function)

```
% Define two input sequences x[n] and h[n]
x = [1, 2, 3];  % Example input sequence x[n]
h = [4, 5, 6];  % Example impulse response h[n]

% Length of the sequences (assuming both have the same length)
N = length(x);

% Initialize the output sequence y[n] with zeros
y_manual = zeros(1, N);

% Perform circular convolution using the formula
for n = 1:N
    for m = 1:N
        % Circular index using modulo operation
        y_manual(n) = y_manual(n) + x(m) * h(mod(n - m, N) + 1);
    end
end

% Plot the input sequences and their circular convolution result
figure;

subplot(3,1,1);
stem(0:N-1, x, 'filled');
title('Sequence x[n]');
xlabel('n');
ylabel('x[n]');
```

```
subplot(3,1,2);
stem(0:N-1, h, 'filled');
title('Impulse Response h[n]');
xlabel('n');
ylabel('h[n]');

subplot(3,1,3);
stem(0:N-1, y_manual, 'filled');
title('Circular Convolution y[n] (Manual)');
xlabel('n');
ylabel('y[n]');
grid on;
```

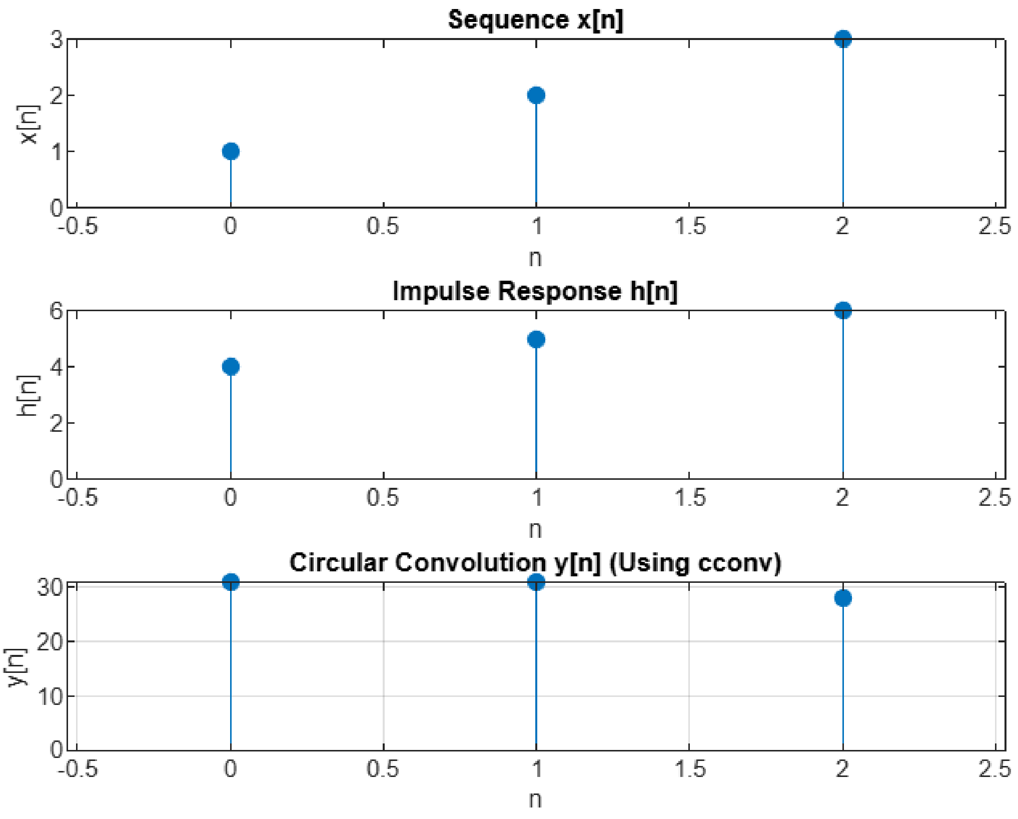
OUTPUT (without 'cconv' function)



PROGRAM (with ‘cconv’ function)

```
% Define two input sequences x[n] and h[n]
x = [1, 2, 3];  % Example input sequence x[n]
h = [4, 5, 6];  % Example impulse response h[n]

% Perform circular convolution using MATLAB's built-in cconv function
y_cconv = cconv(x, h, length(x));

% Plot the input sequences and their circular convolution result using cconv
figure;

subplot(3,1,1);
stem(0:length(x)-1, x, 'filled');
title('Sequence x[n]');
xlabel('n');
ylabel('x[n]');

subplot(3,1,2);
stem(0:length(h)-1, h, 'filled');
title('Impulse Response h[n]');
xlabel('n');
ylabel('h[n]');

subplot(3,1,3);
stem(0:length(y_cconv)-1, y_cconv, 'filled');
title('Circular Convolution y[n] (Using cconv)');
xlabel('n');
ylabel('y[n]');
grid on;
```

OUTPUT (with 'cconv' function)



PRACTICAL No. 7

AIM

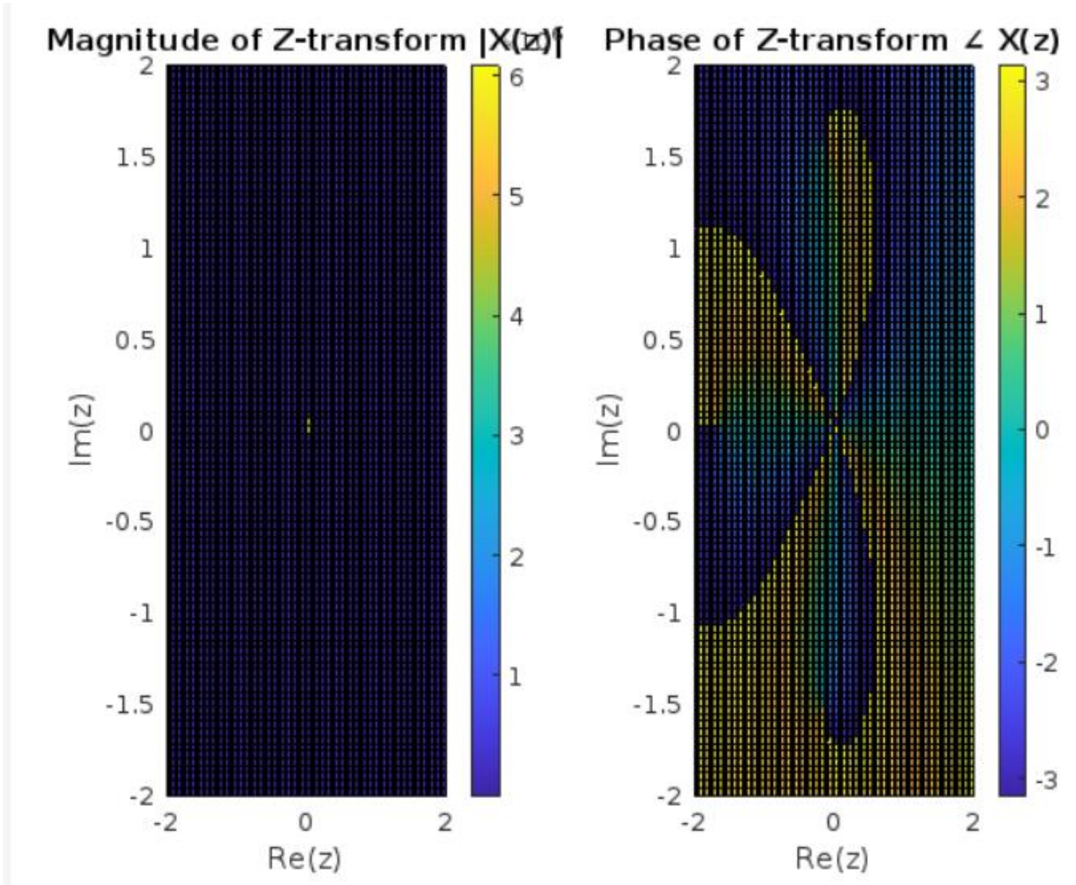
To implement Z transform of the given sequence.

PROGRAM (without 'ztrans' function)

```
% Define the sequence x[n] (replace with your sequence)
x = [1, 2, 3, 4];  % Example sequence (can be any given sequence)

% Length of the sequence
N = length(x);

% Define the range of z values (in complex plane)
% We choose a grid of points for z
re = linspace(-2, 2, 100);  % Real part of z
im = linspace(-2, 2, 100);  % Imaginary part of z

% Create a meshgrid of complex numbers (z = re + j*im)
[Re, Im] = meshgrid(re, im);
Z = Re + 1j * Im;  % Complex numbers

% Initialize the Z-transform result (X(z))
X_z = zeros(size(Z));

% Compute the Z-transform using the definition
for n = 1:N
    X_z = X_z + x(n) * Z.^(-n);  % Z-transform summation
end

% Plot the magnitude of X(z) over the complex plane
figure;
subplot(1,2,1);
surf(Re, Im, abs(X_z));
```

```
title('Magnitude of Z-transform |X(z)|');
xlabel('Re(z)');
ylabel('Im(z)');
zlabel('|X(z)|');
colorbar;
view(2);  % View in 2D for better visualization

% Plot the phase of X(z) over the complex plane
subplot(1,2,2);
surf(Re, Im, angle(X_z));
title('Phase of Z-transform \angle X(z)');
xlabel('Re(z)');
ylabel('Im(z)');
zlabel('Phase of X(z) (radians)');
colorbar;
view(2);  % View in 2D for better visualization;
```

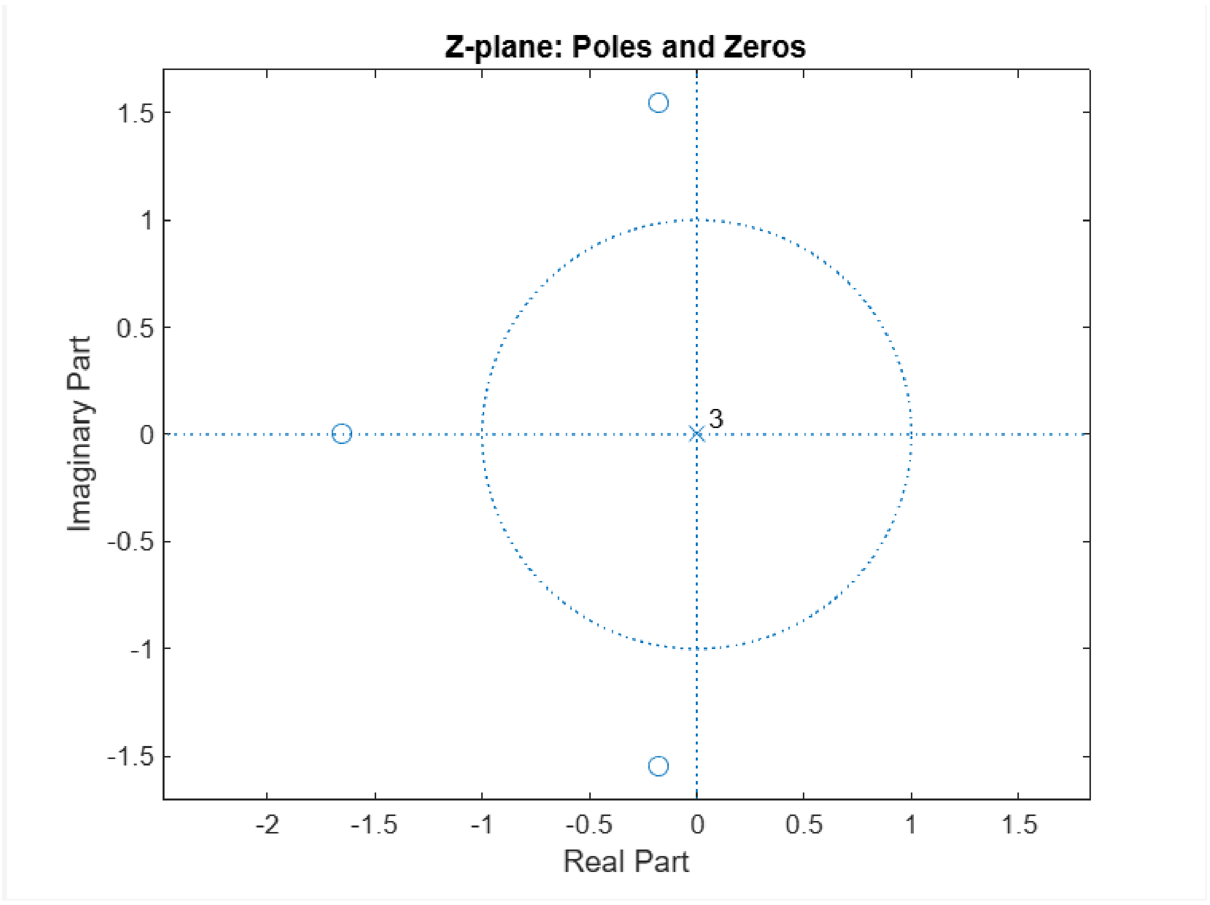
OUTPUT (without 'ztrans' function)



PROGRAM (with ‘ztrans’ function)

```
% Define two input sequences x[n] and h[n]
x = [1, 2, 3];  % Example input sequence x[n]
h = [4, 5, 6];  % Example impulse response h[n]

% Perform circular convolution using MATLAB's built-in cconv function
y_cconv = cconv(x, h, length(x));

% Plot the input sequences and their circular convolution result using cconv
figure;

subplot(3,1,1);
stem(0:length(x)-1, x, 'filled');
title('Sequence x[n]');
xlabel('n');
ylabel('x[n]');

subplot(3,1,2);
stem(0:length(h)-1, h, 'filled');
title('Impulse Response h[n]');
xlabel('n');
ylabel('h[n]');

subplot(3,1,3);
stem(0:length(y_cconv)-1, y_cconv, 'filled');
title('Circular Convolution y[n] (Using cconv)');
xlabel('n');
ylabel('y[n]');
grid on;
```

OUTPUT (with ‘ztrans’ function)



PRACTICAL No. 8

AIM

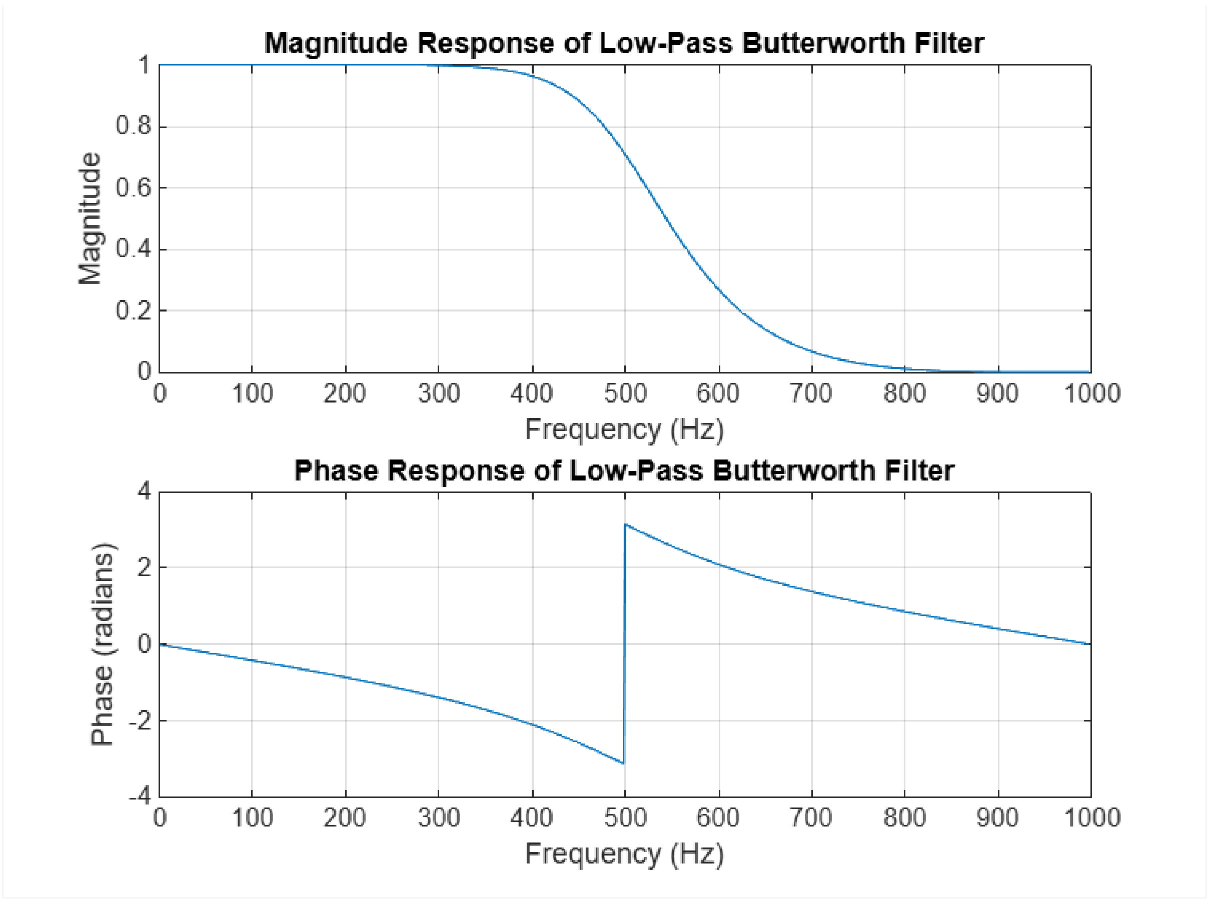
To design low pass butterworth filter.

PROGRAM

```
% Define filter parameters
fc = 500;   % Cutoff frequency in Hz
fs = 2000;  % Sampling frequency in Hz
N = 4;      % Filter order

% Normalize the cutoff frequency (fc / fs/2)
Wn = fc / (fs / 2);

% Design the low-pass Butterworth filter using the butter function
[b, a] = butter(N, Wn, 'low');  % 'low' specifies a low-pass filter

% Frequency response of the filter
[H, f] = freqz(b, a, 512, fs);  % Compute frequency response

% Plot the frequency response (magnitude and phase)
figure;

subplot(2,1,1);
plot(f, abs(H));  % Magnitude response
title('Magnitude Response of Low-Pass Butterworth Filter');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

subplot(2,1,2);
plot(f, angle(H));  % Phase response
title('Phase Response of Low-Pass Butterworth Filter');
```

```
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');
grid on;
```

OUTPUT


Magnitude Response of Low-Pass Butterworth Filter
Magnitude
Frequency (Hz)
Phase Response of Low-Pass Butterworth Filter
Phase (radians)
Frequency (Hz)