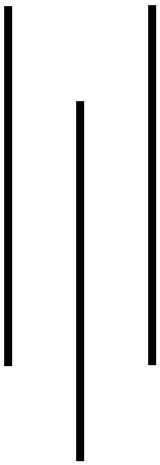
TRIVUWAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

LAB



Lab : DSAP Lab 1, 2, 3, 4, 5 and 6

Submitted By:

Name : Raman Bhattarai

Roll No. : THA077BEI033

Submitted To:

Department of Electronics And Computer Engineering

LAB 1:

Problem 1:

Calculate $(1 + \frac{2}{n^2})^{n}$ for n= 3, 7.

Code:

```
n = 7;
for i=1 : length(n)
n_i = n(i);
    result = (1 + 2/n_i^2)^n_i;
    fprintf('For n = %d, (1 + 2/n_i^2)^n_i = %.6f\n', n, result);
end
```

Output:

For n = 3, (1 + 2/n_i^2)^n_i = 1.825789
For n = 7, (1 + 2/n_i^2)^n_i = 1.323179

Analysis and Conclusion:

Using the formula above, the values of the given expression for the values of n as 3 and 7 are determined. By defining the value of n, the code only processes one value of n and outputs the result. The loop will go through each value in an array of values denoted by n, computing the corresponding expression as it goes.

This code demonstrates the use of loops, array indexing, mathematical operations and formatted output MATLAB.

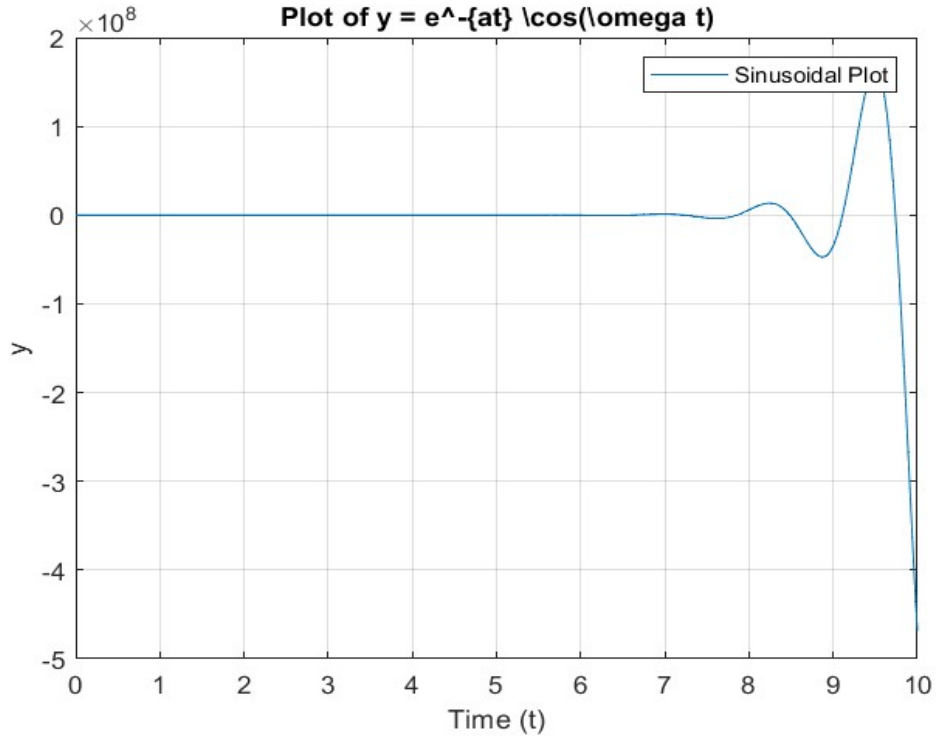
Problem 2:

Plot the function: y = $e^{-at} * cos(wt)$, for a = 2, ω= 5, and t = 0-10.

Code:

```
a = 2;
w = 5;
t = linspace(0, 10,1000);
y = exp(-a*t).* cos(w*t);
figure;
plot(t,y);
title('Plot of y = e^{-at} cos(wt)');
xlabel('Time (t)');
ylabel('y');
grid on;
```

Output:



Analysis and Conclusion:

The code generates a plot of the function: y = $e^{-at} * cos(wt)$,

In this code, parameters a and w are defined to create and plot the function over a range from 0 to 10 with 1000 points. Using the linspace function, a time vector t is generated. The element-wise operations in y = exp(-a*t) .* cos(w*t) ensure that each value of y is computed correctly for the corresponding value of t. The .* operator is crucial as it allows for element-wise multiplication between the exponential decay term and the cosine term, resulting in accurate values for plotting. The code then uses plotting functions to visualize the damped oscillatory behavior of y.

Problem 3:

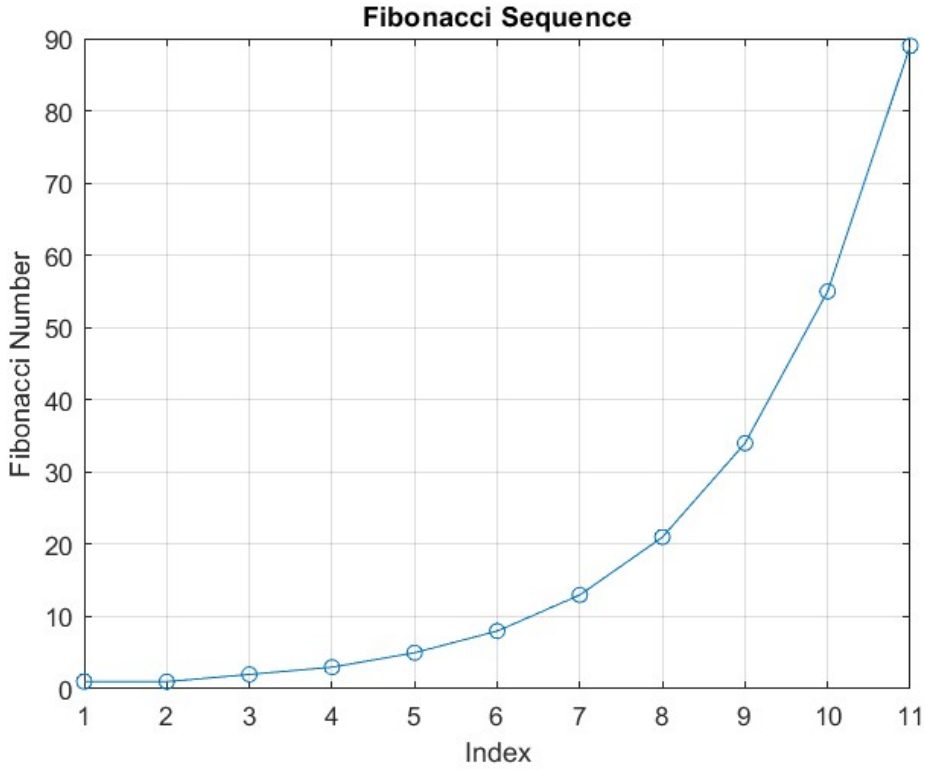
Try using the WHILE and the IF statements to calculate all the Fibonacci numbersso that the sum of two consecutive numbers is smaller than 10,000. How many are even? How many are odd? Try to plot them.

Code:

```
fib = [0, 1];
while fib(end) + fib(end-1) < 10000
    fib = [fib, fib(end) + fib(end-1)];
end
```

```
num_even = sum(rem(fib, 2) == 0);
num_odd = sum(rem(fib, 2) == 1);
fprintf('Number of even numbers: %d\n', num_even);
fprintf('Number of odd numbers: %d\n', num_odd);
figure;
plot(fib);
title('Fibonacci Series');
xlabel('Index');
ylabel('Value');
grid on;
```

OUTPUT:



Analysis and Conclusion:

The code generates the Fibonacci series up to a maximum value of 10,000, counts the number of even and odd numbers within the series, and then plots the series. The Fibonacci series is initialized with the two numbers 0 and 1. A while loop is used to generate the series by appending the sum of the last two elements to the fib array. The loop stops when the next Fibonacci number is 10,000 or greater. The number of even Fibonacci numbers obtained was 3 and number of odd Fibonacci numbers obtained was 8.

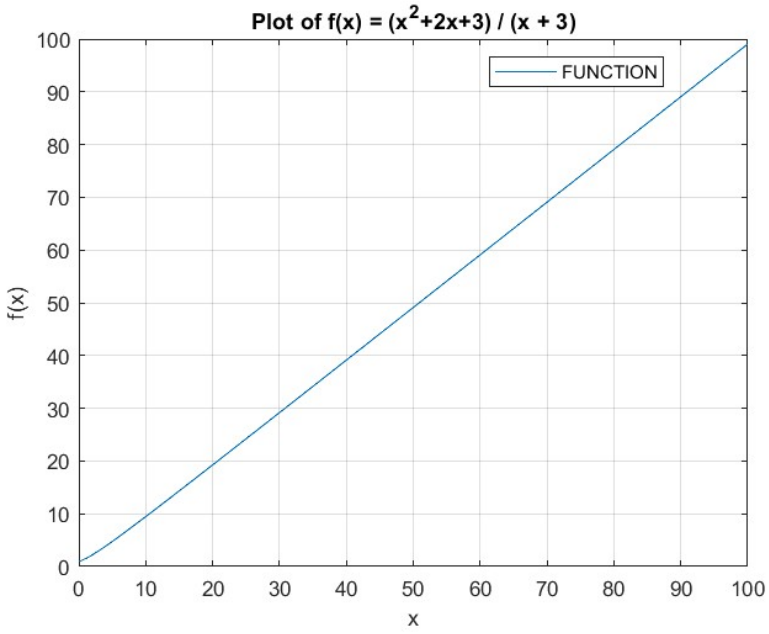
Problem 4:

Given $f(x)=\frac{x^2+2x+3}{x+3}$. Plot *f(x)* for $0 \leq x \leq 100$.

Code:

```
f = @(x) (x.^2 + 2*x + 3) ./ (x + 3);
x = linspace(0, 100, 1000);
y = f(x);
for i = 1:length(x)
      fprintf('f(%f) = %f\n', x(i), y(i));
end figure;
plot(x, y);
title('Plot of f(x) = (x^2 + 2x + 3) / (x + 3)', 'FontSize', 20);
xlabel('x', 'FontSize', 16);
ylabel('f(x)', 'FontSize', 16);
grid on;
```

OUTPUT:



Analysis and Conclusion:

The plot shows a linear trend as x increases from 0 to 100. This suggests that the function behaves like a linear function over this range. The function f(x) can be simplified by performing polynomial long division:

$$f(x) = \frac{x^2+2x+3}{x+3} = x - 1 + \frac{6}{x+3}$$

The y-intercept occurs when $x = 0$.

LAB 2:

Problem 1:

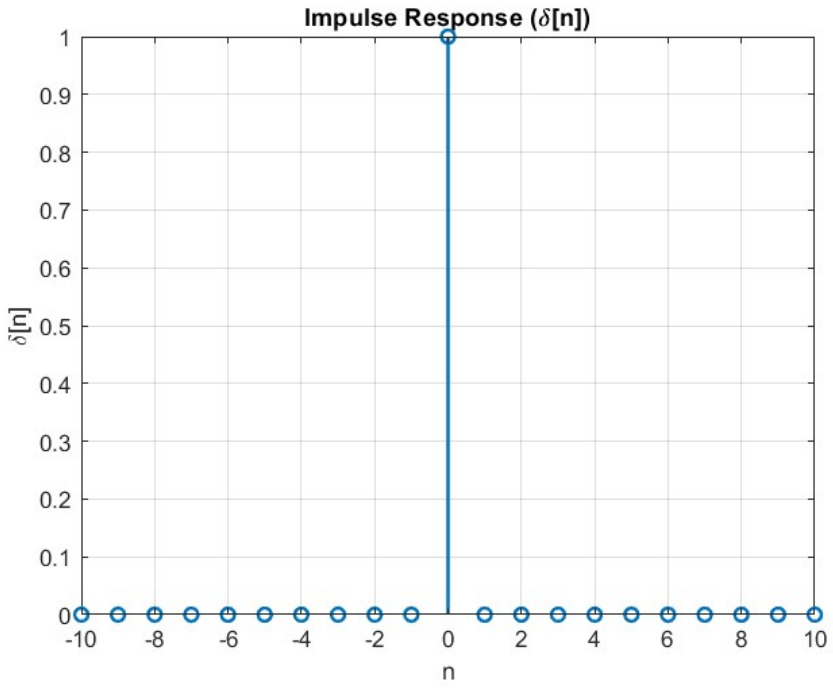
Plot the basic signal using MATLAB

1. **Unit Impulse**

Code:

```
t = -5:0.1:5;
ImpulseResponse = t==0;
subplot(2, 1, 1);
stem(t,ImpulseResponse,'LineWidth',2,'Marker','o','MarkerSize',8);
title('Impulse Response(\delta[n]) (Stem)', 'FontSize', 14);
xlabel('Time (t)', 'FontSize', 12);
ylabel('Amplitude', 'FontSize', 12);
grid on;
axis([min(t), max(t), 0, 1.5]);
subplot(2, 1, 2);
plot(t, ImpulseResponse, 'LineWidth', 2);
title('Impulse Response(\delta[n]) (Plot)', 'FontSize', 14);
xlabel('Time (t)', 'FontSize', 12);
ylabel('Amplitude', 'FontSize', 12);
grid on;
axis([min(t), max(t), 0, 1.5]);
```

Output:

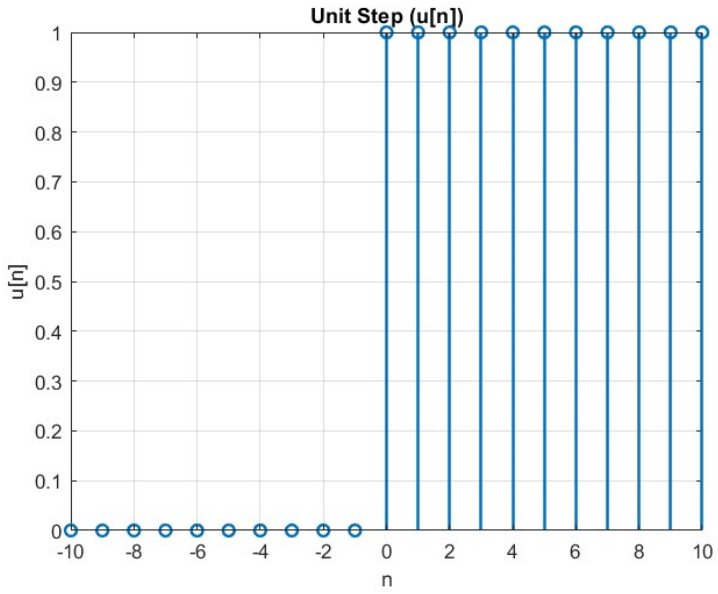


2. **Unit Step**

Code:

```
t = -5:0.1:5;
u = zeros(size(t));
u(t >= 0) = 1;
subplot(2, 1, 1);
stem(t, u, 'LineWidth', 2, 'Marker', 'o', 'MarkerSize', 8);
title('Unit Step Signal (Stem)');
xlabel('Time (t)');
ylabel('Amplitude');
grid on;
axis([min(t), max(t), 0, 1.5]);
subplot(2, 1, 2);
plot(t, u, 'LineWidth', 2);
title('Unit Step Signal (Plot)');
xlabel('Time (t)');
ylabel('Amplitude');
grid on;
axis([min(t), max(t), 0, 1.5]);
```

Output:



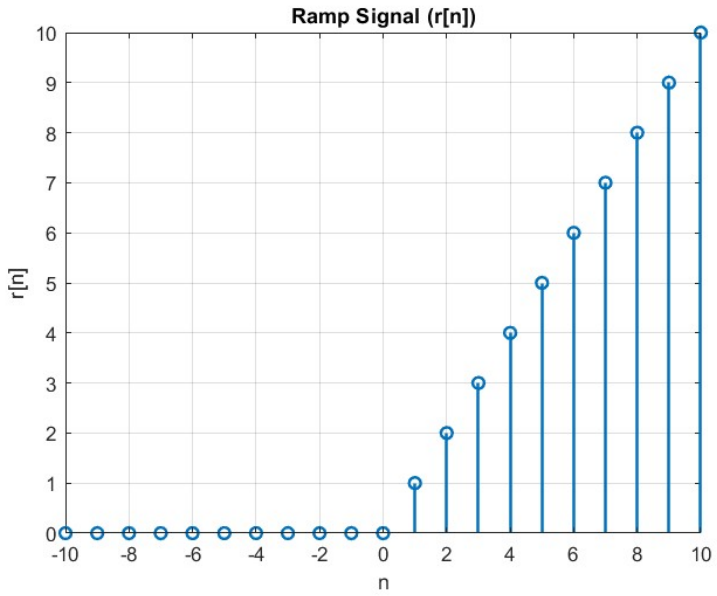
3. **Ramp**

Code:

```
unitStep = zeros(size(t));
unitStep(t >= 0) = 1;
unitRamp = t .* unitStep;
subplot(1, 2, 1);
stem(t, unitRamp,'LineWidth',1,'Marker','o', 'MarkerSize', 8);
title('Unit Ramp Signal (Stem)');
xlabel('Time (t)');
```

```
ylabel('Amplitude');
grid on;
axis([min(t), max(t), 0, max(unitRamp) + 0.5]);
subplot(1, 2, 2);
plot(t, unitRamp, 'LineWidth', 2);
title('Unit Ramp Signal (Plot)');
xlabel('Time (t)');
ylabel('Amplitude');
grid on;
axis([min(t), max(t), 0, max(unitRamp) + 0.5]);
```

Output:



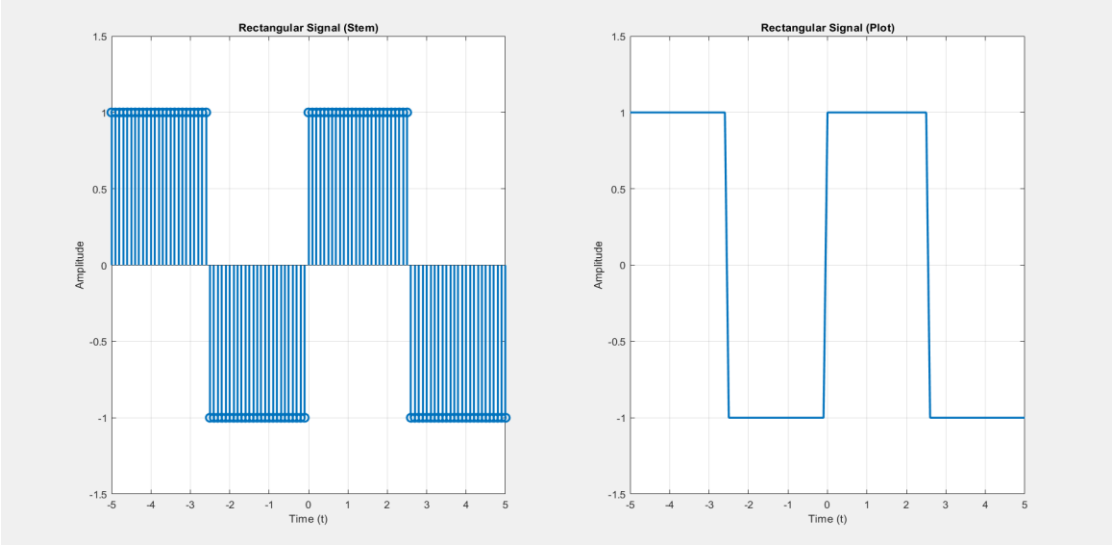
4. **Rectangular**

Code:

```
t = -5:0.1:5;
fs = 1/5;
amplitude = 1;
rect = amplitude * sin(2*pi*fs*t);
rect(rect>=0)= amplitude;
rect(rect<0)= -amplitude;
subplot(1, 2, 1);
stem(t, rect, 'LineWidth', 2, 'Marker', 'o', 'MarkerSize', 8);
title('Rectangular Signal (Stem)');
xlabel('Time (t)');
ylabel('Amplitude');
grid on;
axis([min(t), max(t), -1.5, 1.5]);
```

```
subplot(1, 2, 2);
plot(t, rect, 'LineWidth', 2);
title('Rectangular Signal (Plot)');
xlabel('Time (t)');
ylabel('Amplitude');
grid on;
axis([min(t), max(t), -1.5, 1.5]);
```

Output:



Problem 2:

Plot the following continuous-time signals.

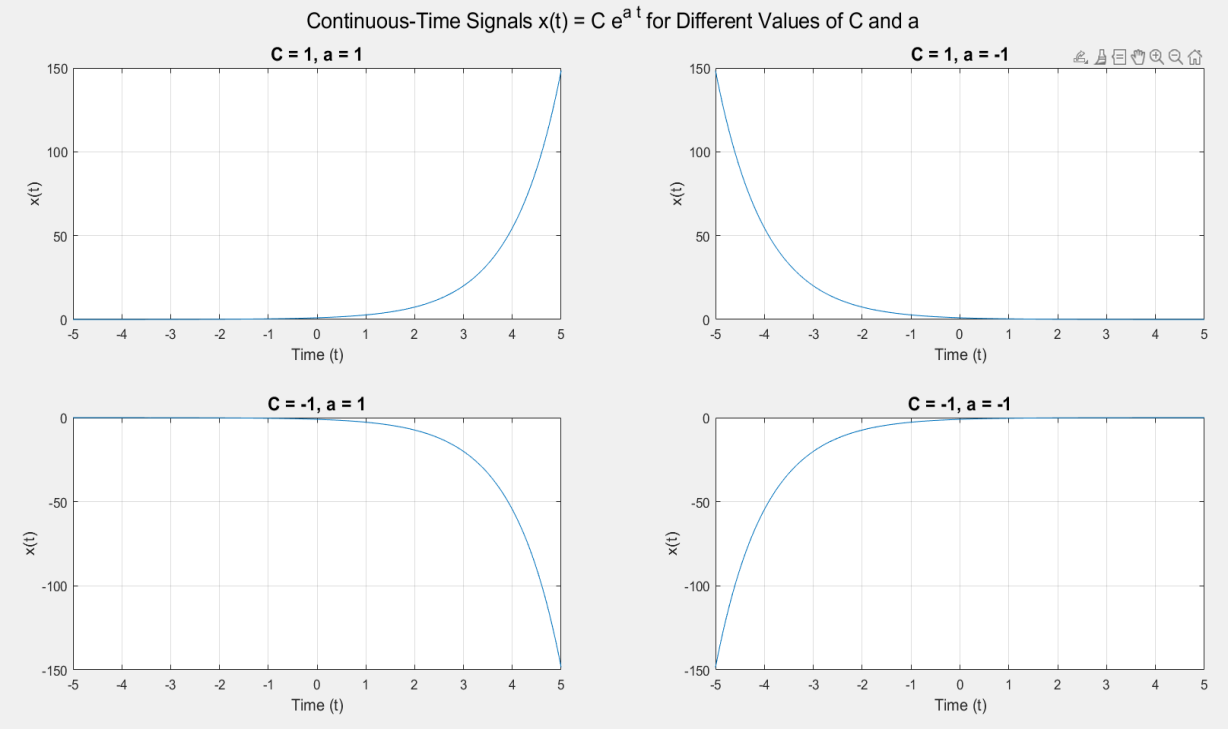
1) $x(t) = Ce^{at}$ **where C and a are real numbers and choose C and a both positive and negative.**

Code:

```
t = -5:0.01:5;
C_values = [1, -1];
a_values = [1, -1];
figure;
for i = 1:length(C_values)
  for j = 1:length(a_values)
    C = C_values(i);
    a = a_values(j);
    x_t = C * exp(a * t);
    subplot(length(C_values),length(a_values),(i-1)*length(a_values)+j);
    plot(t, x_t);
    title(['C = ', num2str(C), ', a =', num2str(a)],'FontSize', 14);
    xlabel('Time (t)', 'FontSize', 12);
    ylabel('x(t)','FontSize',12);
    grid on;
```

```
 end
sgtitle('Continuous-Time Signals x(t) = C e^{a t} for Different Values of C
and a', 'FontSize', 16);
```

Output:



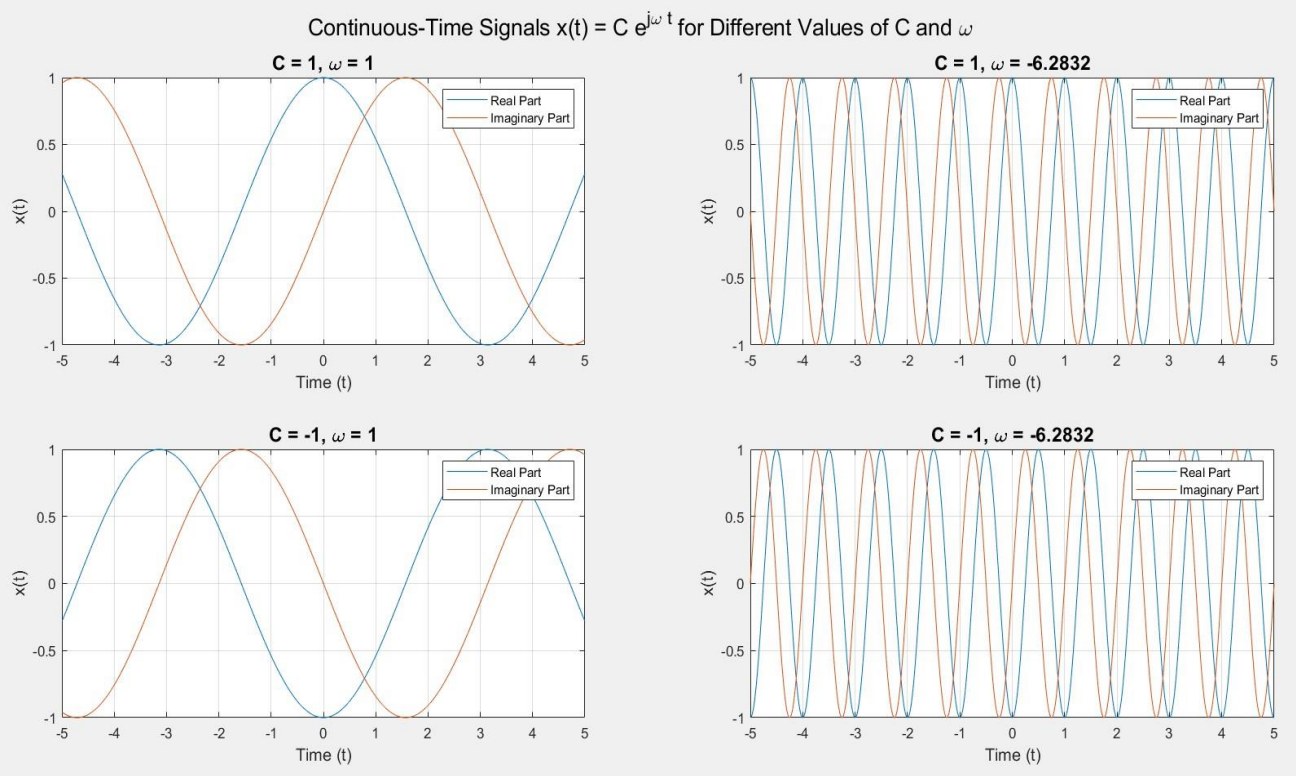
2) Plot the same signal taking *a* as pure imaginary number.

Code:

```
t = -5:0.01:5;
C_values = [1, -1];
omega_values = [1, -2*pi];
figure;
for i = 1:length(C_values)
    for j = 1:length(omega_values)
      C = C_values(i);
      omega = omega_values(j);
      x_t = C * exp(1i * omega * t); subplot(length(C_values),length(omega_values),(i-
      1)*length(omega_values)+j);
      plot(t, real(x_t)); hold on;
      plot(t, imag(x_t)); hold off;
    title(['C=',num2str(C),',\omega =',num2str(omega)],'FontSize',14);
    xlabel('Time (t)', 'FontSize', 12);
    ylabel('x(t)', 'FontSize', 12);
    legend('Real Part', 'Imaginary Part');
    grid on;
end
end
```

```
sgtitle('Continuous-Time Signals x(t) = C e^{j\omega t} for Different Values of C
and \omega', 'FontSize', 16);
```

Output:



3) Consider complex exponential signal as specified in b) where C is expressed in polar form i.e., ($C=|C|e^{j\theta}$) & a in rectangular form i.e., ($a=r+j\omega o$). Then your function $x(t)$, on simplification, becomes

$$x(t)= |C|\, e^{rt} * [cos(\omega ot+\theta) + jsin(\omega ot+\theta)]$$

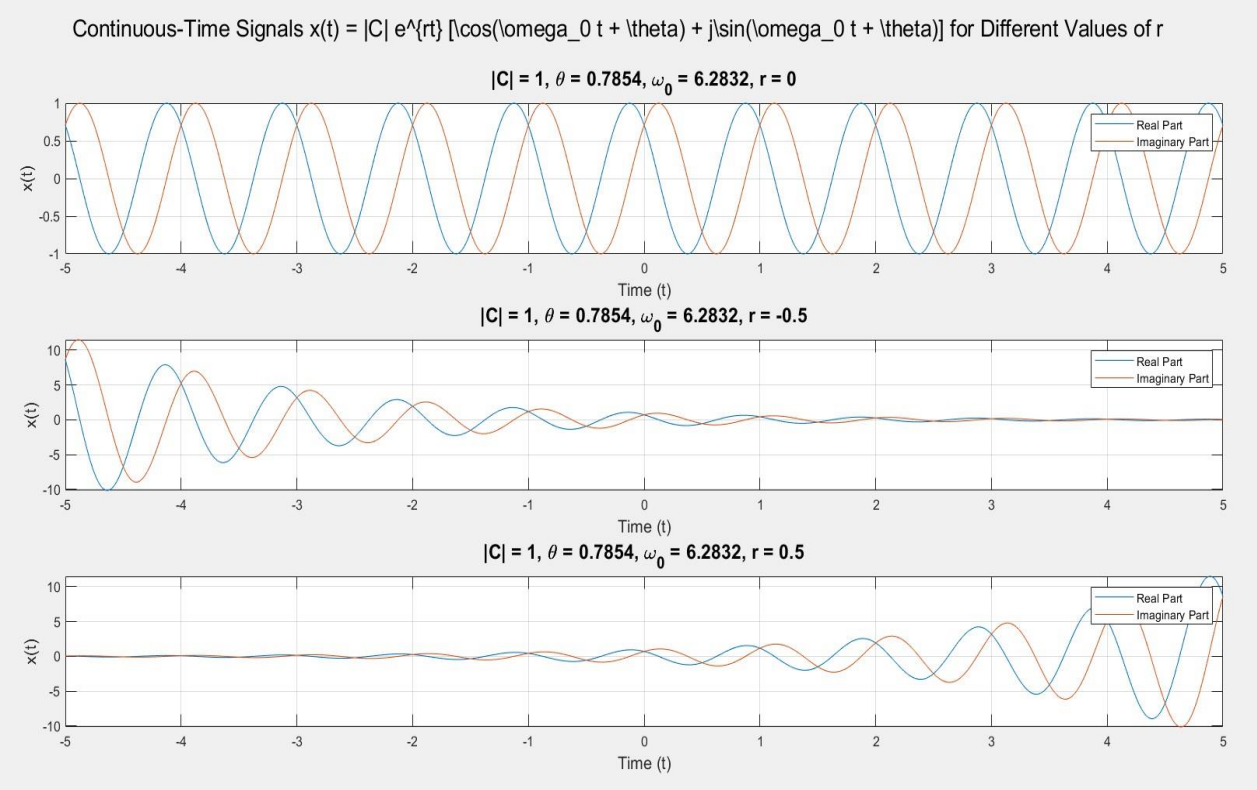
Now, plot the signal for different values of r and comment on the results. i. $r=0$ ii. $r<0$ iii. $r>0$

Code:

```
t = -5:0.01:5;
C_magnitude = 1;
% |C| theta = pi/4;
% ϑ omega_0 = 2 * pi;
% ω₀ r_values = [0, -0.5, 0.5];
figure;
for i = 1:length(r_values)
    r = r_values(i);
    x_t = C_magnitude * exp(r * t) .* (cos(omega_0 * t + theta) + 1i * ...
          sin(omega_0 * t + theta));
    subplot(length(r_values), 1, i);
    plot(t, real(x_t));
```

```
    hold on;
    plot(t, imag(x_t));
    hold off;
    title(['|C| = ', num2str(C_magnitude), ', \theta = ', ...
          num2str(theta), ', \omega_0 = ', num2str(omega_0), [', ' ...
          'r = '], num2str(r)], 'FontSize', 14);
    xlabel('Time (t)', 'FontSize', 12);
    ylabel('x(t)', 'FontSize', 12);
    legend('Real Part', 'Imaginary Part');
    grid on;
end
sgtitle(['Continuous-Time Signals x(t) = |C| e^{rt} [\cos(\omega_0 t + ' ...
'\theta) + j\sin(\omega_0 t + \theta)] for Different Values of r'], 'FontSize',
16);
```

Output:



Analysis and Conclusion:

The signal $x(t)= |C|e^{rt}[cos(\omega ot+\theta)+jsin(\omega ot+\theta)]$ can be broken down as follows:

1. Exponential Term
2. Complex Exponential

This form shows that the signal consists of a complex sinusoid modulated by an exponential function.

The behavior of the signal is influenced by the value of r:

a. The amplitude of the signal remains constant over time for r=0.
b. The signal's amplitude decreases exponentially, leading to a damping effect for r<0.
c. The signal's amplitude increases exponentially, leading to an amplification effect for r>0.

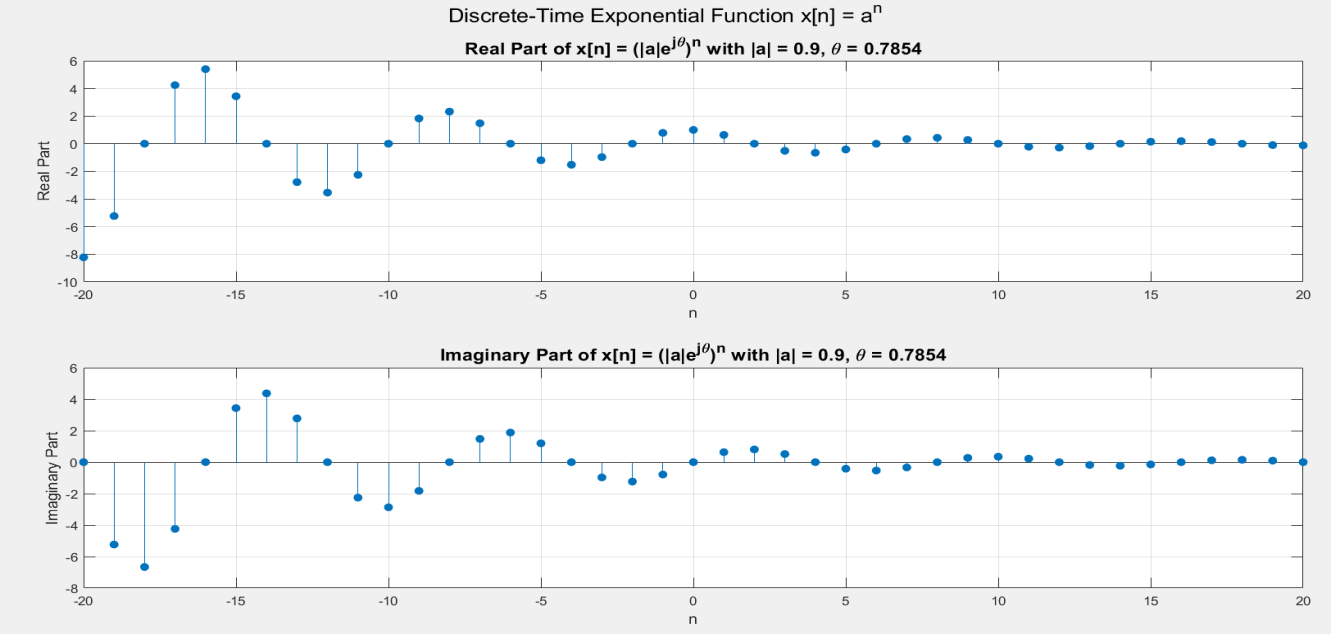
Problem 3:

Plot the DT exponential function $x[n] = a^n$, $a = |a|e^{j\theta}$. Choose the suitable value of |a| and Θ, then plot the function x[n].

Code:

```
n = -20:20;
a_magnitude = 0.9;
% |a| theta = pi/4; % ϑ
a = a_magnitude * exp(1i * theta);
x_n = a .^ n;
figure;
subplot(2, 1, 1);
stem(n, real(x_n), 'filled');
title(['Real Part of x[n] = (|a|e^{j\theta})^n with |a| = ' ... '', num2str(a_magnitude),
', \theta = ', num2str(theta)], ... 'FontSize', 14);
xlabel('n', 'FontSize', 12);
ylabel('Real Part', 'FontSize', 12);
grid on;
subplot(2, 1, 2);
stem(n, imag(x_n), 'filled');
title(['Imaginary Part of x[n] = (|a|e^{j\theta})^n with |a| = ' ... '',
num2str(a_magnitude), ', \theta = ', num2str(theta)], ... 'FontSize', 14);
xlabel('n', 'FontSize', 12); ylabel('Imaginary Part', 'FontSize', 12);
grid on;
sgtitle(['Discrete-Time Exponential Function x[n] = ' ... 'a^n'], 'FontSize', 16);
```

Output:



Analysis and Conclusion:

The discrete-time exponential function $x[n] = (|a|e^{j\theta})^n$ can exhibit different behaviors based on the values of |a| and θ. In the given example, |a|=0.9 and θ=π/4, the signal exhibits an exponentially decaying oscillation. This is due to |a|<1 causing decay and θ≠0 causing oscillations.

Problem 4:

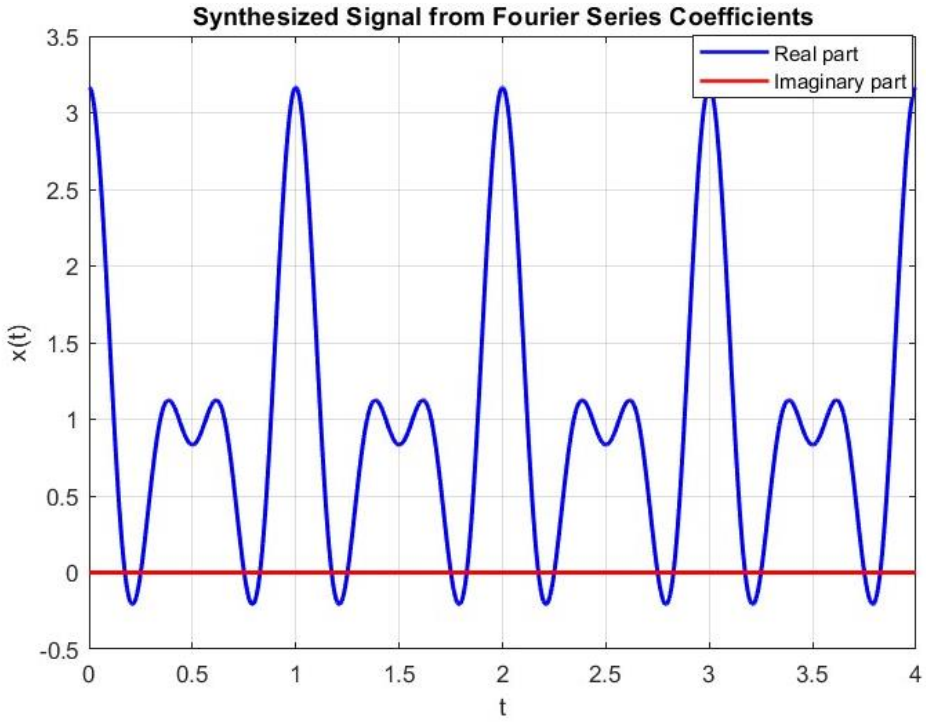
Synthesize the signal from the FS (Fourier Series) Coefficients as *C0=1, C1=C-1=1/4, C2=C-2=1/2, C3=C-3=1/3*.

Code:

```
t = linspace(-2, 2, 1000);
omega_0 = 2 * pi;
% Since T = 1 second C0 = 1;
C1 = 1/4;
C2 = 1/2;
C3 = 1/3;
x_t = C0 ...
      +C1*exp(1i * omega_0 * t) + conj(C1) * exp(-1i * omega_0 * t) ...
      +C2*exp(2i * omega_0 * t) + conj(C2) * exp(-2i * omega_0 * t) ...
      +C3*exp(3i * omega_0 * t) + conj(C3) * exp(-3i * omega_0 * t);
figure;
subplot(2, 1, 1);
plot(t, real(x_t), 'LineWidth', 1.5);
title('Real Part of Synthesized Signal', 'FontSize', 14);
xlabel('Time (t)', 'FontSize', 12);
ylabel('Real Part of x(t)', 'FontSize', 12);
```

```
grid on;
subplot(2, 1, 2);
plot(t, imag(x_t), 'LineWidth', 1.5);
title('Imaginary Part of Synthesized Signal', 'FontSize', 14);
xlabel('Time (t)', 'FontSize', 12);
ylabel('Imaginary Part of x(t)', 'FontSize', 12);
grid on;
sgtitle('Synthesized Signal from Fourier Series Coefficients','FontSize',16);
```

Output:



Analysis and Conclusion:

The real part of the signal *x(t)* shows the actual synthesized signal from the given Fourier coefficients. If all Fourier coefficients are real, the imaginary part of *x(t)* should be zero. This confirms that the signal synthesized from real coefficients will have no imaginary component.

Problem 5:

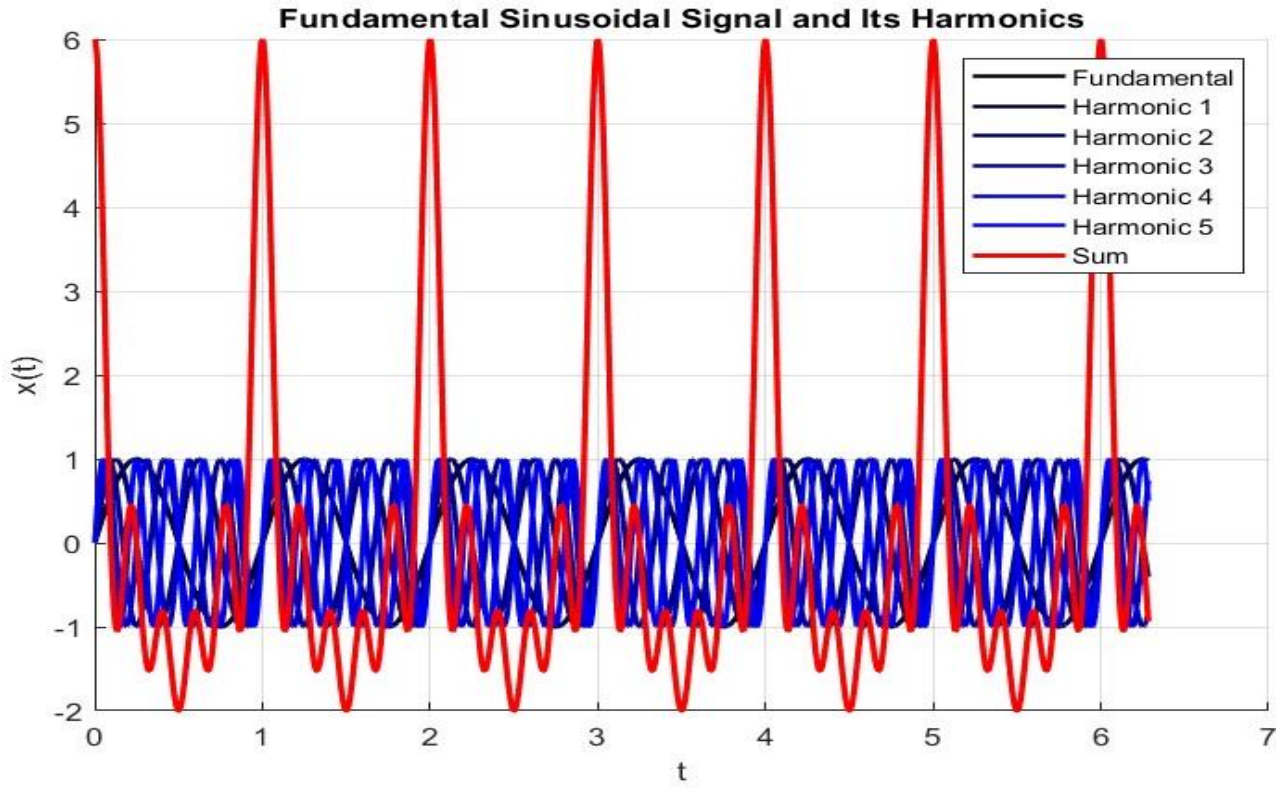
Plot fundamental sinusoidal signal, its higher harmonics up to 5th harmonics and add all of them to see the result. Comment on the result.

Code:

```
omega_0 = 2*pi;
N = 5;
A = 1;
```

```
t = linspace(0, 2*pi, 1000);
x_t = A * cos(omega_0 * t);

for n = 1:N
    x_t = x_t + A * cos(n * omega_0 * t);
end

figure;
hold on;
plot(t, A * sin(omega_0 * t), 'k-', 'LineWidth', 1.5);
for n = 1:N
    plot(t, A * sin(n * omega_0 * t), 'Color', [0 0 1] * (n/N), 'LineWidth', 1.5);
end
plot(t, x_t, 'r-', 'LineWidth', 2);
hold off;

title('Fundamental Sinusoidal Signal and Its Harmonics');
xlabel('t');
ylabel('x(t)');
legend('Fundamental', 'Harmonic 1', 'Harmonic 2', 'Harmonic 3', 'Harmonic 4',
'Harmonic 5', 'Sum');
grid on;
```

Output:



Analysis and Conclusion:

1. The fundamental signal is a simple sinusoid with a frequency of 1 Hz.
2. Each harmonic is a sinusoid with a frequency that is an integer multiple of the fundamental frequency.
3. The resulting signal from the sum of the fundamental and harmonics shows a more complex waveform. This demonstrates how adding harmonics to a fundamental frequency can create more complex periodic signals. This principle is the basis of Fourier series.

LAB 3:

Problem 1:

Find the convolution result of the following signal using basic convolution formula:

X1(n1) = [1, 1, 1, 1, 1], n1= [-2, -1, 0, 1, 2]

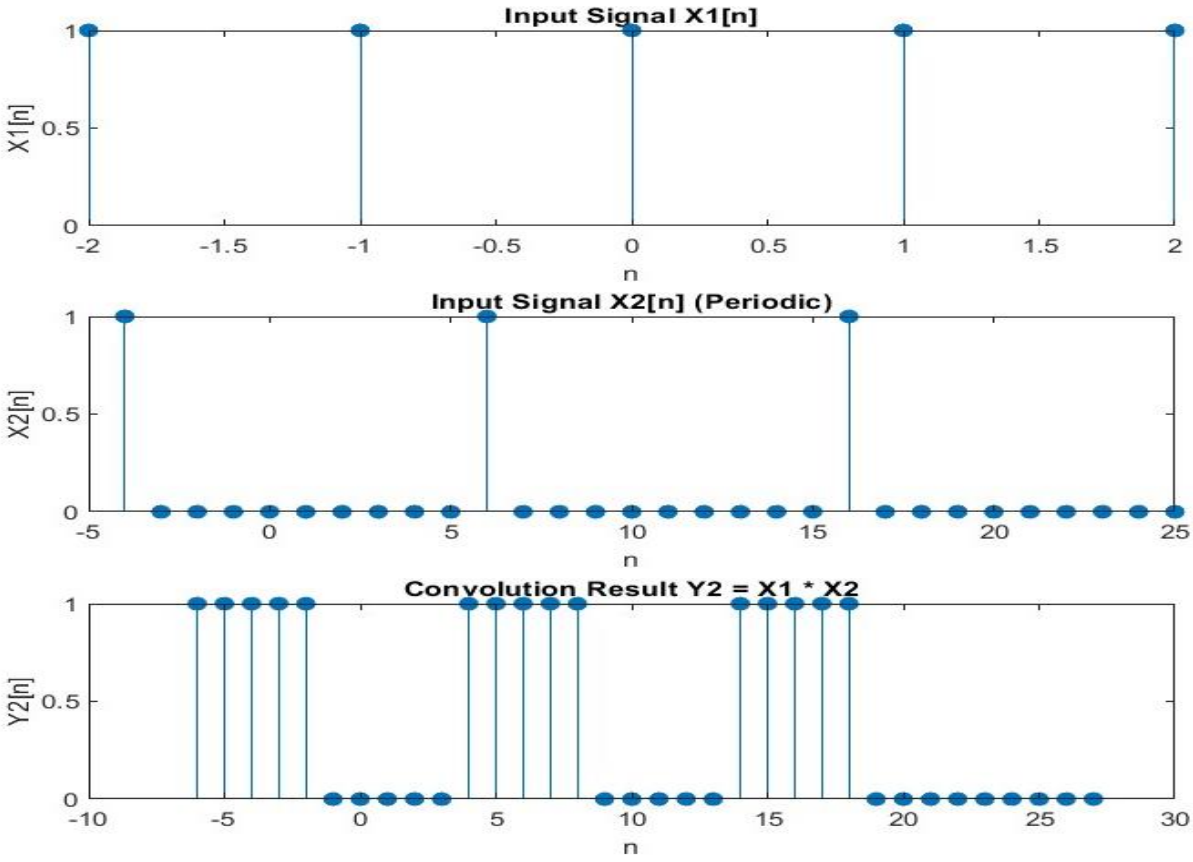
X2(n2) = [1,0,0,0,0,0,0,0,0,0], n2 = [-4, -3, -2, -1, 0, 1, 2, 3, 4, 5]

X2 is a periodic signal. Y2= X1*X2

Code:

```
X1 = [1, 1, 1, 1, 1];
n1 = -2:2;
X2_period = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0];
n2_period = -4:5;
num_repeats = 3;
X2 = repmat(X2_period, 1, num_repeats);
n2 = -4:(length(X2)-5);
Y2 = conv(X1, X2);
n_Y2 = (n1(1) + n2(1)):(n1(end) + n2(end));
figure; subplot(3, 1, 1);
stem(n1, X1, 'filled');
title('Input Signal X1[n]');
xlabel('n');
ylabel('X1[n]');
subplot(3, 1, 2);
stem(n2, X2, 'filled');
title('Input Signal X2[n] (Periodic)');
xlabel('n'); ylabel('X2[n]');
subplot(3, 1, 3);
stem(n_Y2, Y2, 'filled');
title('Convolution Result Y2 = X1 * X2');
xlabel('n');
ylabel('Y2[n]');
disp('Convolution result Y2:');
disp(Y2);
```

Output:



Analysis and Conclusion:

Convolution of two discrete-time signals X1[n] and X2[n] combines the effects of both signals. It effectively slides one signal over the other, multiplying and summing the overlapping values. The result, Y2[n], shows how the rectangular pulse X1[n] modifies the periodic impulse X2[n]. The sum accumulates the overlapping products of X1 and X2, reflecting how the shape and duration of X1 influence the periodic structure of X2. The Y2 columns from -6 to 27 are:

Y2[n] = [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]

Problem 2:

Find the convolution of the following using conv function:

- **X[n]=** $\begin{cases} \frac{1}{3}n & for\ 0 \le n \le 6 \\ 0 & else \end{cases}$

  $h[n] = \begin{cases} 1 & for\ -2 \le n \le 2 \\ 0 & else \end{cases}$

Code:

```
n_x = 0:6;
x = (1/3) * n_x;
n_h = -2:2;
h = ones(size(n_h));
y = conv(x, h);
n_y = (n_x(1) + n_h(1)) : (n_x(end) + n_h(end));
subplot(3,1,1);
stem(n_x, x, 'filled');
title('Input Signal (x[n])', 'FontSize', 14);
xlabel('n', 'FontSize', 12);
ylabel('x[n]', 'FontSize', 12);
xlim([n_x(1)-1, n_x(end)+1]);
grid on;
subplot(3,1,2);
stem(n_h, h, 'filled');
title('Impulse Response(h[n])', 'FontSize', 14);
xlabel('n', 'FontSize', 12);
ylabel('h[n]', 'FontSize', 12);
xlim([n_h(1)-1, n_h(end)+1]);
grid on;
subplot(3,1,3);
stem(n_y, y, 'filled');
title('Convolution y[n] = x[n] * h[n]', 'FontSize', 14);
xlabel('n', 'FontSize', 12);
ylabel('y[n]', 'FontSize', 12);
xlim([n_y(1)-1, n_y(end)+1]);
grid on;
disp('Input Signal:');
disp(x);
disp('Impulse Response:');
disp(h);
disp('Convolution result:');
disp(y);
```

Output:

Input Signal:

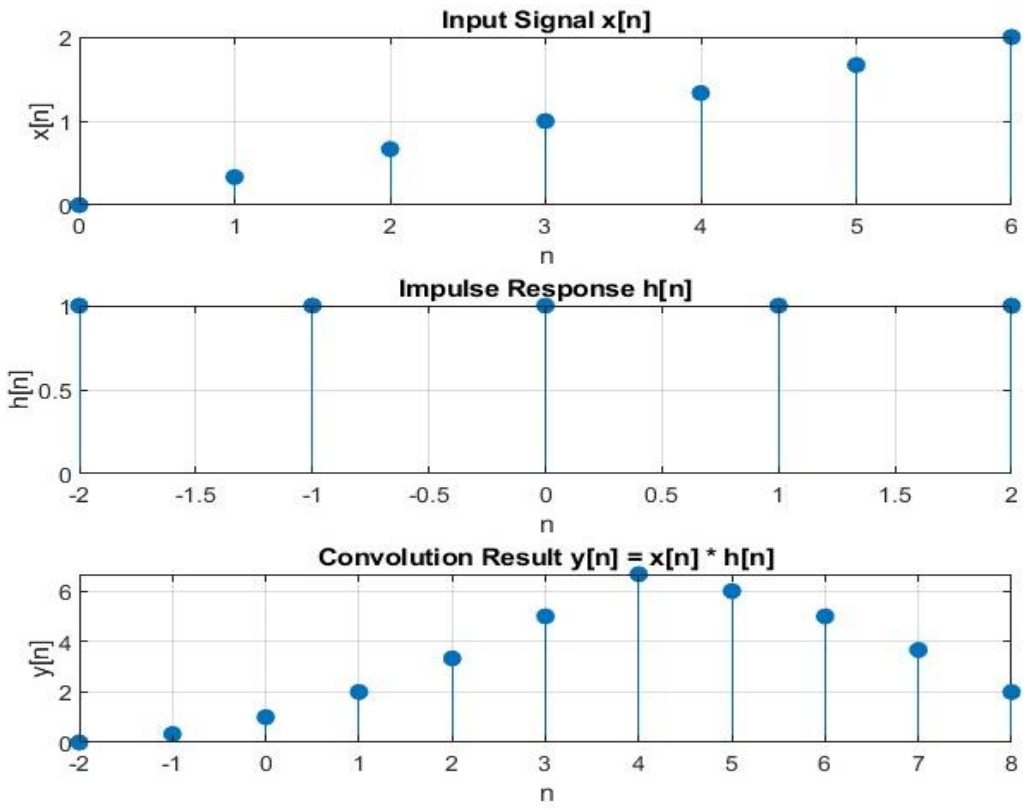
0 0.3333 0.6667 1.0000 1.3333 1.6667 2.0000

Impulse Response:

1 1 1 1 1

Convolution result:

[0, 0.3333, 1.0000, 2.0000, 3.3333, 5.0000, 6.6667, 6.0000, 5.0000, 3.6667, 2.0000]



Analysis and Conclusion:

Above, the convolution of two discrete sequences, x[n] and h[n], is computed. The sequence x[n] is defined as a linearly increasing sequence from 0 to 6, where each value is given by (1/3) n. The sequence h[n] is a rectangular pulse with a value of 1 for indices ranging from -2 to 2. The convolution result, y[n], combines these two sequences. The resulting plot displays three subplots: the first shows x[n], the second shows h[n], and the third illustrates the convolution result y[n].

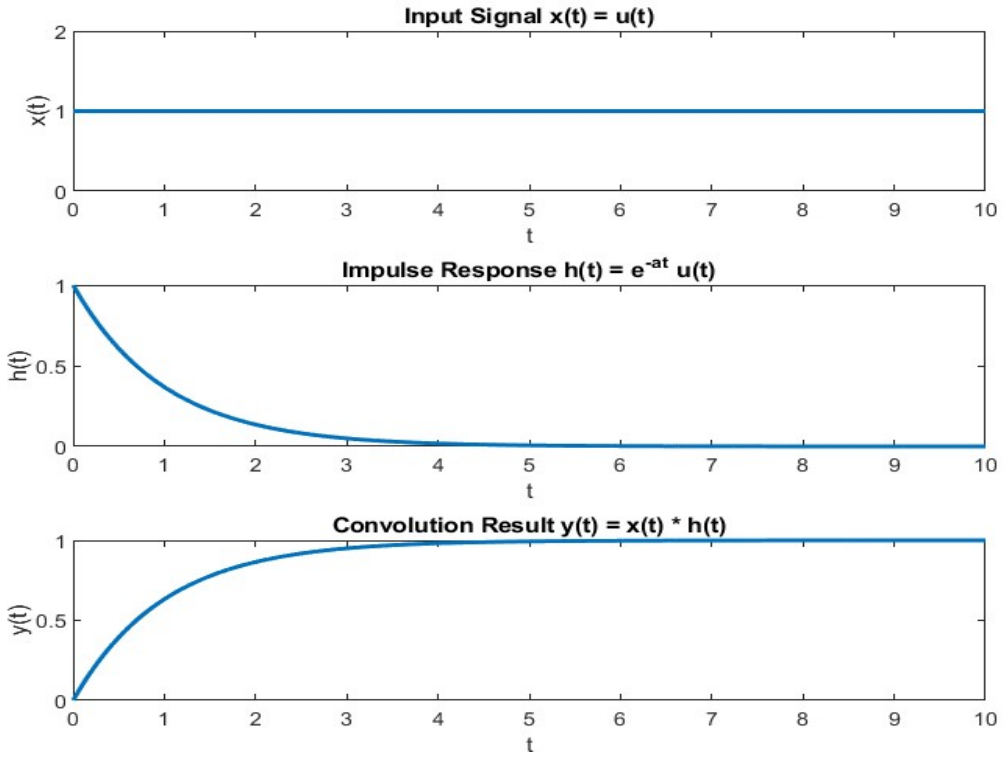
- **$x(t) = u(t)$, $h(t) = e^{-at}u(t)$, where a > 0**

Code:

```
a = 1;
t = 0:0.01:10;
x = ones(size(t));
% x(t) = u(t)
h = exp(-a * t);
% h(t) = e^{-at} u(t)
y = (1/a) * (1 - exp(-a * t));
% Convolution result figure;
subplot(3, 1, 1);
plot(t, x, 'LineWidth', 1.5);
title('Input Signal x(t) = u(t)', 'FontSize', 14); xlabel('t', 'FontSize', 12);
ylabel('x(t)', 'FontSize', 12);
xlim([0, 10]);
```

```
grid on;
subplot(3, 1, 2);
plot(t, h, 'LineWidth', 1.5);
title('Impulse Response h(t) = e^{-at} u(t)', 'FontSize', 14); xlabel('t',
'FontSize', 12);
ylabel('h(t)', 'FontSize', 12);
xlim([0, 10]); grid on; subplot(3, 1, 3);
plot(t, y, 'LineWidth', 1.5);
title('Convolution Result y(t) = x(t) * h(t)', 'FontSize', 14); xlabel('t',
'FontSize', 12);
ylabel('y(t)', 'FontSize', 12);
xlim([0, 10]);
grid on;
disp('Convolution result y(t) = (1/a) * (1 - exp(-at)):');
disp(y);
```

Output:



Analysis and Conclusion:

The result (convolution) $y[t] = x[t] * h[t]$ of above given signals of LTI system shows how the system's output evolves from result of applying input signal to the system characterized by the impulse response. The result reflects the weighted sum of the shifted versions of the impulse response based on the input's duration and values. The convolution of a unit step *x(t)* with an exponential decaying impulse response *h(t)* resulted in the output *y(t)* that starts at 0 and rises towards a steady-state value of 1.

Problem 3:

Consider two discrete time sequences *x[n]* and *h[n]* given by:

x[n] = 1 for 0 ≤ n ≤ 4, elsewhere 0,

h[n] = 2n for 0 ≤ n ≤ 6, elsewhere 0.

a) Find the response of the LTI system with impulse response *h[n]* to input *x[n].*

b) Plot the signals and comment on the result.

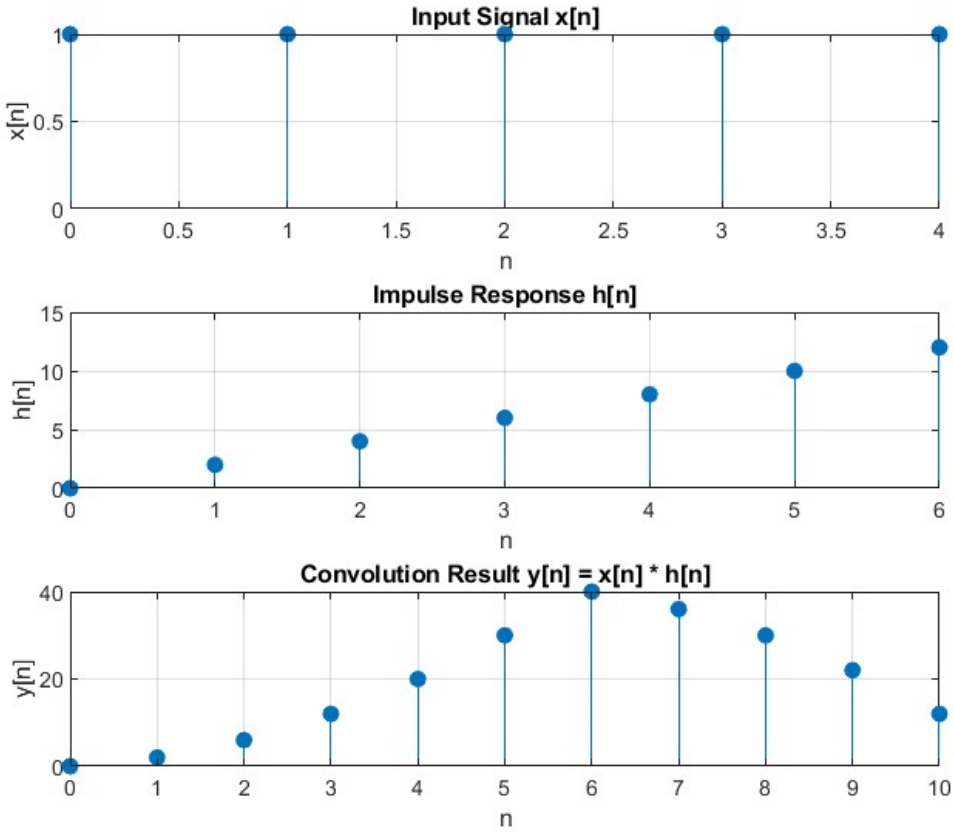
Code:

```
n_x = 0:4;
x = ones(size(n_x));
n_h = 0:6;
h = 2 * n_h;
y = conv(x, h);
n_y = (n_x(1) + n_h(1)) : (n_x(end) + n_h(end));
subplot(3, 1, 1);
stem(n_x, x, 'filled');
title('Input Signal x[n]');
xlabel('n');
ylabel('x[n]');
xlim([n_x(1)-1, n_x(end)+1]);
grid on;
subplot(3, 1, 2);
stem(n_h, h, 'filled');
title('Impulse Response h[n]');
xlabel('n');
ylabel('h[n]');
xlim([n_h(1)-1, n_h(end)+1]);
grid on;
subplot(3, 1, 3);
stem(n_y, y, 'filled');
title('Convolution Result y[n] = x[n] * h[n]');
xlabel('n');
ylabel('y[n]');
xlim([n_y(1)-1, n_y(end)+1]);
grid on;
disp('Convolution result y[n]:');
disp(y);
```

OUTPUT:

Convolution result y[n]:

0 2 6 12 20 30 40 36 30 22 12



Analysis and Conclusion:

The input signal $x[n]$ is a simple step function that is active for a finite range 0 to 4 and the impulse response $h[n]$ represents a linearly increasing response, indicating that the system's output grows over time with value 2^n from 0 to 6 for this system. The result of the convolution of these functions shows that the system accumulates the effects of the impulse response over the duration of the input signal. The result starts at 0 and gradually increases, reflecting the integration of the impulse response's effect over the period n.

Problem 4:

If the impulse response of a LTI system is given by sinc function as

$$h[n] = \frac{2\tau}{T_p}\, sinc(k\, \frac{2\tau}{T_p})$$

and input signal is a rectangular wave given by

x(t) = 1 for 1 to 100

0 elsewhere,

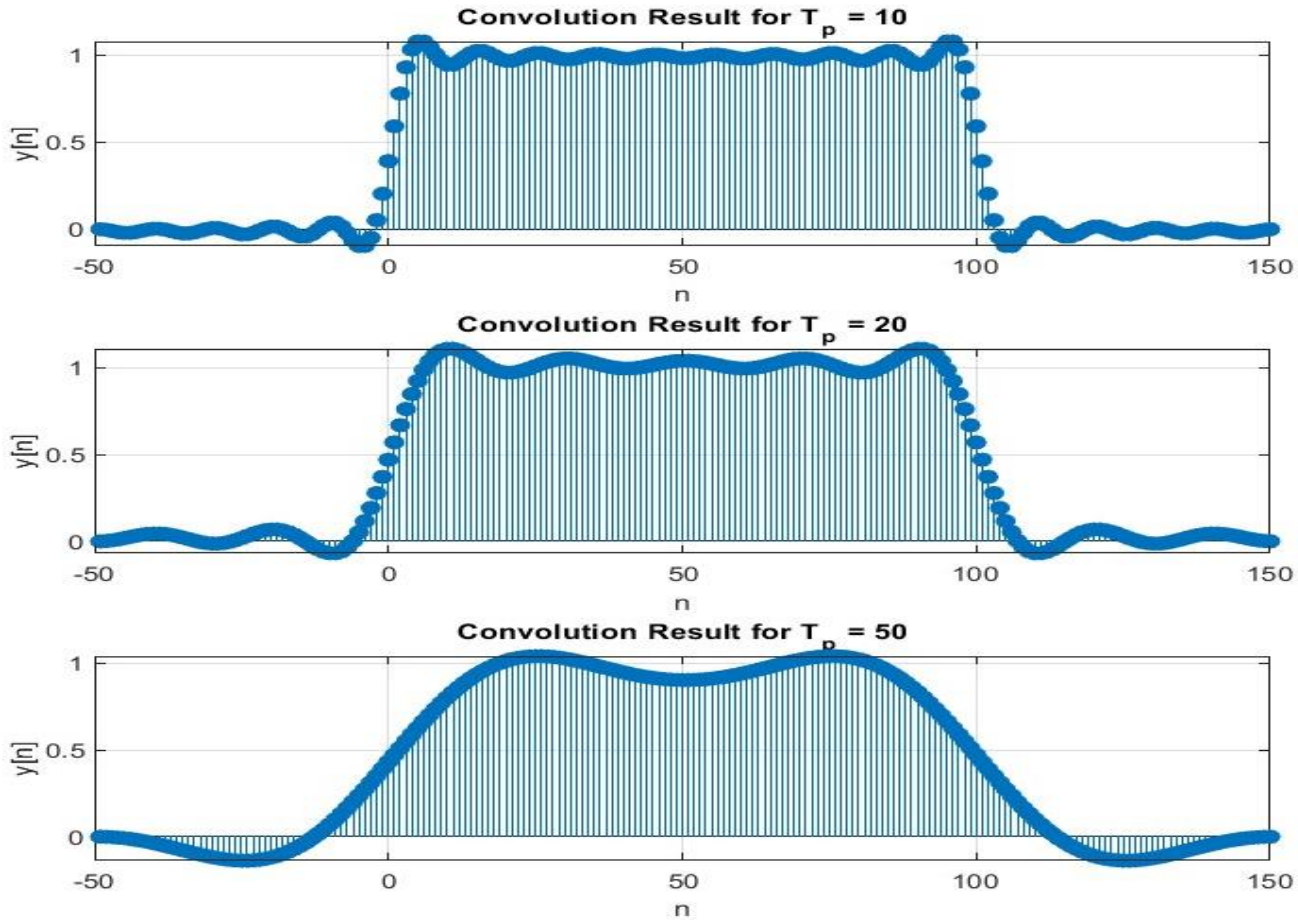
Find output of the system for different values of τ. Comment on the result.

Code:

At $\tau = 1$:

```
n_x = 1:100;
x = ones(1, 100);
Tp_values = [10, 20, 50];
tou = 1;
figure;
for i = 1:length(Tp_values)
      Tp = Tp_values(i);
      k = -50:50;
      h = (2*tou/Tp) * arrayfun(@(x) sinc(x * 2*tou / Tp), k);
      y = conv(x, h);
      n_y = (n_x(1) + k(1)):(n_x(end) + k(end)); subplot(length(Tp_values), 1, i);
      stem(n_y, y, 'filled');
      title(['Convolution Result for T_p =',num2str(Tp)], 'FontSize',14);
      xlabel('n', 'FontSize',12);
      ylabel('y[n]', 'FontSize',12);
      grid on;
end
disp('Convolution results for different T_p values have been plotted.');
```

Output:



Analysis and Conclusion:

- For smaller values of Tp, the sinc function is narrower and has higher oscillations. The convolution result shows that the rectangular pulse is modulated by a sinc function that is almost like a sharp peak with significant side lobes.
- As Tp increases, the sinc function becomes wider and its side lobes become less pronounced. The convolution result shows a smoother transition, with less pronounced oscillations.
- For large values of Tp, the sinc function is very wide and smooth, with minimal oscillations. The result reflects a smooth, nearly continuous transition, where the rectangular pulse is effectively averaged over a wide range.
- **Impact of Tp:**

  As Tp increases, the width of the sinc function increases, which causes the system's impulse response to become smoother.
- The system's behavior in terms of filtering or smoothing is evident from the convolution results.

LAB 4:

Problem 1:

In the given LTI system of fig above, if the coefficients 'b' & 'a' are specified as b0=0.0663, b1= 0.1989, b2=0.1989, b3=0.0663 a0=1, a1= -0.9349, a2=0.5668, a3= -0.1015, then the order of the system is 3 i.e. N=3.

a. Plot the frequency response of the system.

b. From the magnitude response of the system, find out the cut-off frequency.

c. Identify the nature of the system analyzing its frequency response.

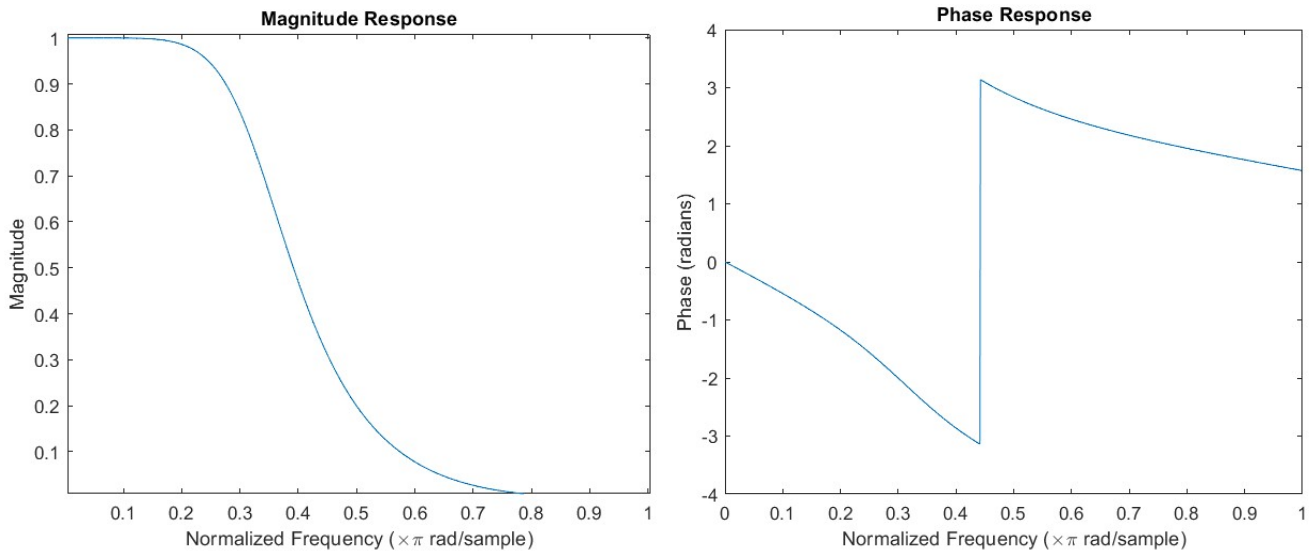
Code:

```
b = [0.0663 0.1989 0.1989 0.0663];
a = [1 -0.9349 0.5668 -0.1015];
[freq_response, w] = freqz(b, a, 'half', 1024);
figure;
plot(w/pi, abs(freq_response));
title('Magnitude Response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude');
figure;
plot(w/pi, angle(freq_response));
title('Phase Response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Phase (radians)');
magnitude = abs(freq_response);
cutoff_index = find(magnitude <= (max(magnitude) / sqrt(2)), 1);
cutoff_frequency = w(cutoff_index) / pi;
disp(['Cut-off Frequency: ', num2str(cutoff_frequency), ' x ? rad/sample']);
if cutoff_frequency < 0.5
    disp('The system behaves like a low-pass filter.');
elseif cutoff_frequency > 0.5
    disp('The system behaves like a high-pass filter.');
else
    disp('The system has characteristics of a band-pass or band-stop filter.');
end
```

Output:

The cut-off Frequency: 0.33887 rad/sample.

The system behaves like a low-pass filter.



Analysis and Conclusion:

The LTI system with coefficient 'a' and 'b' is plotted. The magnitude and phase response of the LTI system is obtained. From the magnitude response and phase response we can observe that the system acts as a low-pass filter. The cut-off frequency of the LTI system is obtained to be 0.33887 rad/sample.

Problem 2:

The transfer function of the fourth-order discrete time system is given as:

$$H(z) = \frac{0.0018 + 0.0073z^{-1} + 0.011z^{-2} + 0.007z^{-3} + 0.008z^{-4}}{1 - 3.0544z^{-1} + 3.8291z^{-2} - 2.2925z^{-3} + 0.55072z^{-4}}$$

Using MATLAB

a. Find out the poles and zeros of the system and plot them in the z- plane.

b. Use them to determine the second order sections in the cascaded form.

c. Plot the frequency response of the system and comment on the nature of the system.

d. After knowing the numerator and denominator coefficients of each second order section, draw the signal flow graph to represent the cascaded structure.

Code:

```
num = [0.0018, 0.0073, 0.011, 0.007, 0.008];
den = [1, -3.0544, 3.8921, -2.2925, 0.55072];
sys = tf(num, den, -1);
poles = pole(sys);
zeros = zero(sys);
figure;
zplane(zeros, poles);
title('Pole-Zero Plot in the z-plane');
xlabel('Real Part');
ylabel('Imaginary Part');
grid on;
[sos, g] = tf2sos(num, den);
```

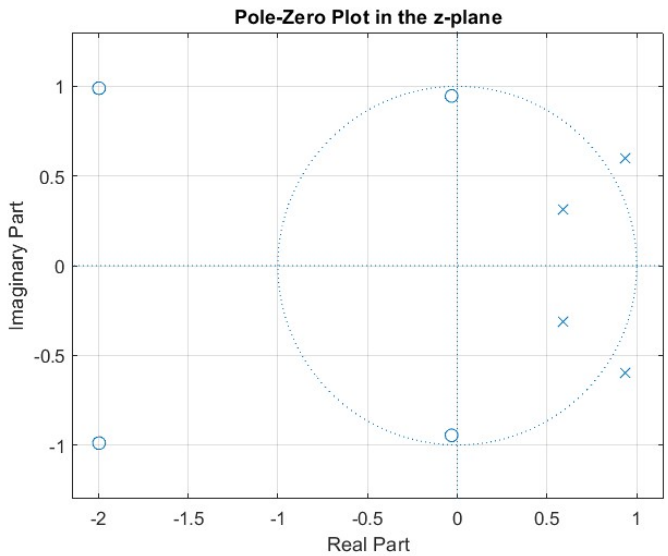
```
disp('Second-order sections:');
disp(sos);
disp('Gain:');
disp(g);
figure;
freqz(num, den);
title('Frequency Response of the System');
grid on;
figure;
for i = 1:size(sos, 1)
    b = sos(i, 1:3);
    a = sos(i, 4:6);

    subplot(size(sos, 1), 1, i);
    zplane(b, a);
    title(['Second-Order Section ', num2str(i)]);
    xlabel('Real Part');
    ylabel('Imaginary Part');
    grid on;
end
```

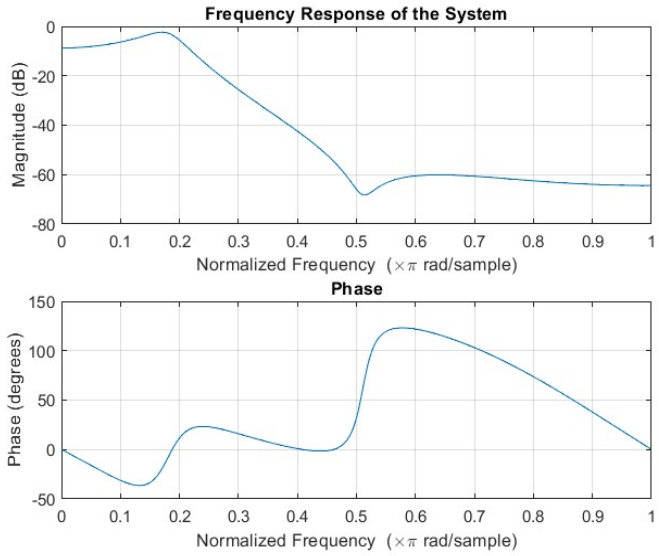
Output:

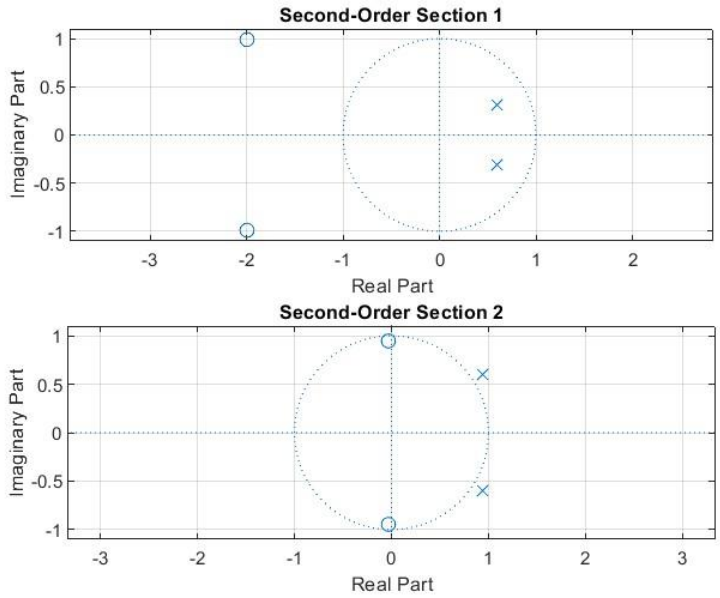
The second order section are:

1.0000	3.9923	4.9632	1.0000	-1.1793	0.4460
1.0000	0.0632	0.8955	1.0000	-1.8751	1.2347

The gain is 0.0018







Analysis and Conclusion:

From the given transfer function H(z) the poles and zeros are obtained. The poles and zeros are plotted in the z-plane. The magnitude and phase response of the system is plotted. The second order sections in cascade form obtained are: 1.0000, 3.9923, 4.9632, 1.0000, -1.1793, 0.4460, 1.0000, 0.0632, 0.8955, 1.0000, -1.875, 1.2347. The gain of H(z) is 0.0018. The signal flow graph to represent the cascaded structure is plotted.

Problem 3:

Let a discrete time system be implemented by cascading of the following three second order sections:

a. Using above three second order sections in cascaded form determine the poles and zeros of the system and plot them in z-plane.

b. Determine the transfer function of the system, formed by cascading of the above three sections. Determine the poles and zeros from this transfer function and plot them in z plane. Your result should match with that from 3(a).

c. Draw the direct form Structures-I & II of the system.

Code:

```
b1 = [0.0007378, 0.0007378*2, 0.0007378];
a1 = [1, -1.2686, 0.7051];
b2 = [1, 2, 1];
a2 = [1, -1.016, 0.3583];
b3 = [1, 2, 1];
a3 = [1, -0.9044, 0.2155];
sos = [b1, a1; b2, a2; b3, a3];
 g = [1; 1; 1];
 [b, a] = sos2tf(sos, g);
```

```
disp('Numerator coefficients of the overall transfer function:');
disp(b);
disp('Denominator coefficients of the overall transfer function:');
disp(a);
poles_overall = roots(a);
zeros_overall = roots(b);
figure;
zplane(zeros_overall, poles_overall);
title('Poles and Zeros of the Overall Transfer Function in the z-plane');
xlabel('Real Part');
ylabel('Imaginary Part');
grid on;
figure;
subplot(2,1,1);
stem(0:length(b)-1, b, 'filled');
title('Direct Form I Structure - Numerator');
xlabel('n');
ylabel('b[n]');
grid on;
subplot(2,1,2);
stem(0:length(a)-1, a, 'filled');
title('Direct Form I Structure - Denominator');
xlabel('n');
ylabel('a[n]');
grid on;
figure;
len_b = length(b);
len_a = length(a);
if len_b > len_a
    a = [a, zeros(1, len_b - len_a)];
elseif len_a > len_b
    b = [b, zeros(1, len_a - len_b)];
end
stem(0:length(b)-1, b, 'filled');
hold on;
stem(0:length(a)-1, -a, 'filled');
hold off;
title('Direct Form II Structure');
xlabel('n');
ylabel('Coefficients');
grid on;
```

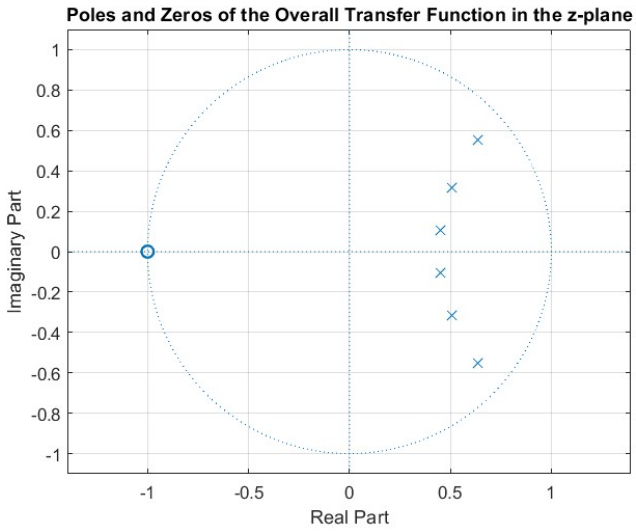
Output:

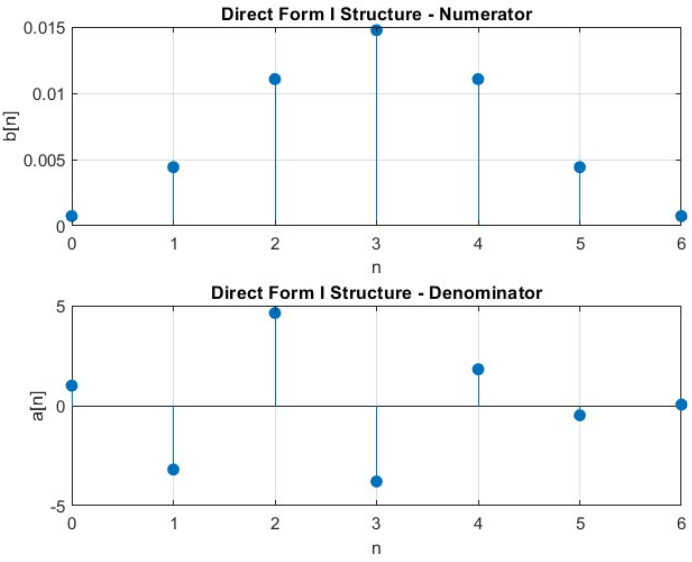
Numerator coefficients of the overall transfer function:

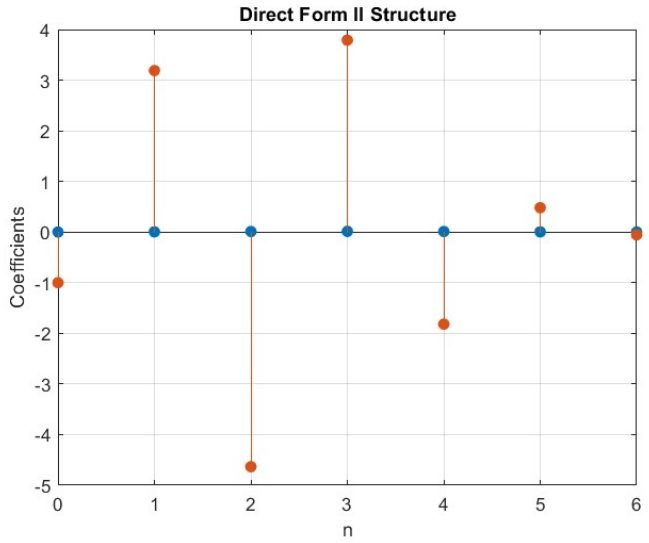
0.0007 0.0044 0.0111 0.0148 0.0111 0.0044 0.0007

Denominator coefficients of the overall transfer function:

1.0000 -3.1890 4.6340 -3.7907 1.8185 -0.4808 0.0544







Analysis and Conclusion:

The poles and zeros of the three second order sections are obtained. The poles and zeros are plotted in the z-plane. The Direct Form I of both numerator and the denominator are plotted. The Direct Form II structure of the sections are plotted. The numerator coefficients of the overall transfer function: 0.0007 0.0044 0.0111 0.0148 0.0111 0.0044 0.0007

The denominator coefficients of the overall transfer function: 1.0000 -3.1890 4.6340 -3.7907 1.8185 -0.4808 0.0544

LAB 5:

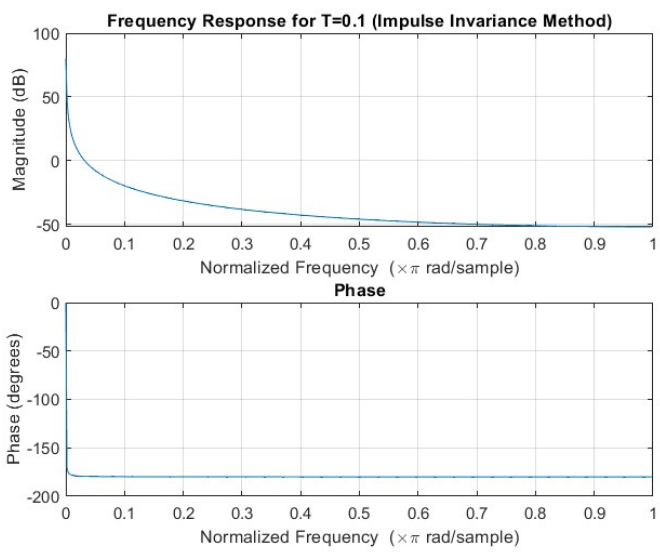
Problem 1:

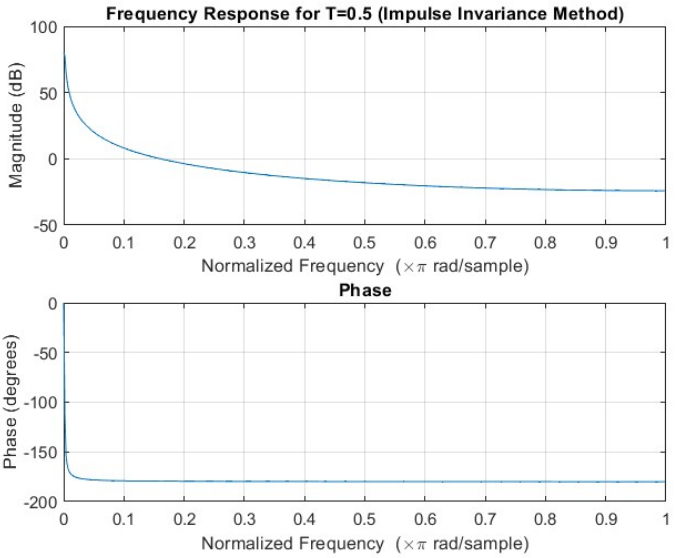
Convert the analog filter $H(s) = \frac{s+0.1}{(s+0.1)^2 + 9}$ in to a digital IIR filter by means of the impulse invariance method. Plot the frequency response (magnitude) of the designed filter taking sampling interval (T) of 0.1, 0.5 seconds. Compare the response of the filter designed to that of the analog one. Comment on the effect of T on the response.

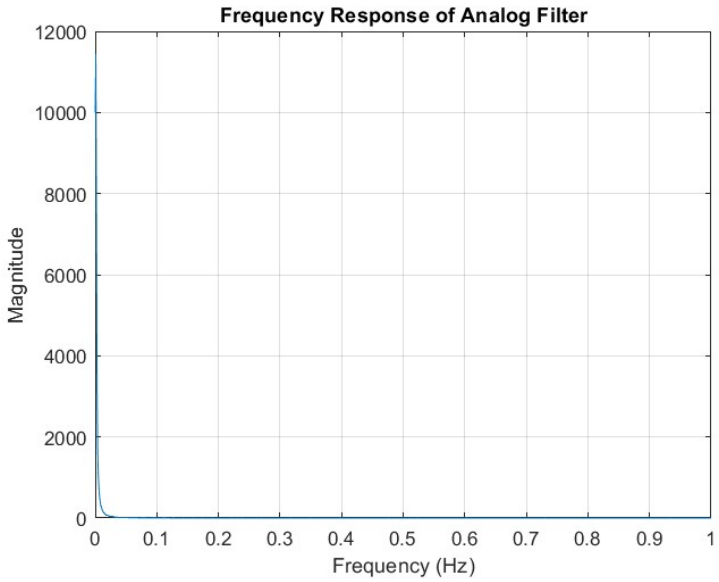
Code:

```
syms s
Hs = s / (s^2 + 0.01*s + 0.09);
T1 = 0.1;
T2 = 0.5;
[b1, a1] = impinvar([0 1], [1 0.01 0.01^2], 1/T1);
[b2, a2] = impinvar([0 1], [1 0.01 0.01^2], 1/T2);
figure;
freqz(b1, a1);
title('Frequency Response for T=0.1 (Impulse Invariance Method)');
figure;
freqz(b2, a2);
title('Frequency Response for T=0.5 (Impulse Invariance Method)');
[H, w] = freqs([0 1], [1 0.01 0.01^2], linspace(0, 2*pi, 1024));
figure;
plot(w/(2*pi), abs(H));
title('Frequency Response of Analog Filter');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

Output:







Analysis and Conclusion:

The given analog filter H(s) is converted to infinite impulse response (IIR) by the method of impulse invariance method. The magnitude and phase response of the designed filter by using impulse invariance method is plotted taking T at 0.1 and 0.5. The frequency response of the analog filter is plotted. For lower value of T the magnitude is less compared to higher value of T.

Problem 2:

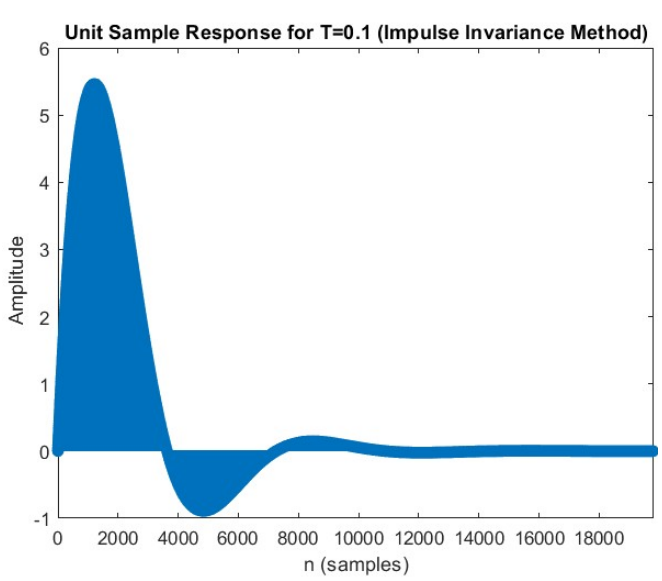
Compare the unit sample response of the designed digital IIR filter with the impulse response of analog filter for T=0.1 and 0.5.

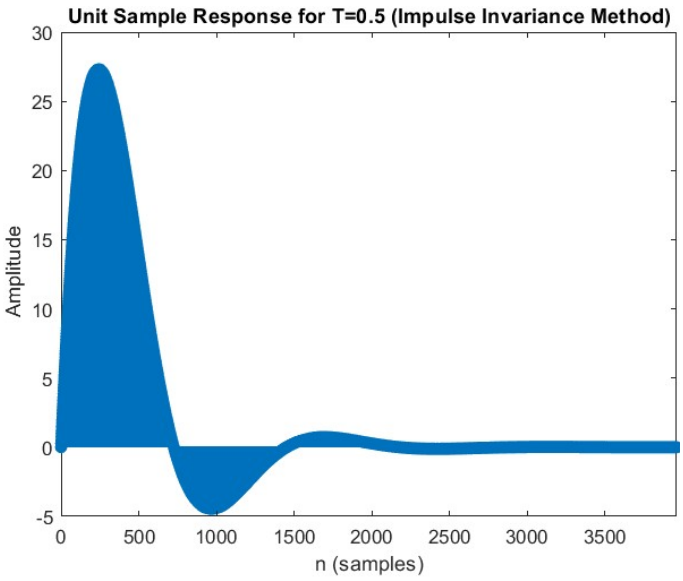
Code:

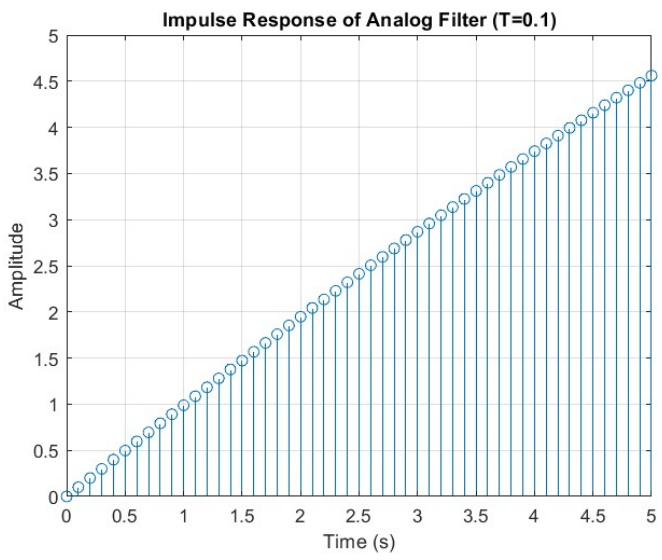
```
syms s
Hs = s / (s^2 + 0.01*s + 0.09);
T1 = 0.1;
T2 = 0.5;
[b1, a1] = impinvar([0 1], [1 0.01 0.01^2], 1/T1);
[b2, a2] = impinvar([0 1], [1 0.01 0.01^2], 1/T2);
figure;
impz(b1, a1);
title('Unit Sample Response for T=0.1 (Impulse Invariance Method)');
figure;
impz(b2, a2);
title('Unit Sample Response for T=0.5 (Impulse Invariance Method)');
t1 = 0:T1:5;
t2 = 0:T2:5;
h_analog = @(t) exp(-0.01*t) .* (sin(0.1*t) / 0.1);
figure;
stem(t1, h_analog(t1));
title('Impulse Response of Analog Filter (T=0.1)');
xlabel('Time (s)');
```

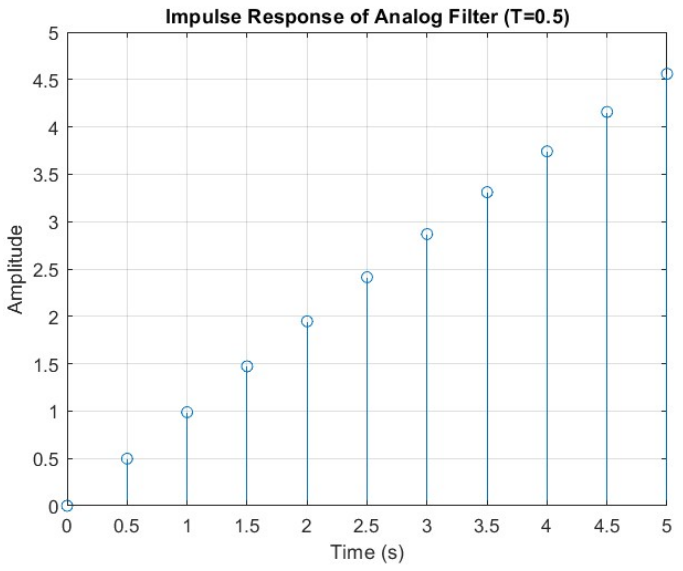
```
ylabel('Amplitude');
grid on;
figure;
stem(t2, h_analog(t2));
title('Impulse Response of Analog Filter (T=0.5)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```

Output:









Analysis and Conclusion:

The unit sample response of the designed IIR filters are plotted for T at 0.1 and 0.5. For T at 0.1 the amplitude reaches peak at 2000 samples while at T 0.5 the amplitude reaches peak at 500 samples. Similarly, the impulse response of the analog IIR filter is plotted at T 0.1 and 0.5.

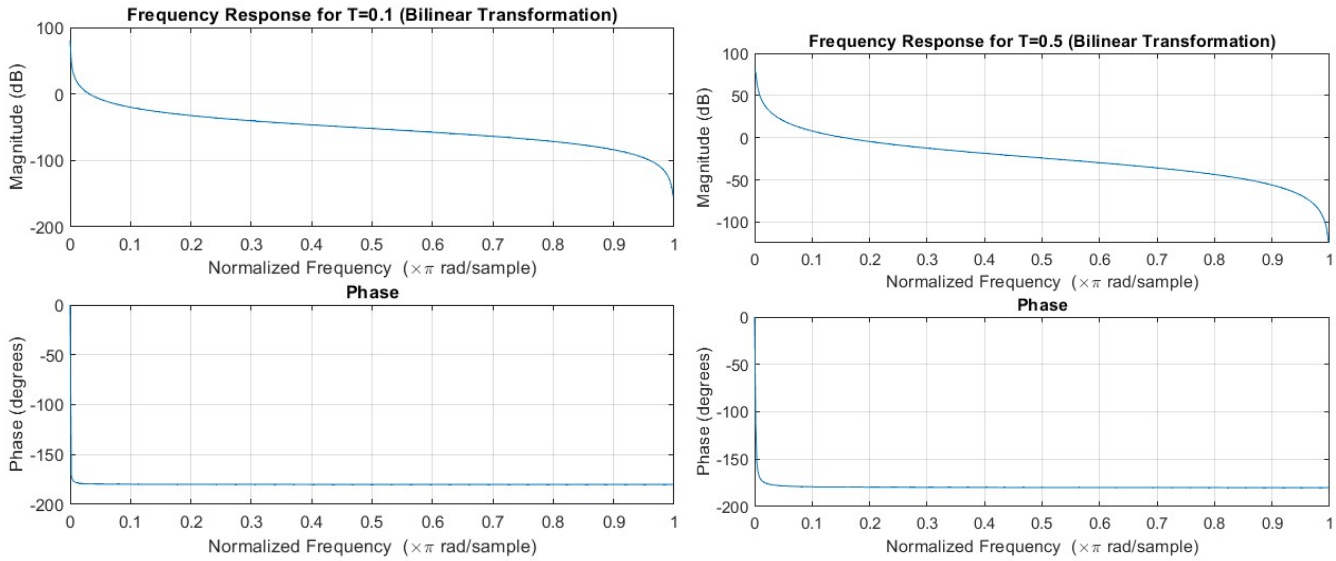
Problem 3:

Convert the above analog filter in to a digital IIR filter by means of bilinear transformation and repeat all the procedures as specified in 1.1.

Code:

```
syms s
Hs = s / (s^2 + 0.01*s + 0.09);
T1 = 0.1;
T2 = 0.5;
[b3, a3] = bilinear([0 1], [1 0.01 0.01^2], 1/T1);
[b4, a4] = bilinear([0 1], [1 0.01 0.01^2], 1/T2);
figure;
freqz(b3, a3);
title('Frequency Response for T=0.1 (Bilinear Transformation)');
figure;
freqz(b4, a4);
title('Frequency Response for T=0.5 (Bilinear Transformation)');
```

Output:



Analysis and Conclusion:

The above analog filter given in Problem 1 is converted to a digital IIR filter by the means of bilinear transformation. The magnitude response and phase response of the converted filter is plotted as T 0.1 and 0.5. The magnitude of IIR filter at T 0.1 reaches zero faster than that of T 0.5.

Problem 4:

An IIR digital low pass filter is required to meet the following specifications:
Pass band ripple (or peak to peak ripple): 0.5 dB
Passband edge: 1.2 kHz
Stopband attenuation: 40 dB
Stopband edge: 2.0 kHz Sample rate: 8.0 kHz
Use the MATLAB Signal Processing Toolbox functions to determine

- **The required filter order,**
- **The cutoff frequency,**
- **The numerator and the denominator coefficients**

for the digital Butterwoth, digital Chebyshev and digital Elliptic filters. Also plot their frequency responses. Describe the nature of each response.

Code:

```
Fs = 8000;
Wp = 1200 / (Fs / 2);
Ws = 2000 / (Fs / 2);
Rp = 0.5;
Rs = 40;
[n_butter, Wn_butter] = buttord(Wp, Ws, Rp, Rs);
[b_butter, a_butter] = butter(n_butter, Wn_butter);
[n_cheby1, Wn_cheby1] = cheb1ord(Wp, Ws, Rp, Rs);
[b_cheby1, a_cheby1] = cheby1(n_cheby1, Rp, Wn_cheby1);
[n_ellip, Wn_ellip] = ellipord(Wp, Ws, Rp, Rs);
[b_ellip, a_ellip] = ellip(n_ellip, Rp, Rs, Wn_ellip);
figure;
freqz(b_butter, a_butter, 1024, Fs);
title('Butterworth Filter Frequency Response');
figure;
freqz(b_cheby1, a_cheby1, 1024, Fs);
title('Chebyshev Type I Filter Frequency Response');
figure;
freqz(b_ellip, a_ellip, 1024, Fs);
title('Elliptic Filter Frequency Response');
fprintf('Butterworth Filter:\n');
fprintf('Order: %d\n', n_butter);
fprintf('Numerator Coefficients: \n');
disp(b_butter);
fprintf('Denominator Coefficients: \n');
disp(a_butter);
fprintf('\nChebyshev Type I Filter:\n');
fprintf('Order: %d\n', n_cheby1);
fprintf('Numerator Coefficients: \n');
disp(b_cheby1);
fprintf('Denominator Coefficients: \n');
disp(a_cheby1);
fprintf('\nElliptic Filter:\n');
fprintf('Order: %d\n', n_ellip);
fprintf('Numerator Coefficients: \n');
```

```
disp(b_ellip);
fprintf('Denominator Coefficients: \n');
disp(a_ellip);
```

Output:

1. For Butterworth Filter:

   Order: 9

   Numerator Coefficients:

   0.0004 0.0032 0.0129 0.0302 0.0453 0.0453 0.0302 0.0129 0.0032 0.0004

   Denominator Coefficients:

   1.0000 -2.7996 4.4582 -4.5412 3.2404 -1.6330 0.5780 -0.1370 0.0197 -0.0013

2. For Chebyshev Type I Filter:

   Order: 5

   Numerator Coefficients:

   0.0026 0.0132 0.0264 0.0264 0.0132 0.0026

   Denominator Coefficients:

   1.0000 -2.9775 4.2932 -3.5124 1.6145 -0.3334

3. For Elliptic Filter:

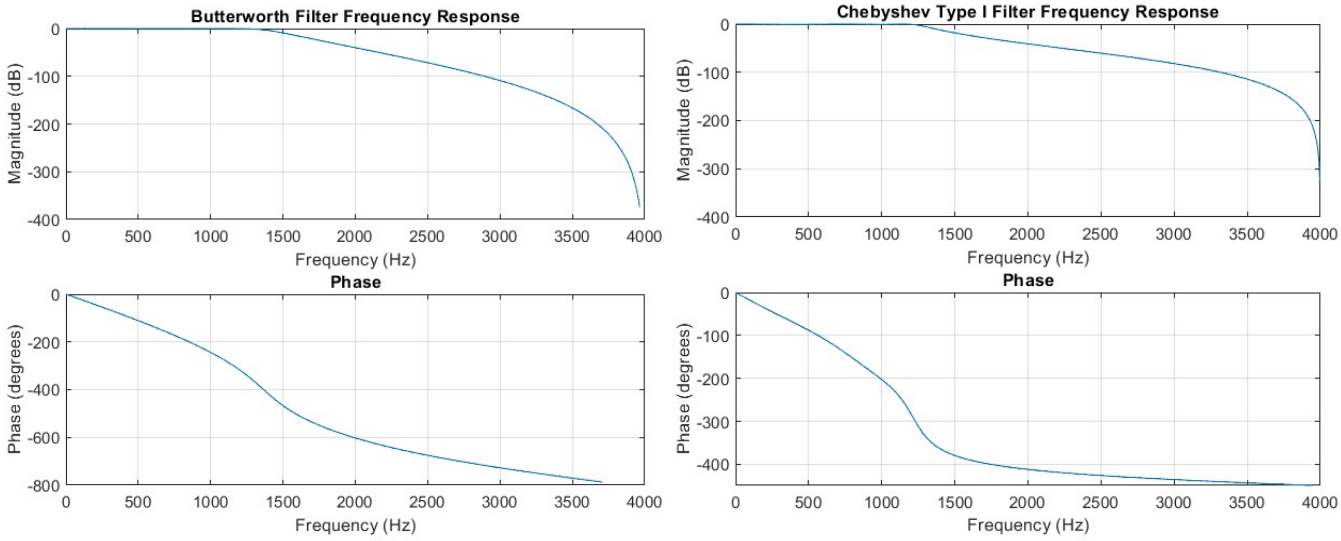
   Order: 4

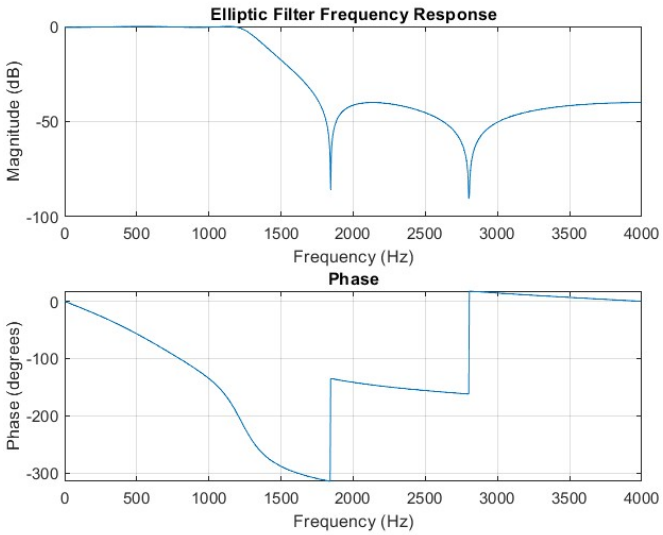
   Numerator Coefficients:

   0.0389 0.0363 0.0665 0.0363 0.0389

   Denominator Coefficients:

   1.0000 -2.1444 2.3658 -1.3250 0.3332





Analysis and Conclusion:

The above code was used to obtain digital IIR filter with given specification. For these specification digital Butterworth filter, digital Chebyshev and digital Elliptical filters were obtained. The required filter order, cutoff frequency and numerator and denominator coefficient were obtained for each filter. The magnitude and phase response of each filter was plotted.

LAB 6:

Problem 1:

Design an FIR linear phase digital filter approximating the ideal frequency response

H(w)= 1, for w /6

0, for /6 w

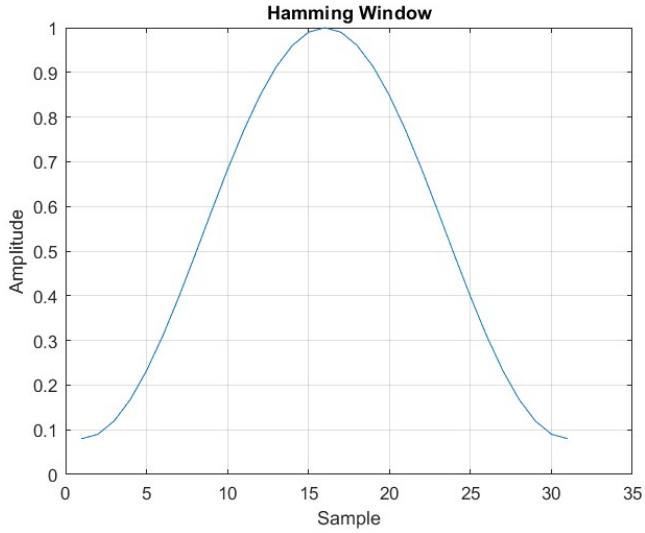
a. Plot the window function for Hamming and its frequency response for length of M=31.
b. Using the Hamming window plot the frequency response of the truncated FIR filter.
c. Repeat parts (a) and (b) for the Hanning, Blackman and Bartlett windows.
d. Repeat parts (a), (b) and (d) for filter length of M=61.

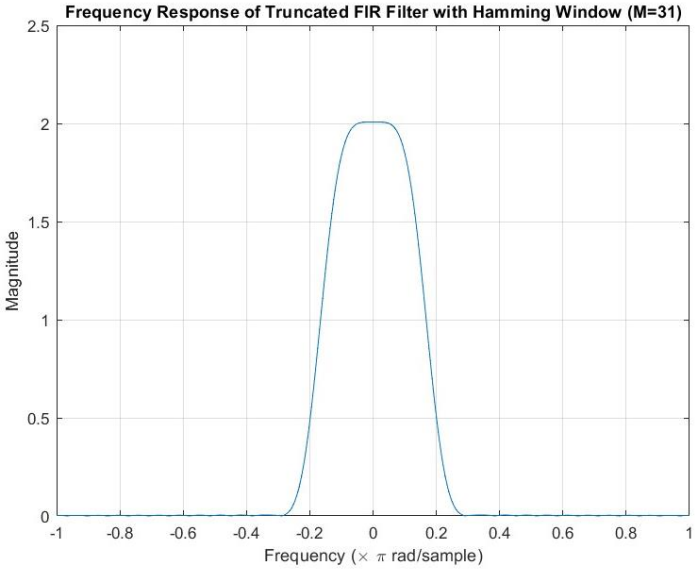
Code:

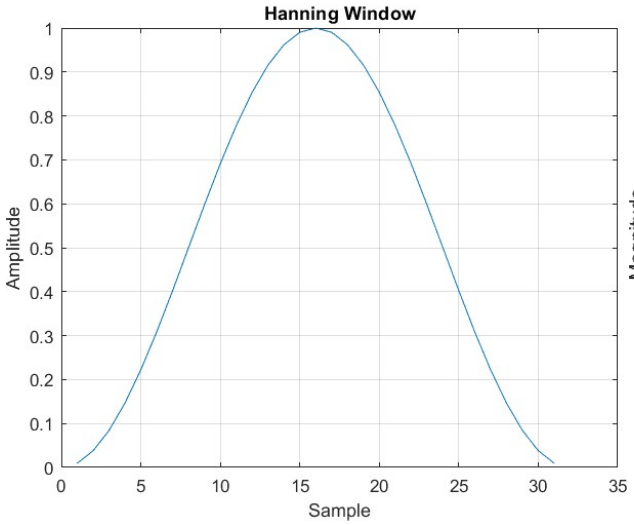
```
M = 31;
N = 1024;
omega = linspace(-pi, pi, N);
H_ideal = double(abs(omega) <= pi/6);
window_hamming = hamming(M);
n = -floor(M/2):floor(M/2);
impulse_response = (2/pi) * sin(pi * n / 6) ./ n;
impulse_response(floor(M/2)+1) = 2/pi * pi/6;
hamming_impulse_response = impulse_response .* window_hamming';
H_truncated_hamming = abs(fftshift(fft(hamming_impulse_response, N)));
figure;
plot(linspace(-pi, pi, N)/pi, H_truncated_hamming);
title('Frequency Response of Truncated FIR Filter with Hamming Window
(M=31)');
xlabel('Frequency (\times \pi rad/sample)');
ylabel('Magnitude');
grid on;
window_hanning = hanning(M);
hanning_impulse_response = impulse_response .* window_hanning';
H_truncated_hanning = abs(fftshift(fft(hanning_impulse_response, N)));
window_blackman = blackman(M);
blackman_impulse_response = impulse_response .* window_blackman';
H_truncated_blackman = abs(fftshift(fft(blackman_impulse_response, N)));
window_bartlett = bartlett(M);
bartlett_impulse_response = impulse_response .* window_bartlett';
H_truncated_bartlett = abs(fftshift(fft(bartlett_impulse_response, N)));
figure;
plot(window_hanning);
title('Hanning Window');
xlabel('Sample');
ylabel('Amplitude');
grid on;
figure;
plot(linspace(-pi, pi, N)/pi, H_truncated_hanning);
title('Frequency Response of Truncated FIR Filter with Hanning Window
(M=31)');
xlabel('Frequency (\times \pi rad/sample)');
ylabel('Magnitude');
grid on;
```

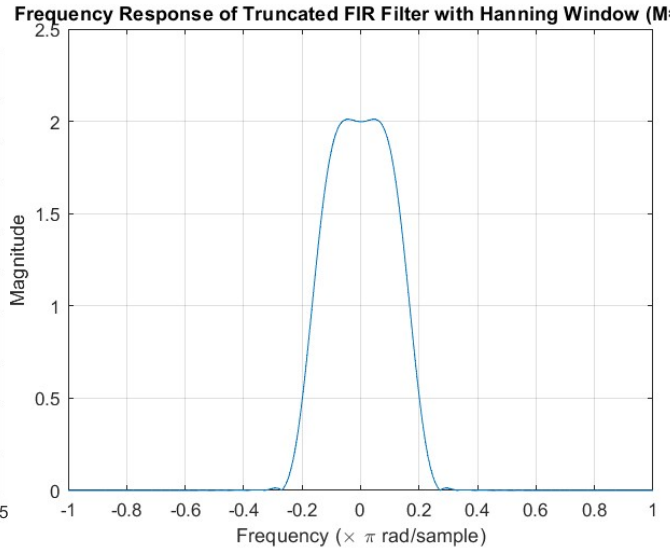
```
figure;
plot(window_blackman);
title('Blackman Window');
xlabel('Sample');
ylabel('Amplitude');
grid on;
figure;
plot(linspace(-pi, pi, N)/pi, H_truncated_blackman);
title('Frequency Response of Truncated FIR Filter with Blackman Window
(M=31)');
xlabel('Frequency (\times \pi rad/sample)');
ylabel('Magnitude');
grid on;
figure;
plot(window_bartlett);
title('Bartlett Window');
xlabel('Sample');
ylabel('Amplitude');
grid on;
figure;
plot(linspace(-pi, pi, N)/pi, H_truncated_bartlett);
title('Frequency Response of Truncated FIR Filter with Bartlett Window
(M=31)');
xlabel('Frequency (\times \pi rad/sample)');
ylabel('Magnitude');
grid on;
M = 61;
N = 1024;
omega = linspace(-pi, pi, N);
H_ideal = double(abs(omega) <= pi/6);
n = -floor(M/2):floor(M/2);
impulse_response = (2/pi) * sin(pi * n / 6) ./ n;
impulse_response(floor(M/2)+1) = 2/pi * pi/6;
window_hamming = hamming(M);
hamming_impulse_response = impulse_response .* window_hamming';
H_truncated_hamming = abs(fftshift(fft(hamming_impulse_response, N)));
figure;
plot(window_hamming);
title('Hamming Window (M=61)');
xlabel('Sample');
ylabel('Amplitude');
grid on;
figure;
plot(linspace(-pi, pi, N)/pi, H_truncated_hamming);
title('Frequency Response of Truncated FIR Filter with Hamming Window
(M=61)');
xlabel('Frequency (\times \pi rad/sample)');
ylabel('Magnitude');
grid on;
```

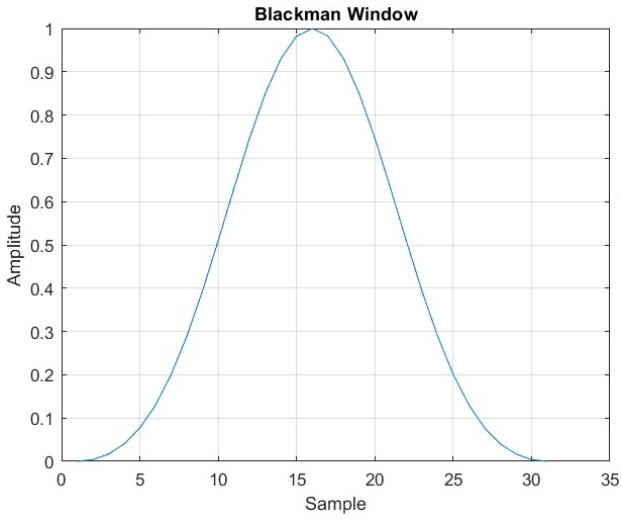
Output:

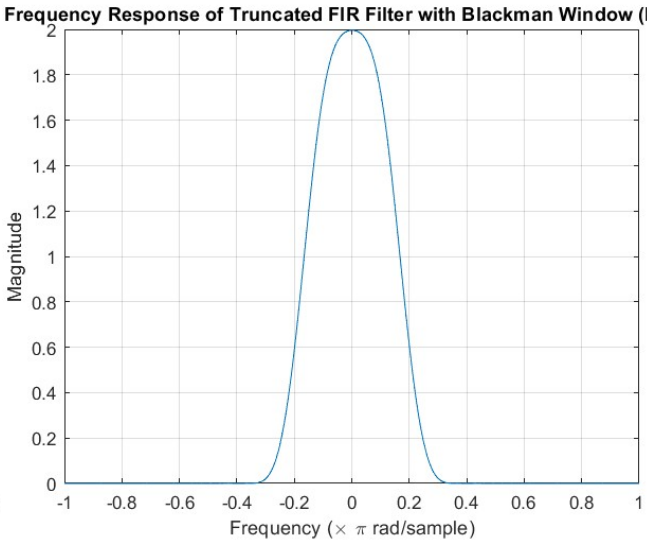

Hamming Window
Amplitude
Sample

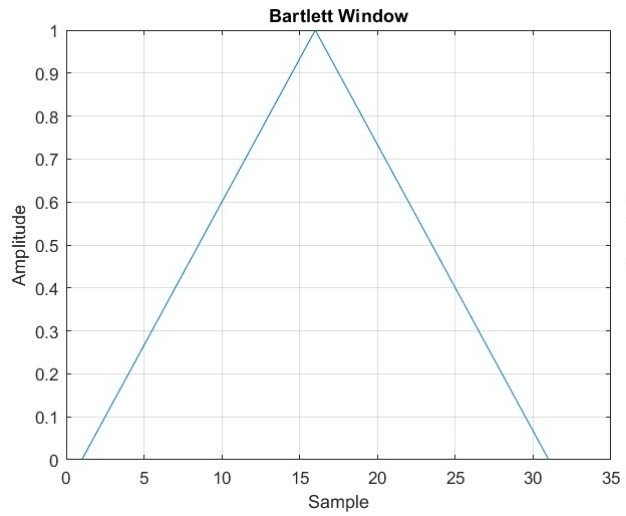

Frequency Response of Truncated FIR Filter with Hamming Window (M=31)
Magnitude
Frequency (× π rad/sample)

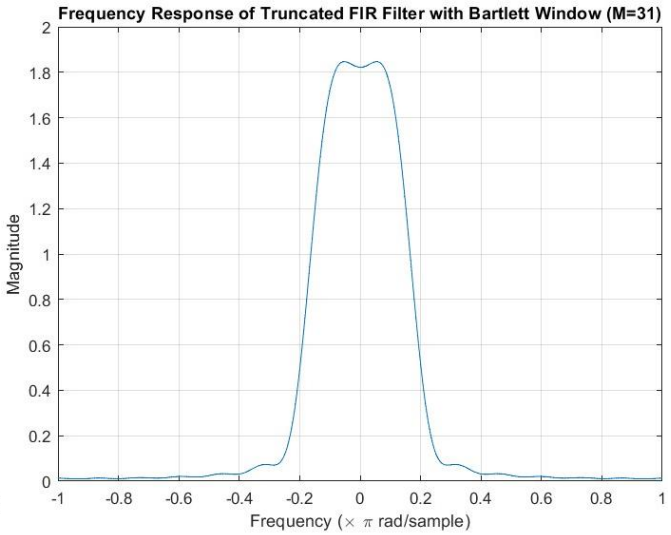

Hanning Window
Amplitude
Sample

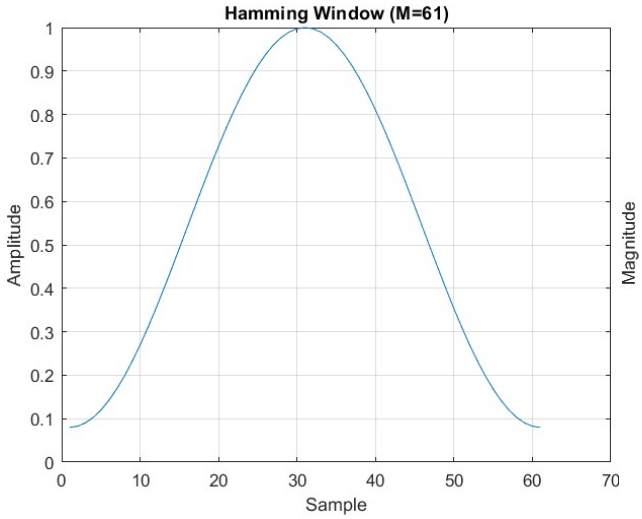

Frequency Response of Truncated FIR Filter with Hanning Window (M=31)
Magnitude
Frequency (× π rad/sample)

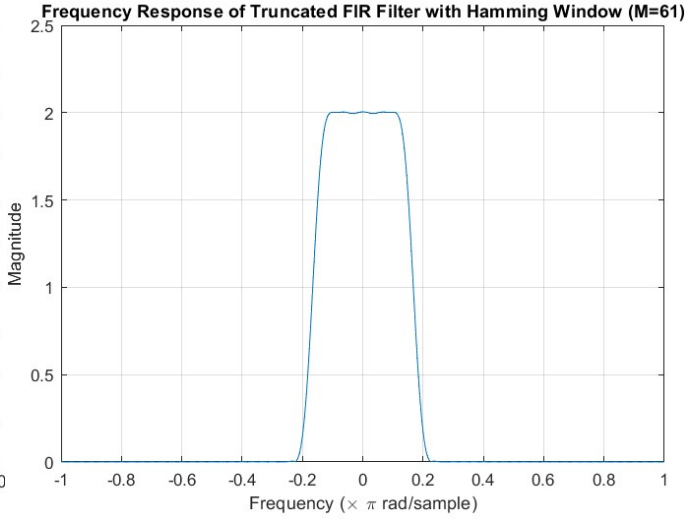

Blackman Window
Amplitude
Sample


Frequency Response of Truncated FIR Filter with Blackman Window (M=31)
Magnitude
Frequency (× π rad/sample)









Analysis And Conclusion:

Using above code, we can design FIR filter for given ideal frequency response. The window function for Hamming, Hanning, Blackman and Bartlett windows are plotted. The frequency response for each of these window functions can are also plotted for M = 31. The window function for Hamming window and its frequency response are plotted for M = 31.

Problem 2:

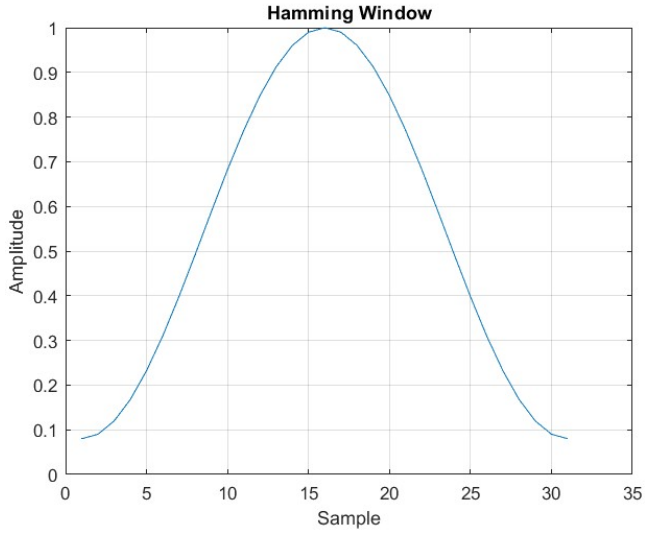
Discuss the effects of different types of the windowing functions on the frequency response of the FIR filter. Carry out the comparative study on the basis of the peak side lobe level, the approximate transition width of the main lobe etc. Also discuss on the effects of increasing value of M.

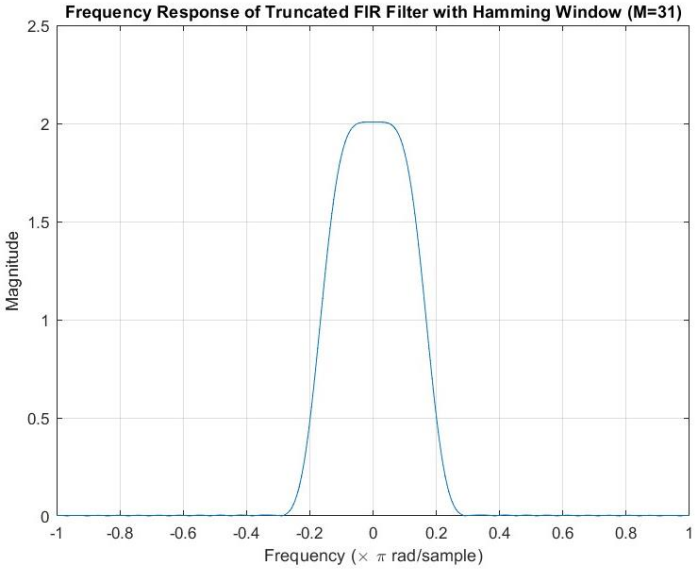
Code:

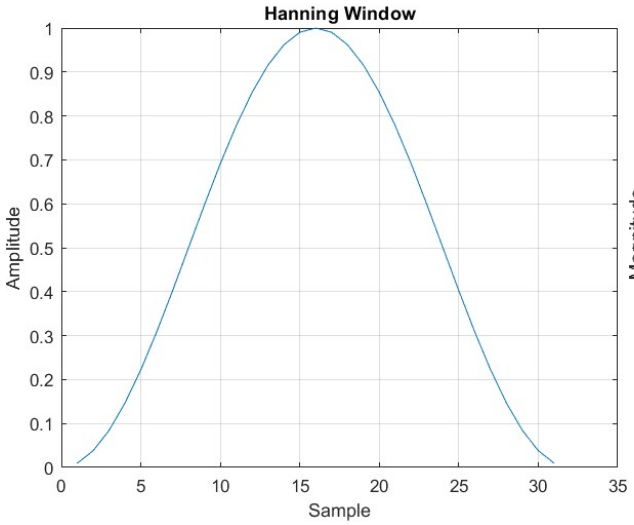
```
M = 31;
N = 1024;
omega = linspace(-pi, pi, N);
H_ideal = double(abs(omega) <= pi/6);
window_hamming = hamming(M);
n = -floor(M/2):floor(M/2);
impulse_response = (2/pi) * sin(pi * n / 6) ./ n;
impulse_response(floor(M/2)+1) = 2/pi * pi/6;
hamming_impulse_response = impulse_response .* window_hamming';
H_truncated_hamming = abs(fftshift(fft(hamming_impulse_response, N)));
figure;
plot(linspace(-pi, pi, N)/pi, H_truncated_hamming);
title('Frequency Response of Truncated FIR Filter with Hamming Window
(M=31)');
xlabel('Frequency (\times \pi rad/sample)');
ylabel('Magnitude');
grid on;
window_hanning = hanning(M);
hanning_impulse_response = impulse_response .* window_hanning';
H_truncated_hanning = abs(fftshift(fft(hanning_impulse_response, N)));
window_blackman = blackman(M);
blackman_impulse_response = impulse_response .* window_blackman';
H_truncated_blackman = abs(fftshift(fft(blackman_impulse_response, N)));
window_bartlett = bartlett(M);
bartlett_impulse_response = impulse_response .* window_bartlett';
H_truncated_bartlett = abs(fftshift(fft(bartlett_impulse_response, N)));
figure;
plot(window_hanning);
title('Hanning Window');
xlabel('Sample');
ylabel('Amplitude');
grid on;
figure;
plot(linspace(-pi, pi, N)/pi, H_truncated_hanning);
title('Frequency Response of Truncated FIR Filter with Hanning Window
(M=31)');
xlabel('Frequency (\times \pi rad/sample)');
ylabel('Magnitude');
grid on;
figure;
plot(window_blackman);
title('Blackman Window');
```

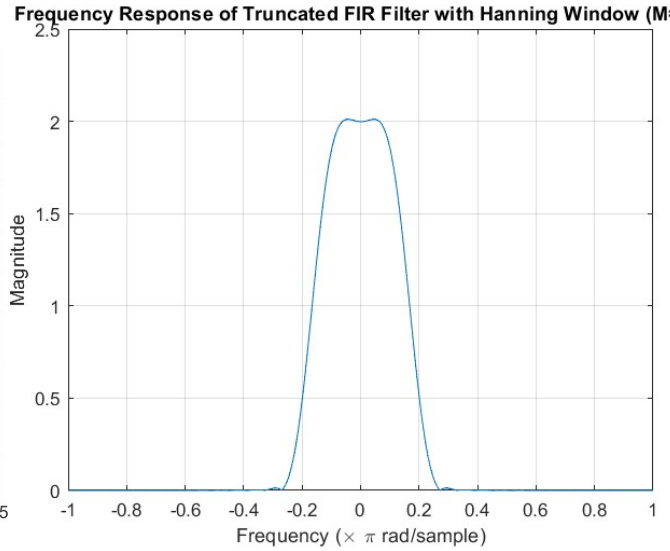
```
xlabel('Sample');
ylabel('Amplitude');
grid on;
figure;
plot(linspace(-pi, pi, N)/pi, H_truncated_blackman);
title('Frequency Response of Truncated FIR Filter with Blackman Window
(M=31)');
xlabel('Frequency (\times \pi rad/sample)');
ylabel('Magnitude');
grid on;
figure;
plot(window_bartlett);
title('Bartlett Window');
xlabel('Sample');
ylabel('Amplitude');
grid on;
figure;
plot(linspace(-pi, pi, N)/pi, H_truncated_bartlett);
title('Frequency Response of Truncated FIR Filter with Bartlett Window
(M=31)');
xlabel('Frequency (\times \pi rad/sample)');
ylabel('Magnitude');
grid on;
M = 61;
N = 1024;
omega = linspace(-pi, pi, N);
H_ideal = double(abs(omega) <= pi/6);
n = -floor(M/2):floor(M/2);
impulse_response = (2/pi) * sin(pi * n / 6) ./ n;
impulse_response(floor(M/2)+1) = 2/pi * pi/6;
window_hamming = hamming(M);
hamming_impulse_response = impulse_response .* window_hamming';
H_truncated_hamming = abs(fftshift(fft(hamming_impulse_response, N)));
figure;
plot(window_hamming);
title('Hamming Window (M=61)');
xlabel('Sample');
ylabel('Amplitude');
grid on;
figure;
plot(linspace(-pi, pi, N)/pi, H_truncated_hamming);
title('Frequency Response of Truncated FIR Filter with Hamming Window
(M=61)');
xlabel('Frequency (\times \pi rad/sample)');
ylabel('Magnitude');
grid on;
```

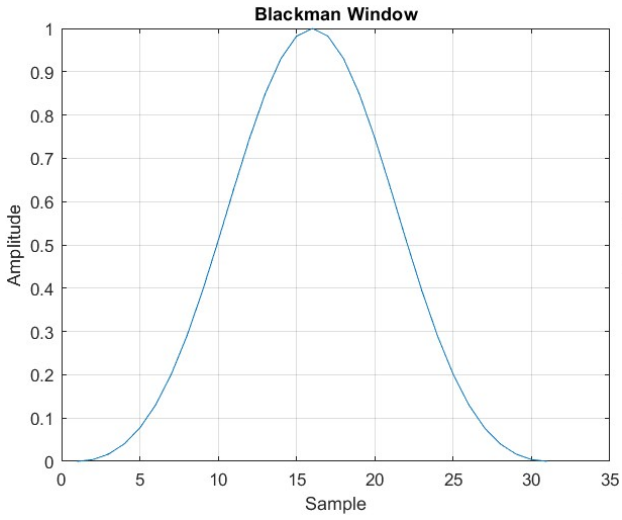
Output:

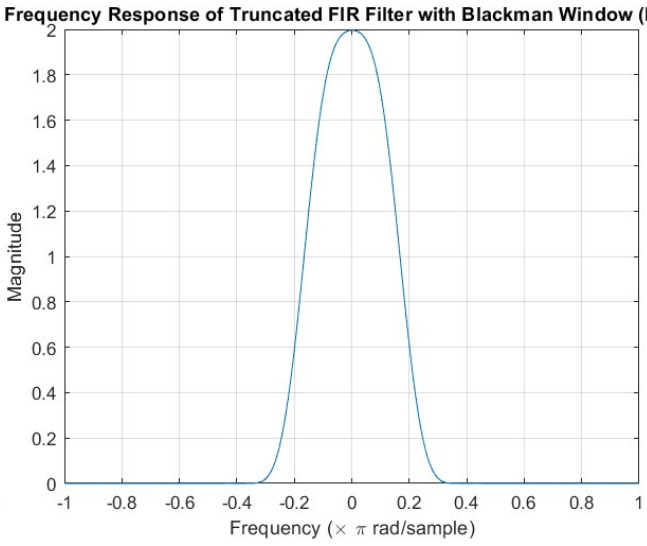

Hamming Window
Amplitude
Sample

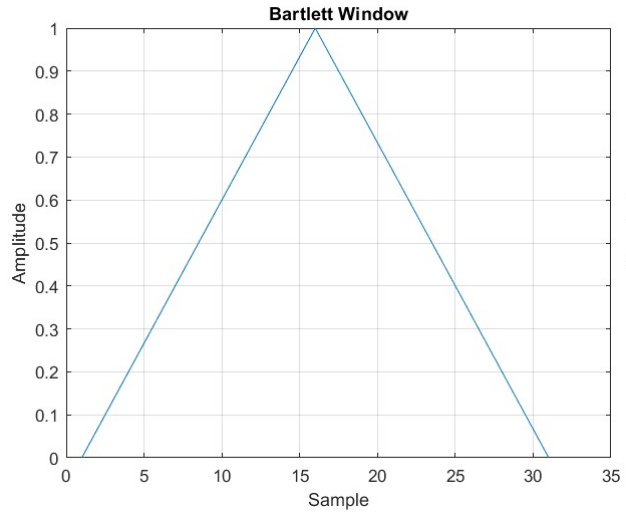

Frequency Response of Truncated FIR Filter with Hamming Window (M=31)
Magnitude
Frequency (× π rad/sample)

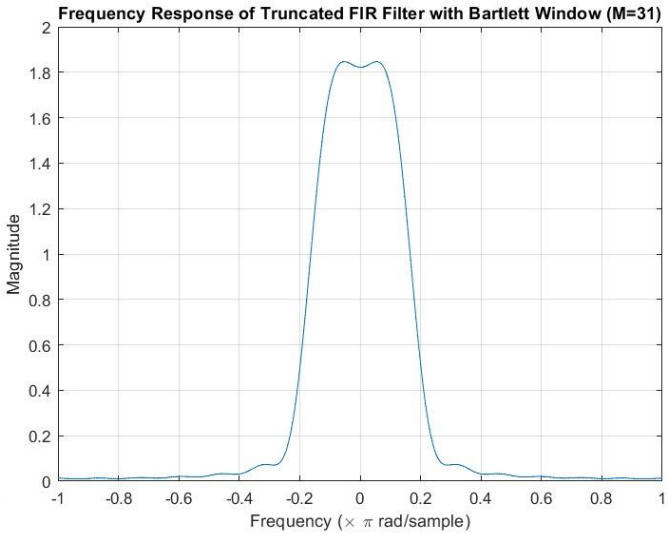

Hanning Window
Amplitude
Sample

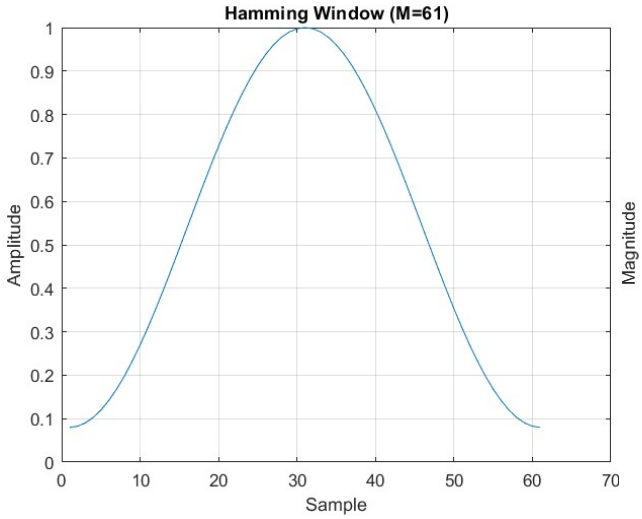

Frequency Response of Truncated FIR Filter with Hanning Window (M=31)
Magnitude
Frequency (× π rad/sample)

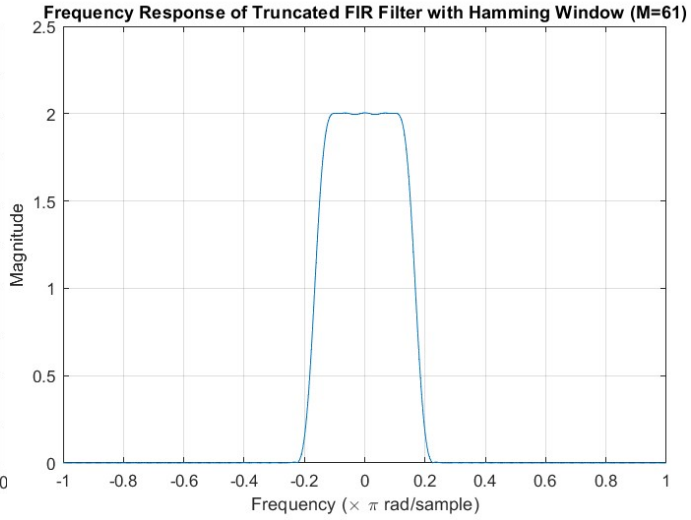

Blackman Window
Amplitude
Sample


Frequency Response of Truncated FIR Filter with Blackman Window (M=31)
Magnitude
Frequency (× π rad/sample)







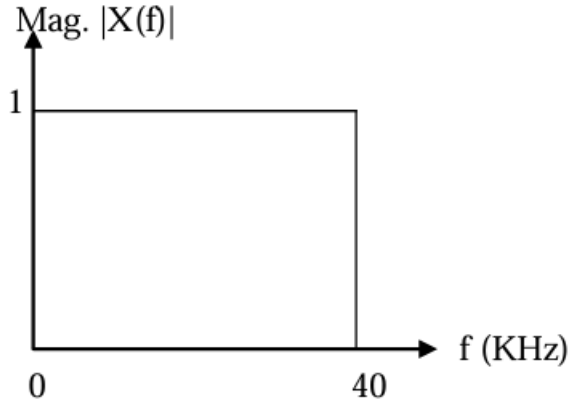


Analysis and Conclusion:

The frequency response of FIR filter depends on the type of window used and the value of M used. The peak of frequency response of Bartlett window is less than that of Hamming, Hanning and Blackman window. The approximate transition width of Blackman window is less than that of other windows. On increasing the value of M the window gets wider hence the frequency response transition width gets larger until it becomes as a rectangular shape.

Problem 3:

An analog signal x(t) consists of the sum of two components x1(t) and x2(t). The spectral characteristics of x(t) is shown in fig 2.1. The signal x(t) is band limited to 40kHz and is sampled at the rate of 100kHz to yield the sequence x(n).



It is desired to suppress the signal x2(t) by passing the sequence x(n) through a digital low pass filter. The allowable distortion on X1(f) is +2% (δ1=0.02) over the range 0 <= F <= 15 kHz. Above 20 kHz, the filter must have an attenuation of at least 40 dB (δ2=0.01).

a. For obtaining the filter with above specifications, use the Kaiser window and determine the length of the required window. Plot the frequency response of the filter and its impulse response also.

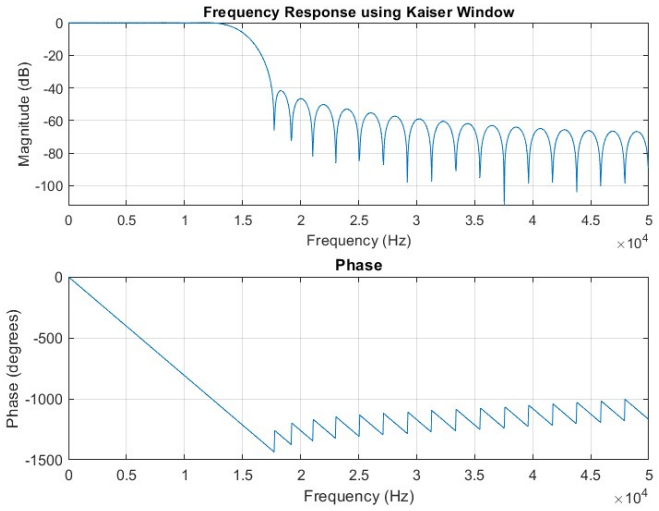
b. If the same filter is to be designed using Hamming window what would be the length of the required window. Plot the frequency and impulse response of the filter.

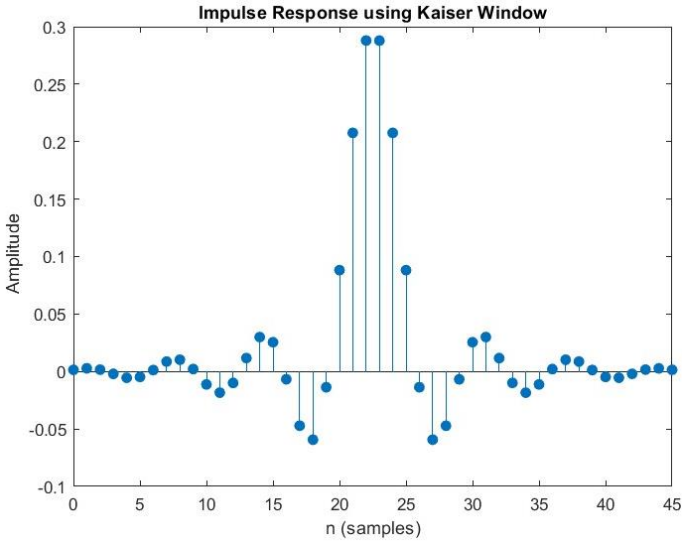
Code:

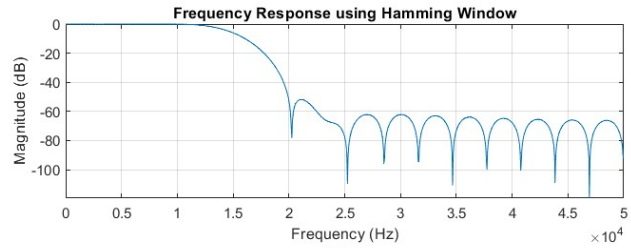
```
fs = 100e3;
fp = 15e3;
fsb = 20e3;
delta_p = 0.02;
delta_s = 0.01;
wp = fp / (fs / 2);
ws = fsb / (fs / 2);
delta_f = fsb - fp;
A = -20 * log10(delta_s);
if A > 50
    beta = 0.1102 * (A - 8.7);
elseif A >= 21
    beta = 0.5842 * (A - 21)^0.4 + 0.07886 * (A - 21);
else
    beta = 0;
end
N = ceil((A - 8) / (2.285 * (2 * pi * delta_f / fs)));
if mod(N, 2) == 0
    N = N + 1;
end
b = fir1(N, wp, kaiser(N + 1, beta));
freqz(b, 1, 1024, fs);
title('Frequency Response using Kaiser Window');
```

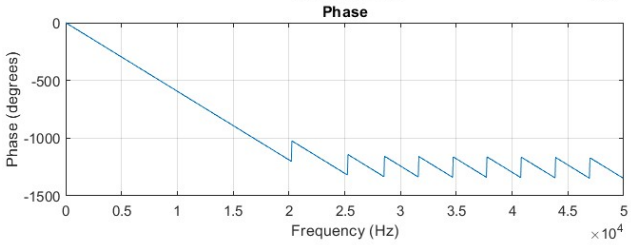
```
figure;
impz(b, 1);
title('Impulse Response using Kaiser Window');

fs = 100e3;
fp = 15e3;
fsb = 20e3;
delta_f = fsb - fp;
wp = fp / (fs / 2);
N_hamming = ceil(3.3 * pi / (2 * pi * delta_f / fs));
if mod(N_hamming, 2) == 0
    N_hamming = N_hamming + 1;
end
b_hamming = fir1(N_hamming, wp, hamming(N_hamming + 1));
freqz(b_hamming, 1, 1024, fs);
title('Frequency Response using Hamming Window');
figure;
impz(b_hamming, 1);
title('Impulse Response using Hamming Window');
```

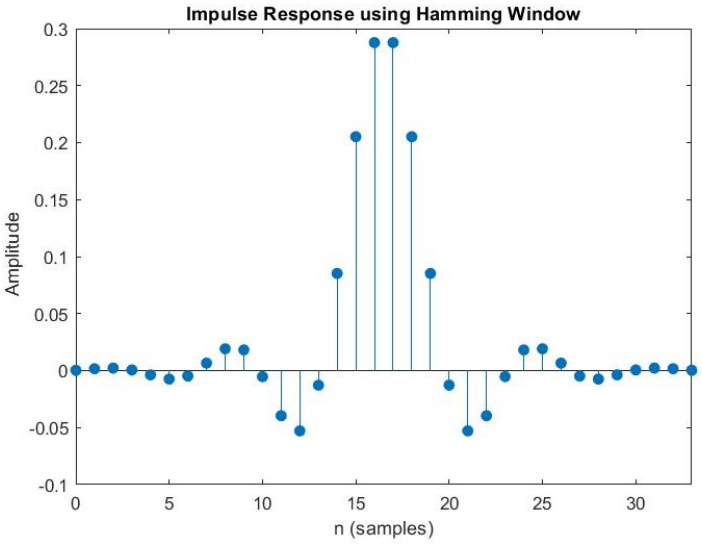
Output:











Analysis and Conclusion:

The code above is used to obtain the required digital low pass filter with given range and attenuation. Two types of windows are used: Kaiser Window and Hamming Window. Using each window, a digital low pass filter is obtained. The plot of magnitude response and phase response for each window are plotted. Also the impulse response using both Kaiser window and Hamming window is also plotted.