AM5510 : BIOMEDICAL SIGNALS & SYSTEMS

Programming Assignment #1: Convolution & Digital Filtering

AM23M022
DINESH KUMAR M

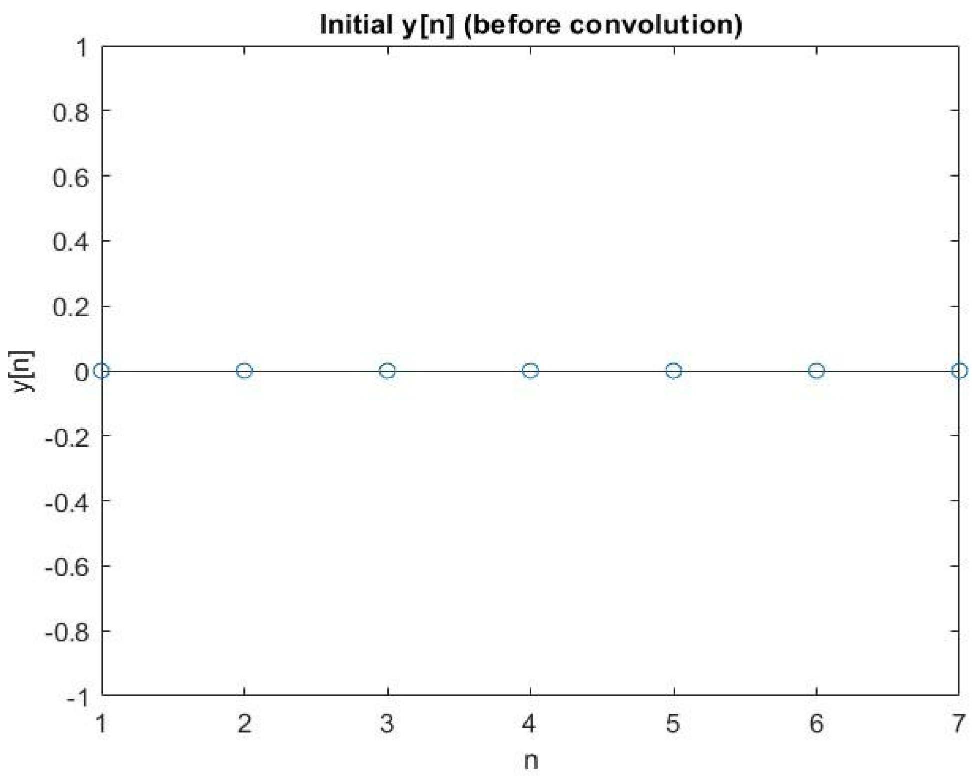
1. Write a program routine to calculate the convolution sum below. Your program should take two arrays of length N and M containing input signal values x[n], and the impulse response values, h[n], respectively. Test your program using simple functions for x[n] and h[n]. Graphically display your x[n], h[n] and y[n]. Plot the i/p and o/p in the same screen/ subplots.

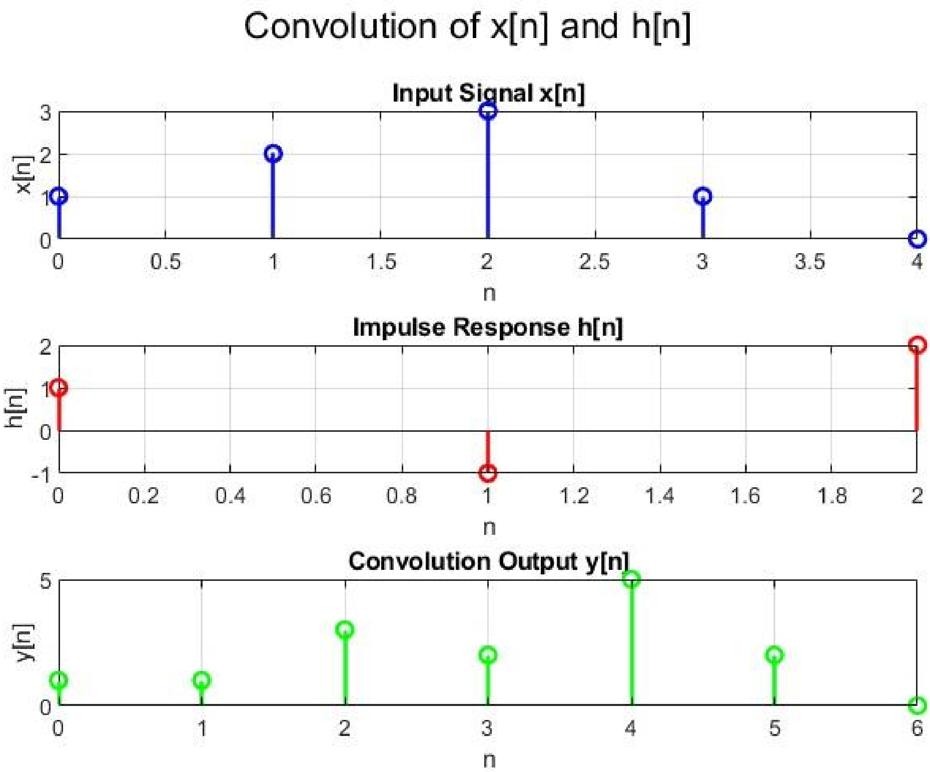
$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

DISCRETE CONVOLUTION

```
% Convolution of x[n] and h[n]

% Defining the lengths of signals
N = 5;
M = 3;

% Defining the input signal x[n]
input_signal = [1, 2, 3, 1, 0];

% Defining the impulse response h[n]
impulse_response = [1, -1, 2];

% Initializing the output signal y[n] with zeros
output_signal = zeros(1, N + M - 1);

% Plotting the initial y[n] signal (all zeros)
figure;
stem(output_signal);
title('Initial y[n] (before convolution)');
xlabel('n');
ylabel('y[n]');

% Performing convolution
for n = 1:N + M - 1
for k = max(1, n - M + 1):min(n, N)
output_signal(n) = output_signal(n) + input_signal(k) * impulse_response(n
- k + 1);
end
end

% Defining time indices
time_indices_x = 0:N - 1;
time_indices_h = 0:M - 1;
time_indices_y = 0:N + M - 2;

% Plotting the input signal x[n]
figure;
subplot(3, 1, 1);
```

```
stem(time_indices_x, input_signal, 'b', 'LineWidth', 1.5);
title('Input Signal x[n]');
xlabel('n');
ylabel('x[n]');
grid on;

% Plotting the impulse response h[n]
subplot(3, 1, 2);
stem(time_indices_h, impulse_response, 'r', 'LineWidth', 1.5);
title('Impulse Response h[n]');
xlabel('n');
ylabel('h[n]');
grid on;

% Plotting the convolution output y[n]
subplot(3, 1, 3);
stem(time_indices_y, output_signal, 'g', 'LineWidth', 1.5);
title('Convolution Output y[n]');
xlabel('n');
ylabel('y[n]');
grid on;

% Convolution of x[n] and h[n]
sgtitle('Convolution of x[n] and h[n]');
```





OBSERVATIONS

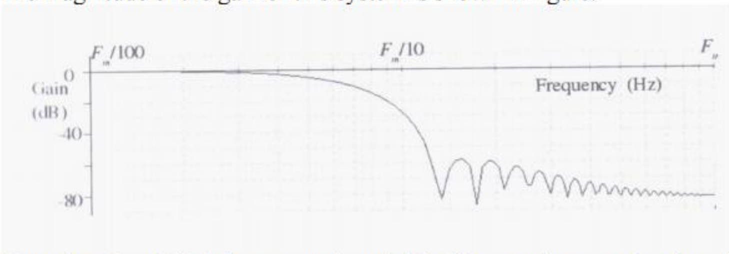
1. The input signal consists of 5 discrete values and the impulse response consists of 3 discrete values and by implementation the convolution output has 7 discrete values (which is 5+3-1).

2. The convolution gives overall output values of n at discrete time intervals, and it gives us the insight that if both of the inputs are present at a particular point n the it simply multiplies the two values. So we would be able to know that at what discrete values the signals are overlapping.

2. You are given a LPF with h[n] below.

$$h[n] = \frac{\left[0.42 - 0.5cos\left(\frac{2\pi n}{M-1}\right) + 0.08\cos\left(\frac{4\pi n}{M-1}\right)\right]}{20.58} \; for\; n = 0,1,\ldots, M-1, ; M = 50$$

The magnitude of the gain of this system is shown in figure.



Note that $F_m = F_s/2$. The properties of this filter can be tested using sinusoidal i/p signals. Generate sinusoids of frequency 5, 20, and 50 Hz, sampled @ F_s=2,000 sps, for a duration of 0.5 s, using the expression:

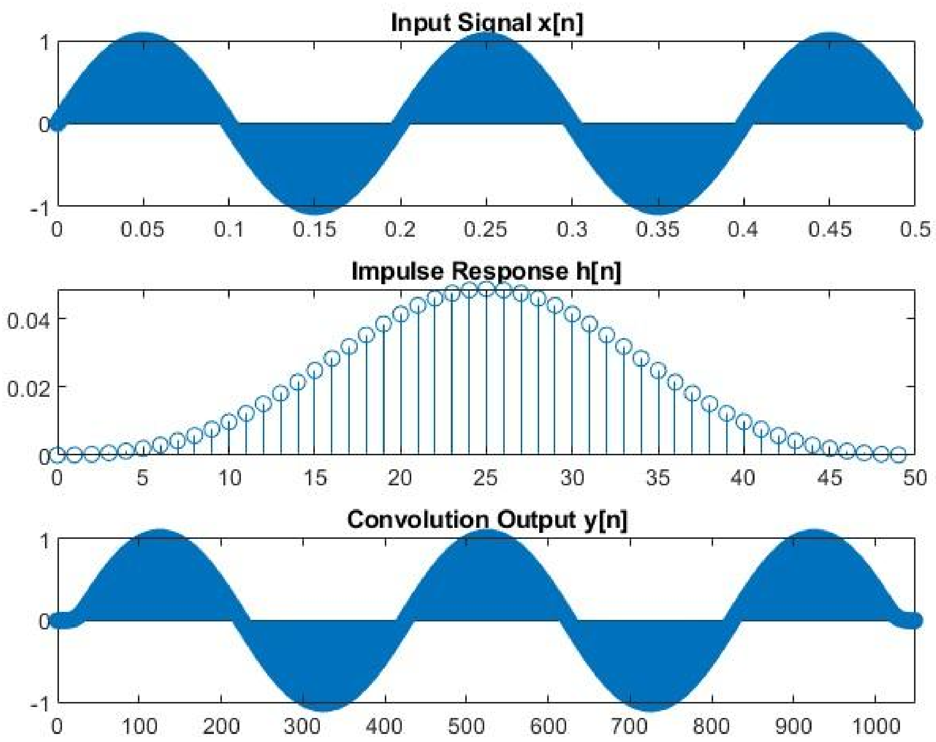
$$x[n] = \sin\left(2\pi f_o n T\right)$$

Use convolution routine from problem 1 to calculate o/p of the system to these 3 inputs. Plot the i/p and o/p functions. Tabulate the gain of the system for the three test frequencies.

```
% Initialization
M_value = 50;
SamplingFrequency = 2000;
SignalFrequency = 5;
SignalDuration = 0.5;

% Time vector
n = 0:M_value - 1;

% Defining the impulse response h[n]
h_n = (0.42 - 0.5 * cos(2 * pi * n / M_value) + 0.08 * cos(4 * pi * n /
M_value)) / 20.58;

% Defining the input signal x[n]
t = 0:1/SamplingFrequency:SignalDuration-1/SamplingFrequency;
x_n = sin(2 * pi * SignalFrequency * t);

% Initialize the output signal y[n] with zeros
y_n = zeros(1, length(x_n) + length(h_n) - 1);

% Performing convolution using nested loops
for k = 1:length(y_n)
y_n(k) = 0; % Initialize y[n] at each time index
% Computing the convolution sum for y[n]
for j = 1:length(h_n)
if k - j + 1 > 0 && k - j + 1 <= length(x_n)
y_n(k) = y_n(k) + x_n(k - j + 1) * h_n(j);
end
end
end

% Plotting input signal x[n]
figure;
subplot(3, 1, 1);
stem(t, x_n);
```

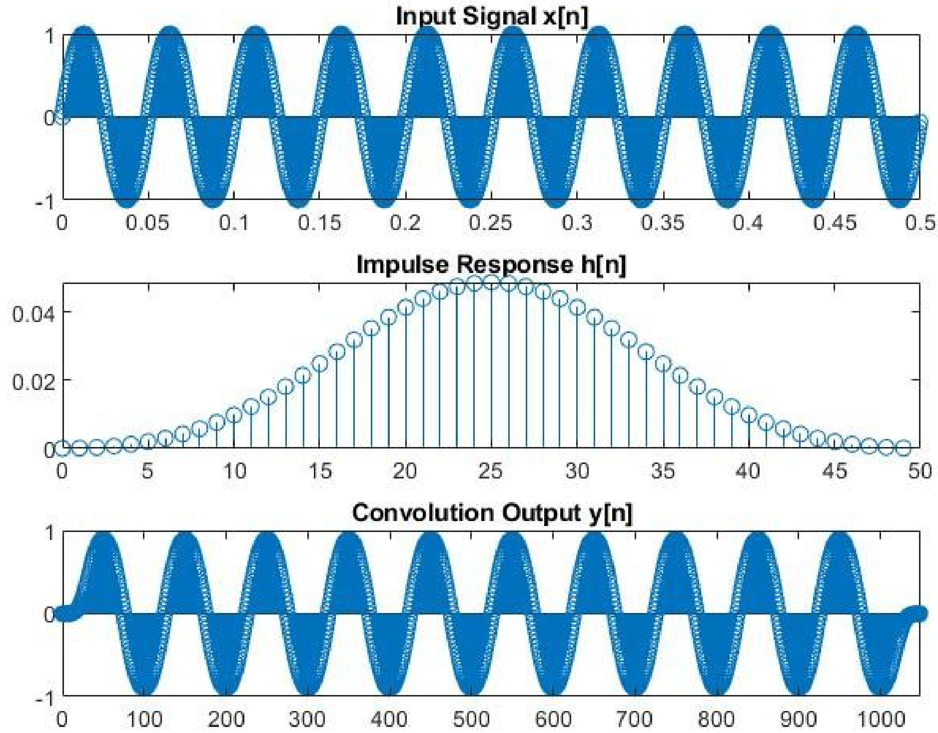
```
title('Input Signal x[n]');

% Plotting impulse response h[n]
subplot(3, 1, 2);
stem(0:M_value-1, h_n);
title('Impulse Response h[n]');

% Plotting convolution output y[n]
n_y_n = 0:length(y_n)-1;
subplot(3, 1, 3);
stem(n_y_n, y_n);
title('Convolution Output y[n]');
xlim([0, length(y_n)]);

% Calculating the gain of the system
system_gain = sum(abs(y_n)) / sum(abs(x_n));
disp(['The gain of the system is: ' num2str(system_gain)]);
```

OUTPUT AND GAIN FOR F0 = 5Hz



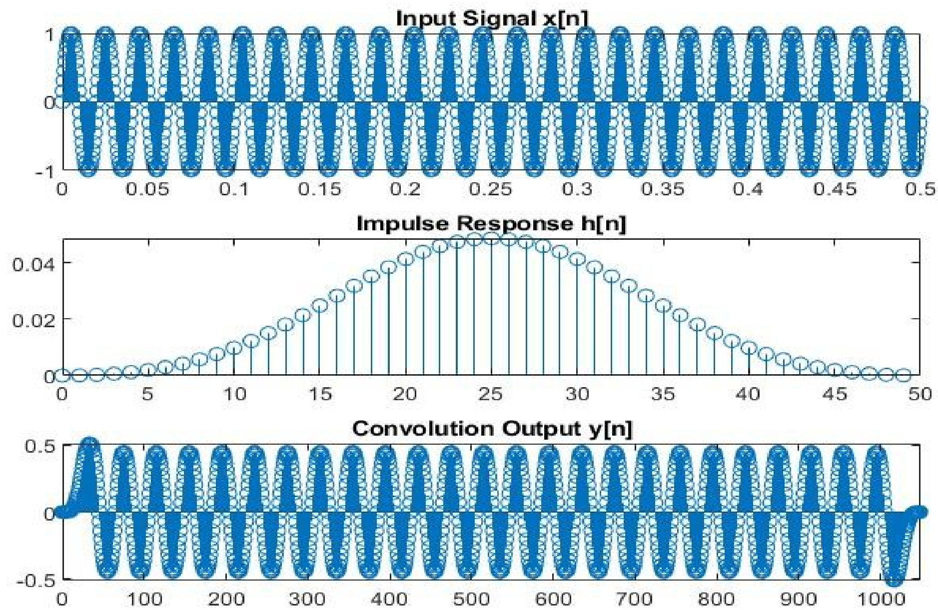

```
>> A1Q2
The gain of the system is: 1.014
```

OUTPUT AND GAIN FOR F0 = 20Hz




```
>> A1Q2
The gain of the system is: 0.90521
```

OUTPUT AND GAIN FOR F0 = 50Hz




```
>> A1Q2
The gain of the system is: 0.45778
```

OBSERVATIONS

1. The input signal contains 1000 samples for 0.5 seconds and the output signal contains 1050
samples because of the convolution.
2. The convoluted signal is the multiplication of the original signal and the LPF function (which is inversed and shifted over a period of time).

3. A recursive digital implementation of a general 2[nd] order filter can be done using the following formula:

$$b_o y[n] + b_1 y[n-1] + b_2 y[n-2] = a_o x[n] + a_1 x[n-1] + a_2 x[n-2]$$

Bessel filters are commonly used in physiological measurement instruments. The following eqns. specify the coefficient for discretized Bessel filters. The cutoff of the discrete filter is $\Omega_c = \frac{2\pi f_c}{F_s}$ where fc is cutoff freq. in Hz.

HP Bessel filter coefficients:

$$ca_o = 4, b_o = \frac{\Omega_c^2}{3} + 2\Omega_c + 4,$$
$$a_1 = -8, b_1 = 2\left[\frac{\Omega_c^2}{3} - 4\right],$$
$$a_2 = 4, b_2 = \Omega_c^2 - 2\Omega_c + 4.$$

LP Bessel filter coefficients:

$$a_o = 3, b_o = \frac{4}{\Omega_c^2} + \frac{6}{\Omega_c} + 3,$$
$$a_1 = 6, b_1 = \frac{-8}{\Omega_c^2} + 6,$$
$$a_2 = 3, b_2 = \frac{4}{\Omega_c^2} - \frac{6}{\Omega_c} + 3.$$

Write a program to implement the above two filters separately, with LPF filter cutoff: f_{c1}=100Hz and HPF cutoff: f_{c2}=100Hz. The program should contain an input array of 1,000 points and an output array of 1000 points. The final output of the program should be the graphical display of the i/p and o/p.

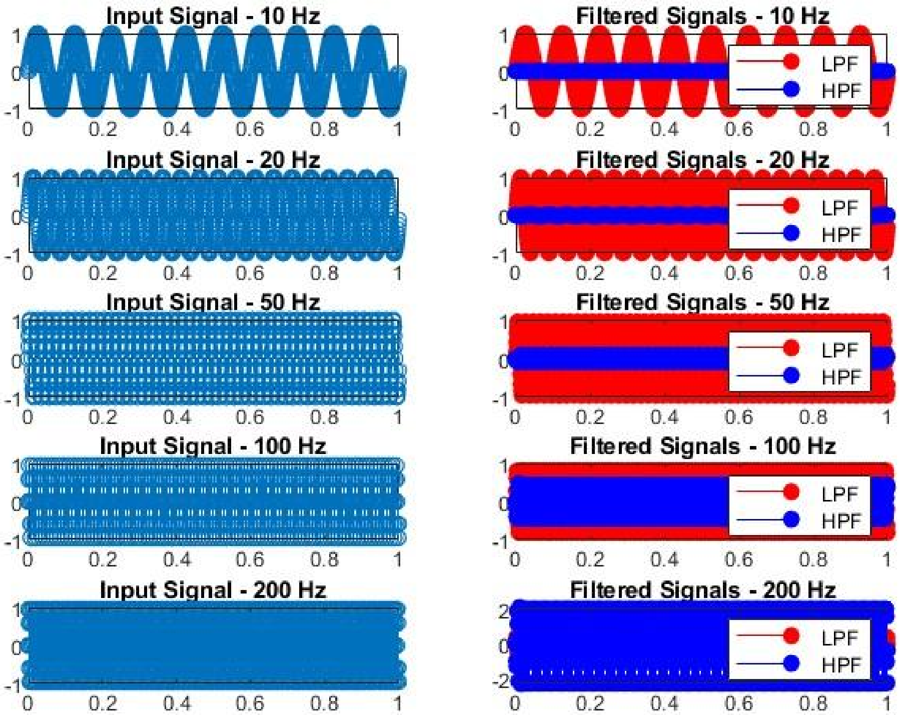
Generate sinusoids of frequency 10, 50, 100 and 200 Hz with sampling rates of Fs=1000 sps. In each case pass the signal through LPF and HPF and note the amplitude and phase of the O/p relative to the i/p. Plot the gain and phase shift against frequency.

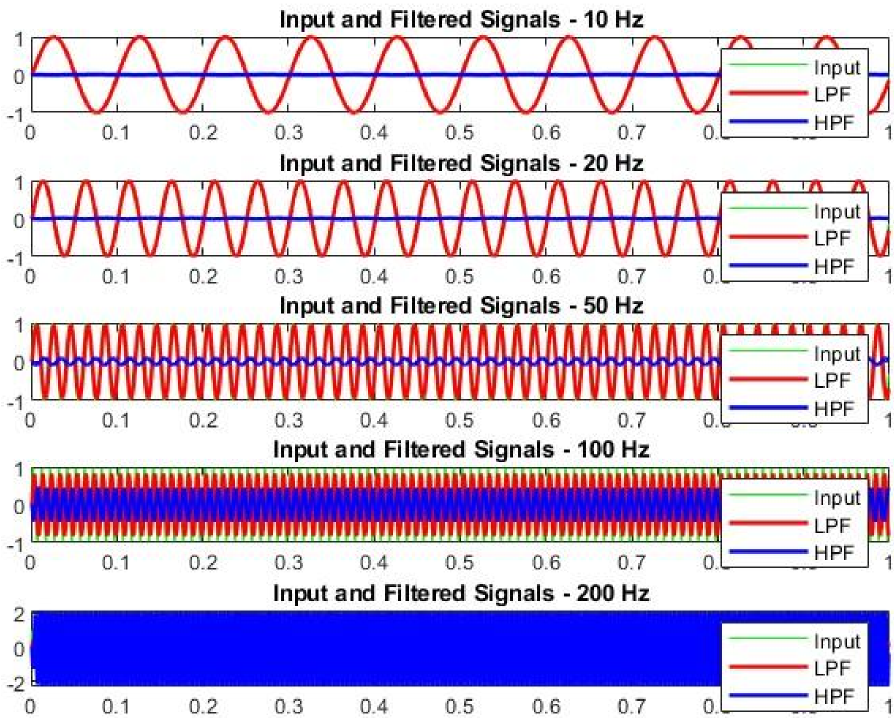
```
% Initialization
Fs = 1000;
T = 1/Fs;
N = 1000;
t = (0:N-1) * T;


% Declaring Cutoff frequencies
fc1 = 100; % LPF cutoff frequency in Hz
fc2 = 100; % HPF cutoff frequency in Hz


% Bessel filter coefficients
Omega_c1 = 2 * pi * fc1 / Fs;
Omega_c2 = 2 * pi * fc2 / Fs;


% LP Bessel filter coefficients
a0_LP = 3;
a1_LP = 6;
```

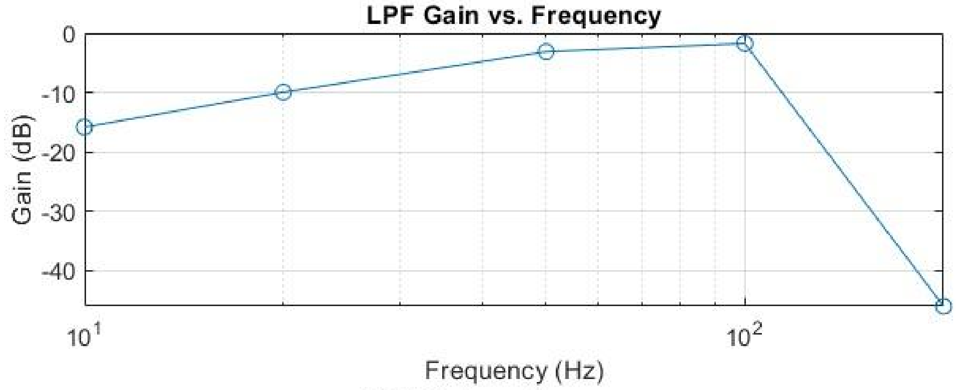
```
a2_LP = 3;
b0_LP = (4 / (Omega_c1^2)) + (6 / Omega_c1) + 3;
b1_LP = (-8 / (Omega_c1^2)) + 6;
b2_LP = (4 / (Omega_c1^2)) - (6 / Omega_c1) + 3;


% HP Bessel filter coefficients
a0_HP = 4;
a1_HP = -8;
a2_HP = 4;
b0_HP = ((Omega_c2^2) / 3) + 2 * Omega_c2 + 4;
b1_HP = 2 * ((Omega_c2^2) / 3) - 4;
b2_HP = (Omega_c2^2) - 2 * Omega_c2 + 4;


% Initializing output arrays
output_LP = zeros(1, N);
output_HP = zeros(1, N);


% Generating input sinusoids
frequencies = [10, 20, 50, 100, 200];
input_signals = zeros(length(frequencies), N);
for i = 1:length(frequencies)
input_signals(i, :) = sin(2 * pi * frequencies(i) * t);
end


% Applying LPF and HPF to input signals
for i = 1:length(frequencies)
x = input_signals(i, :);
y_LP = zeros(1, N);
y_HP = zeros(1, N);


% Applying LPF
for n = 3:N
y_LP(n) = (a0_LP * x(n) + a1_LP * x(n-1) + a2_LP * x(n-2) - b1_LP * y_LP(n-
1) - b2_LP * y_LP(n-2)) / b0_LP;
end


% Applying HPF
for n = 3:N
y_HP(n) = (a0_HP * x(n) + a1_HP * x(n-1) + a2_HP * x(n-2) - b1_HP * y_HP(n-
1) - b2_HP * y_HP(n-2)) / b0_HP;
end
output_LP(i, :) = y_LP;
output_HP(i, :) = y_HP;


% Calculating the magnitude of the frequency response (gain) at this
frequency
gain_lpf(i) = abs(y_LP(N));
gain_hpf(i) = abs(y_HP(N));
phase_lpf(i) = atan2(imag(y_LP(N)), real(y_LP(N))) * (180 / pi);
phase_hpf(i) = atan2(imag(y_HP(N)), real(y_HP(N))) * (180 / pi);
end
```

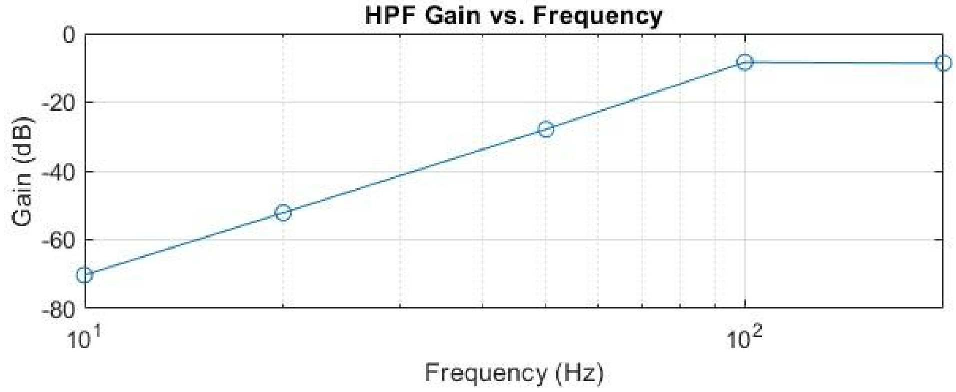
```matlab
% Ploting the filtered signals
figure;
for i = 1:length(frequencies)
subplot(length(frequencies), 2, 2*i-1);
stem(t, input_signals(i, :));
title(['Input Signal - ' num2str(frequencies(i)) ' Hz']);
subplot(length(frequencies), 2, 2*i);
stem(t, output_LP(i, :), 'r', 'MarkerFaceColor', 'r');
hold on;
stem(t, output_HP(i, :), 'b', 'MarkerFaceColor', 'b');
title(['Filtered Signals - ' num2str(frequencies(i)) ' Hz']);
legend('LPF', 'HPF');
end


% Ploting the output graph for all frequencies
figure;
for i = 1:length(frequencies)
subplot(length(frequencies), 1, i);
plot(t, input_signals(i, :), 'g');
hold on;
plot(t, output_LP(i, :), 'r', 'LineWidth', 1.5);
plot(t, output_HP(i, :), 'b', 'LineWidth', 1.5);
title(['Input and Filtered Signals - ' num2str(frequencies(i)) ' Hz']);
legend('Input', 'LPF', 'HPF');
end


% Ploting the gain vs. frequency using semilogx
figure;
subplot(211);
semilogx(frequencies, 20 * log10(gain_lpf), '-o');
title("LPF Gain vs. Frequency");
xlabel('Frequency (Hz)');
ylabel('Gain (dB)');
grid on;
subplot(212);
semilogx(frequencies, 20 * log10(gain_hpf), '-o');
title("HPF Gain vs. Frequency");
xlabel('Frequency (Hz)');
ylabel('Gain (dB)');
grid on;


% Ploting the phase shift vs. frequency using semilogx
figure;
subplot(211);
semilogx(frequencies, phase_lpf, '-o');
title("LPF Phase Shift vs. Frequency");
xlabel('Frequency (Hz)');
ylabel('Phase Shift (degrees)');
grid on;
subplot(212);
semilogx(frequencies, phase_hpf, '-o');
title("HPF Phase Shift vs. Frequency");
xlabel('Frequency (Hz)');
ylabel('Phase Shift (degrees)');
grid on;
```

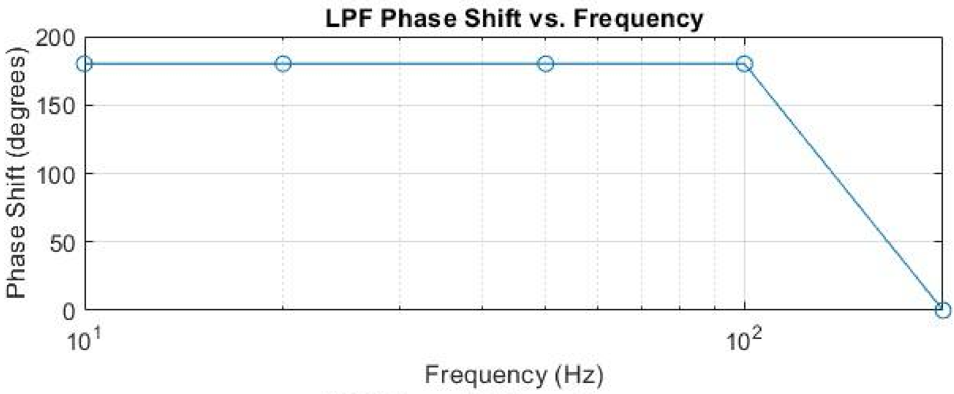
OUTPUTS

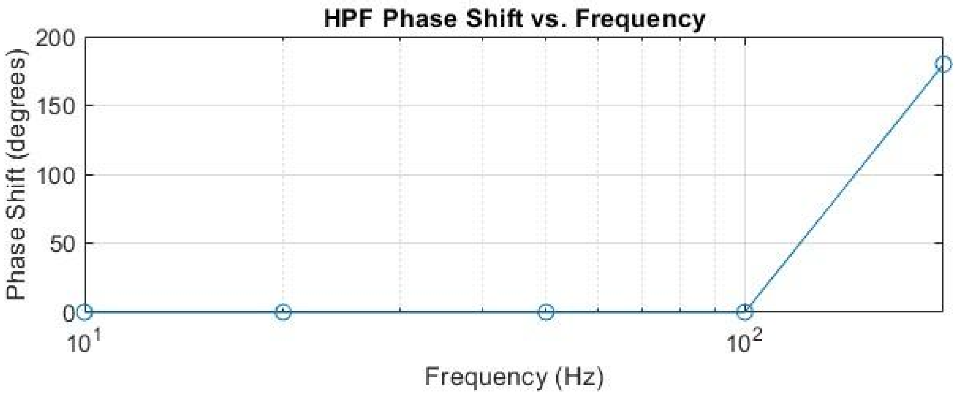

Input Signal - 10 Hz
Filtered Signals - 10 Hz
Input Signal - 20 Hz
Filtered Signals - 20 Hz
Input Signal - 50 Hz
Filtered Signals - 50 Hz
Input Signal - 100 Hz
Filtered Signals - 100 Hz
Input Signal - 200 Hz
Filtered Signals - 200 Hz
LPF
HPF


Input and Filtered Signals - 10 Hz
Input and Filtered Signals - 20 Hz
Input and Filtered Signals - 50 Hz
Input and Filtered Signals - 100 Hz
Input and Filtered Signals - 200 Hz
Input
LPF
HPF


LPF Gain vs. Frequency
Gain (dB)
0
-10
-20
-30
-40
10^1
10^2
Frequency (Hz)


HPF Gain vs. Frequency
Gain (dB)
0
-20
-40
-60
-80
10^1
10^2
Frequency (Hz)


LPF Phase Shift vs. Frequency
Phase Shift (degrees)
200
150
100
50
0
10^1
10^2
Frequency (Hz)


HPF Phase Shift vs. Frequency
Phase Shift (degrees)
200
150
100
50
0
10^1
10^2
Frequency (Hz)

OBSERVATIONS

1. For the LPF, as the input frequency increases, the gain decreases. This is the characteristic behaviour of a low-pass filter, which allows lower frequencies to pass through while attenuating higher frequencies.
2. Conversely, for the HPF, as the input frequency increases, the gain increases. HPFs allow higher frequencies to pass through while attenuating lower frequencies.