CAPAÑARIHAN, Alein Miguel P. (2022167757)
IBASAN, Chamel Miles S. (2022167811)
3ECE-A

SUMMATIVE ASSESSMENT #1

Instructions: Follow the procedures then answer the following questions and fill in the required items.

Questions and Procedures:

1. Create a time vector t from 0 to 10 seconds with 1000 samples. Generate a clean signal x defined as $x(t) = sin(2\pi ft)$ where f = 2kHz.

    a. Generate Gaussian noise y using `randn` and add to the clean signal to create a noisy signal z.

    MATLAB code:

```
F = 2e3; %Frequency
Fs = 1000/10; %Sampling Frequency
f = F/Fs;
n = 0:1000-1; %No. of samples
duration = 10;
t = 0:1/Fs:duration-1/Fs;

x_t = sin(2*pi*f*t); %Clean Signal

rng(654) %random seed
y = randn(1,1000); % randomly-distributed noise
z = x_t + y; % noisy signal
```

    b. Transpose the noisy signal to convert it into a column vector. Replace all values in the noisy signal greater than 1 with a fixed value of 0.5. Use logical function to find all values in the modified signal less than -0.5 and replace them with -0.5.

    MATLAB code:

```
ztrans = z.'; % modified noisy signal w/c is transposed
ztrans(find(ztrans>1)) = 0.5;
ztrans(find(ztrans<-0.5)) = -0.5;
```

c. Use subplot to plot the original clean signal, the noise, the noisy signal, and the modified noisy signal. Label the axes and title of the subplots.

MATLAB code:

```
p = tiledlayout('vertical');
nexttile, plot(n,x_t), title('Clean Signal');
nexttile, plot(n,y),title('Noise Signal');
nexttile, plot(n,z), title('Noisy Signal');
nexttile, plot(n,ztrans), title('Modified noise signal')
xlabel(p,'Time Index(n)')
ylabel(p,'Amplitude')
```

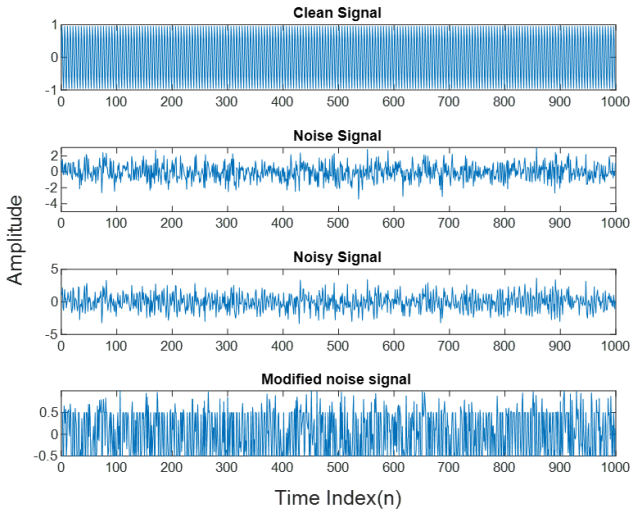

Figure 1 - The Effect of Noise on a Clean Signal

Conclusion: It can be observed that by adding gaussian noise to a sinusoidal signal, the signal becomes highly distorted and unrecognizable.

CAPAÑARIHAN, Alein Miguel P. (2022167757)
IBASAN, Chamel Miles S. (2022167811)
3ECE-A

2. Generate five (5) single-tone sinusoidal signals with the following frequencies: 100 Hz, 300 Hz, 500 Hz, 700 Hz, and 900 Hz with 1V, 0.5V, 0.25V, 0.125V, and 0.0625V, respectively. Set a sampling frequency of 50,000 Hz.

    a. Plot the five signals separately in a subplot and observe the plot from 0 to 0.1 seconds.

    MATLAB code:

```
f = [100 300 500 700 900]; %matrix of frequencies
a = [1 0.5 0.25 0.125 0.0625]; %matrix of amplitudes

Fs = 50000;
duration = 0.1;
t = 0:1/Fs:duration-1/Fs;

x1 = a(1).*sin(2.*pi.*(f(1))*t);
x2 = a(2).*sin(2.*pi.*(f(2))*t);
x3 = a(3).*sin(2.*pi.*(f(3))*t);
x4 = a(4).*sin(2.*pi.*(f(4))*t);
x5 = a(5).*sin(2.*pi.*(f(5))*t);

p = tiledlayout('vertical');

nexttile, plot(t,x1), title('100 Hz Sinusoid');
nexttile, plot(t,x2), title('300 Hz Sinusoid');
nexttile, plot(t,x3), title('500 Hz Sinusoid');
nexttile, plot(t,x4), title('700 Hz Sinusoid');
nexttile, plot(t,x5), title('900 Hz Sinusoid');


xlabel(p,'Time Index(t)')
ylabel(p,'Amplitude')
```

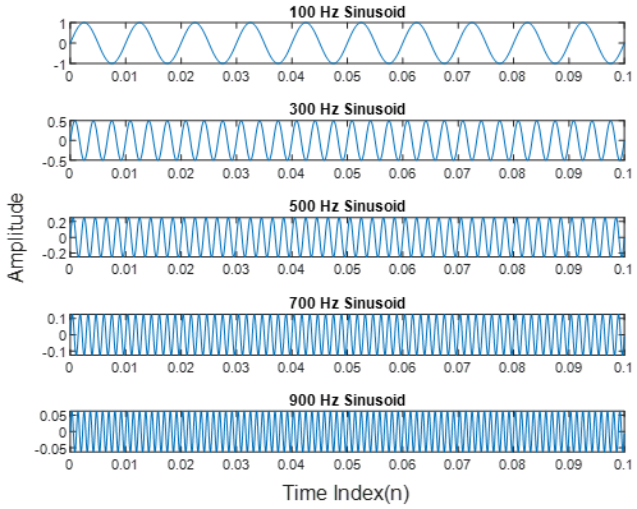

Figure 2a - Single-tone sinusoidal signals

b. Add the signals together, which will be represented as $g(t)$. Plot the result.

MATLAB code:

```
g_t = x1 +  x2 +  x3 +  x4 + x5;
plot(t,g_t);
title('Multi-tone Sinusoidal')
xlabel('Time Index(t)')
ylabel('Amplitude')
```

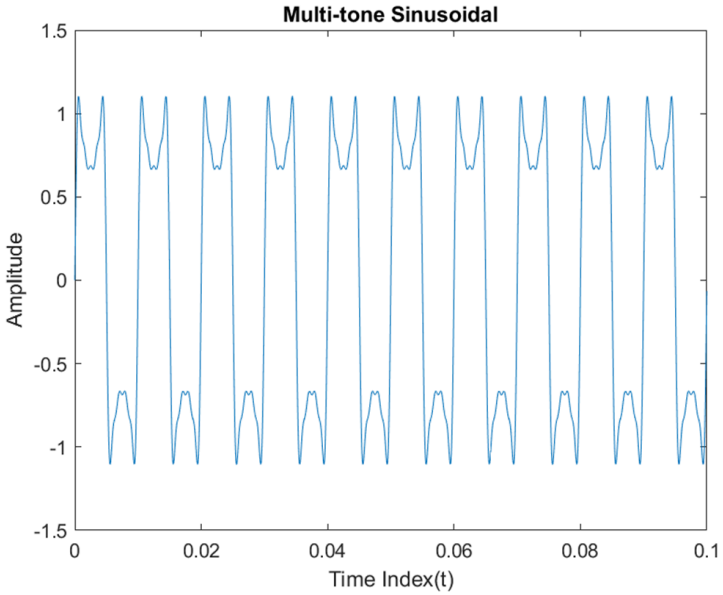

Figure 2b - The composite signal *g(t)*

c. Compute the power of the signal $g(t)$. Specify the result.

MATLAB code:

```
period = find(t==0.01);
power = (1/period) .* sum(abs(g_t(1:period-1).^2));

>> disp(power)
   0.6647
```

d. Supposed that $g(t)$ will be manipulated as shown below:

$$s(t) = [1 + g(t)]\, cos(2\pi(5000)t)$$

Plot the output waveform in the time-domain and frequency-domain. Use 1024 fft-points.

MATLAB code:

```
s_t1 = (1+g_t).*cos(2*pi*(5000)*t);

subplot(2,1,1)
plot(t,s_t1)
title('Time-Domain Graph')
xlabel('Time Index(t)')
ylabel('Amplitude')

S_t1 = fft(s_t1,1024);
f_axis = ((0:511)/512)*(Fs/2);

subplot(2,1,2)
plot(f_axis,abs(S_t1(1:512)))
title('Frequency Domain Graph')
xlabel('Frequency(Hz)')
ylabel('Amplitude')
```

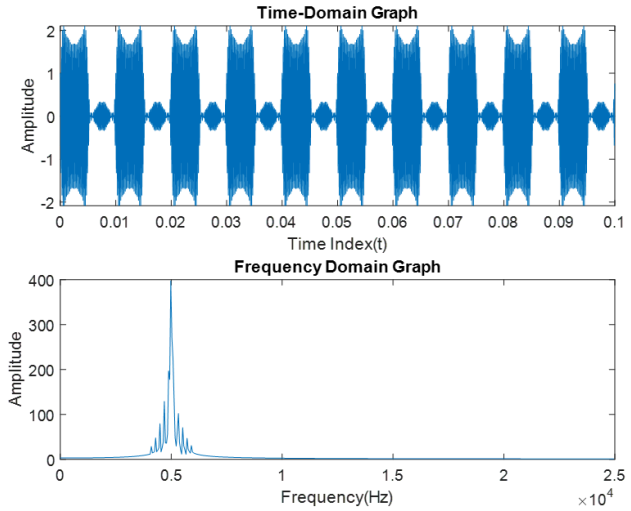

Figure 2d - *s(t)*$_1$ in the time and frequency domain

e. Supposed that the $g(t)$ will be manipulated as shown below:

$$s(t) = [1 + g(t)]\, cos(2\pi(10000)t)$$

Plot the output waveform in the time-domain and frequency-domain. Use 1024 fft-points.

MATLAB code:

```
s_t2 = (1+g_t).*cos(2*pi*(10000)*t);

subplot(2,1,1)
plot(t,s_t2)
title('Time-Domain Graph')
xlabel('Time Index(t)')
ylabel('Amplitude')

S_t2 = fft(s_t2,1024);
f_axis = ((0:511)/512)*(Fs/2);

subplot(2,1,2)
plot(f_axis,abs(S_t1(1:512)))
title('Frequency Domain Graph')
xlabel('Frequency(Hz)')
ylabel('Amplitude')
```

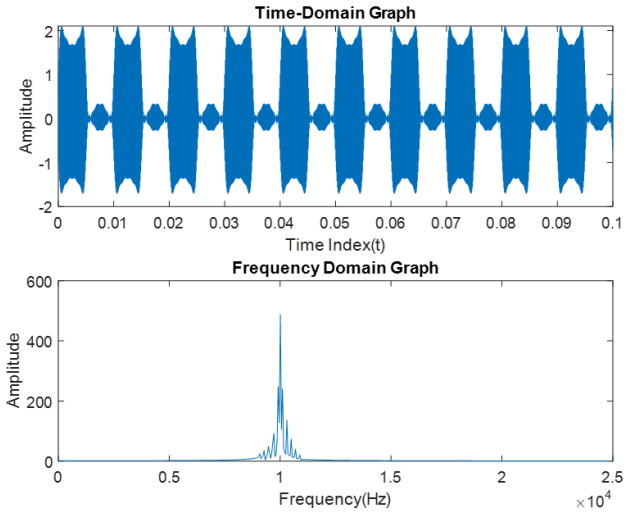

Figure 2e - *s(t)₂* in the time and frequency domain

f. Compare the results of (d) and (e) in terms of the (i) time-domain plot, (ii) frequency-domain plot, and (iii) power.

Comment: the time-domain plot of (d) and (e) appear identical, but there are differences in the frequency domain, as it is clearly visible that two signals have different peaks. This coincides with the parameters we set for both signals, one having a frequency of 5000, and the other a frequency of 10000. As for the power calculations, both signals produced 0.8313W.

Conclusion: In conclusion, analysis in the frequency domain enables us to see changes in the signal that are otherwise unnoticeable in the time domain. Another finding is that increasing the frequency of the signal does not necessarily increase the power output of said signal.

CAPAÑARIHAN, Alein Miguel P. (2022167757)
IBASAN, Chamel Miles S. (2022167811)
3ECE-A

3. The convolution operation defined as

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = x(n) * h(n)$$

involves **four elementary signal manipulation operations**:

1. **Folding:** Fold $h(k)$ to obtain $h(–k)$.
2. **Shifting:** Shift $h(n – k)$ to the right if n is positive and to the left if n is negative.
3. **Multiplication:** Multiply $x(k)$ to $h(n – k)$ to obtain a product sequence at $n = n_0$.
4. **Addition:** Add the terms in the product sequence to obtain $y(n = n_0)$.

Create a MATLAB function `convolve` that generates the convolution sequence of two finite-length sequences with a given delay/advance of k units. The input arguments are the two input matrices a and b their corresponding time indices n_a and n_b. The output of the function is the convolution sum y and its index n.

A help description for the function follows:

```
CONVOLVE Convolution of two sequences.
[C,N] = CONVOLVE(A,Na,B,Nb) convolves the vectors A and B
with their corresponding time indices Na and Nb
respectively. The resulting vector is returned in C and its
time index in N. The length of the resulting vector is
LENGTH(A)+LENGTH(B)-1.
```

Use the `sigshift()`, `sigfold()`, `sigadd()`, and `sigmult()` functions that you made in the experiment/activity previously.

Verify the functions by testing using the following:

a) $x(n) = [2 \quad \underset{\uparrow}{-1} \quad 0 \quad 3] \quad h(n) = [-5 \quad 1 \quad 2 \quad -1]$

b) $x_1(n) = [1 \quad 2 \quad \underset{\uparrow}{1} \quad 3] \quad x_2(n) = [3 \quad \underset{\uparrow}{4} \quad -2 \quad 1]$

WARNING: DO NOT USE MATLAB's built-in function `CONV(H,N)`.

CAPAÑARIHAN, Alein Miguel P. (2022167757)
IBASAN, Chamel Miles S. (2022167811)
3ECE-A

MATLAB Code (Editor):

```
function [C,N] = CONVOLVE(A,Na,B,Nb)

    % sigfold
    nb = -fliplr(Nb);
    b = fliplr(B);

    C = zeros(1,(length(A) + length(B) - 1));
    for n = 1:length(C)

        % sigshift
        m = min(Na) + min(Nb);
        b_shift = b;
        nb_shift = nb + (n + (m - 1));

        % sigmult and summed
        c_min = min(min(Na),min(nb_shift));
        c_max = max(max(Na),max(nb_shift));
        cn = c_min:c_max;

        n_A = zeros(1,length(cn));
        n_A((cn >= min(Na)) & (cn <= max(Na))) = A();
        n_b_shift = zeros(1,length(cn));
        n_b_shift((cn >= min(nb_shift)) & (cn <= max(nb_shift)))
        = b_shift();

        C(n) = sum((n_A).*(n_b_shift));

    end

    N = cn;

end
```

Conclusion: It was found that the `conv()` function can be replicated by applying certain DT signal processing operations (`sigfold(), sigshift(), sigmult(), sigadd()`) on the given functions as it produced the same output response (*y(n)*) and index (*n*). Therefore, it can be said that convolution can be characterized as a process that uses the operations mentioned above on two functions to produce an output.

CAPAÑARIHAN, Alein Miguel P. (2022167757)
IBASAN, Chamel Miles S. (2022167811)
3ECE-A

4. Consider the difference equation shown below.

$$y(n) = 1.6148y(n-1) - 1.9395y(n-2) + 1.2807y(n-3) - 0.5852y(n-4) +$$
$$0.0899x(n) + 0.0257x(n-1) - 0.0257x(n-2) - 0.0899x(n-3)$$

Solve the following:

a. System Function, $H(z)$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_o + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + b_4 z^{-4}}$$

$$H(z) = \left(\frac{b_o + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4}}\right) \times \left(\frac{z^4}{z^4}\right)$$

$$H(z) = \frac{b_o z^4 + b_1 z^3 + b_2 z^2 + b_3 z}{z^4 + a_1 z^3 + a_2 z^2 + a_3 z + a_4}$$

$$H(z) = \frac{0.0899z^4 + 0.0257z^3 - 0.0257z^2 - 0.0899z}{z^4 - 1.6148z^3 + 1.9395z^2 - 1.2807z + 0.5852}$$

b. Position of the poles and zeros on the z-plane

MATLAB code:

```
% Numerator and Denominator coefficients
b = [0.0899 0.0257 -0.0257 -0.0899 0];
a = [1 -1.6148 1.9395 -1.2807 0.5852];

% Pole-Zero Placement of H(z)
zplane(b,a);
title('Poles and Zeros of H(z)');
```

CAPAÑARIHAN, Alein Miguel P. (2022167757)
IBASAN, Chamel Miles S. (2022167811)
3ECE-A

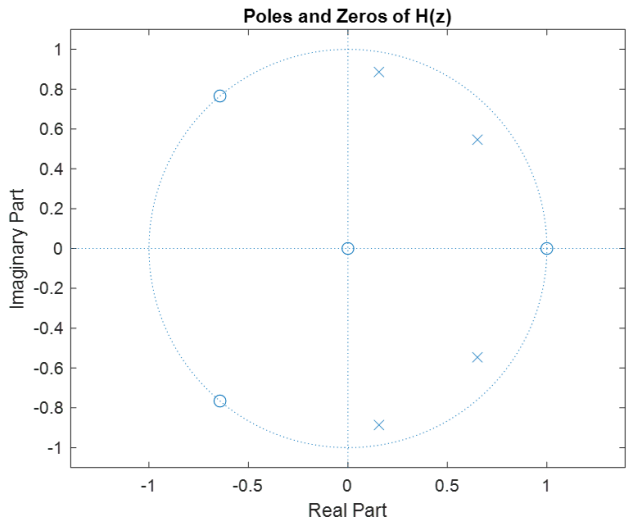

Figure 4a - Pole-Zero Placement of *H(z)*

c. Expression of the impulse response, $h(n)$. Verify your answer by comparing the stem plots of the first 50 samples of $h(n)$ vs. using `impz()` function.

$$h(n) = r_1 p_1^{\ n} u(n) + r_2 p_2^{\ n} u(n) + r_3 p_3^{\ n} u(n) + r_4 p_4^{\ n} u(n) + r_5 p_5^{\ n} u(n)$$

$$h(n) = (-0.0754 - j0.0674)(0.1563 + j0.8864)^n u(n) + (-0.0754 + j0.0674)(0.1563 - j0.8864)^n u(n) + (0.1204 + j0.0749)(0.6511 + j0.5463)^n u(n) + (0.1204 - j0.0749)(0.6511 - j0.5463)^n u(n)$$

d. Magnitude frequency response plot of the system. Comment on the plot of the frequency response in relation to the position of the poles and zeros.

MATLAB code:

```
[hf, wf] = freqz(b,a);

plot(wf/pi, abs(hf));
title('Magnitude Response');
xlabel('Normalized Frequency (Hz)');
ylabel('Magnitude (dB)');
```

CAPAÑARIHAN, Alein Miguel P. (2022167757)
IBASAN, Chamel Miles S. (2022167811)
3ECE-A

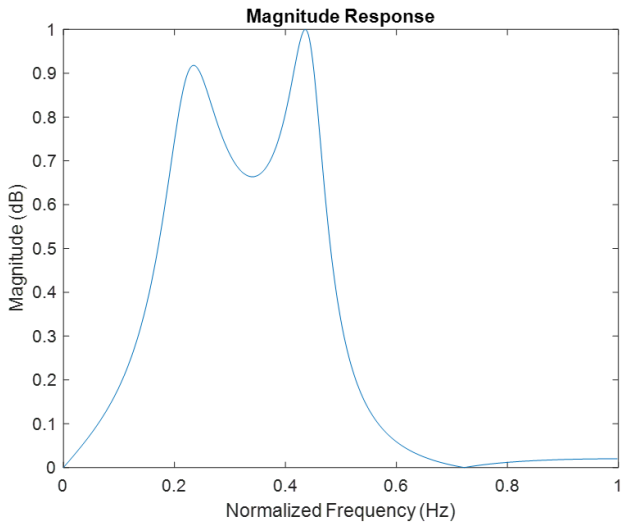

Figure 4d - Magnitude Frequency Response of *H(z)*

Comment: The magnitude response graph is in line with the pole-zero plot produced in Figure 4a. We can clearly see peaks in magnitude at where the poles are present, and a decrease in magnitude where there are zeros.

Conclusion: The magnitude response graph is a much more intuitive and straightforward way of looking at the peaks and dips of magnitude at specific frequencies.

CAPAÑARIHAN, Alein Miguel P. (2022167757)
IBASAN, Chamel Miles S. (2022167811)
3ECE-A

5. Download the data file ECE21113L_SA1_1.mat is a two-column array in which the first column represents the sample index n, while the second column is the samples of the signal x(n). If this signal passes through the DT-LTI system in Problem #4, determine the following:

   a. Output response, $y_f(n)$ using the `filter()` function

      MATLAB code:

```
x_file = load('ECE21113L_SA1_1.mat');
x_array = struct2array(x_file);
x = transpose(x_array(:,2));

b = [0.0899 0.0257 -0.0257 -0.0899 0];
a = [1 -1.6148 1.9395 -1.2807 0.5852];

yf = filter(b,a,x);
```

   b. Output response, $y_c(n)$ using the `conv()` function

      MATLAB code:

```
x_file = load('ECE21113L_SA1_1.mat');
x_array = struct2array(x_file);
x = transpose(x_array(:,2));

b = [0.0899 0.0257 -0.0257 -0.0899 0];
a = [1 -1.6148 1.9395 -1.2807 0.5852];

[r, p, k] = residue(b,[a,0]); %finds the poles & zeros

h = zeros(1,99);
for n = 1:99
   h(n) = sum(r.*(p.^(n-1)));
end

yc = conv(x,h);
```

c. Plot the responses for 5a and 5b in a single figure window (label the plots accordingly). Describe the similarities/differences in the responses obtained.

MATLAB code:

```
n=1:98;
n_conv = 1 : length(x) + length(h) - 1;


p = tiledlayout('flow');
nexttile, plot(n,yf), title('Output Response using filter()');
nexttile, plot(n_conv,yc), title('Output Response using conv()');
xlabel(p,'Time Index (n)');
ylabel(p,'Amplitude');
```

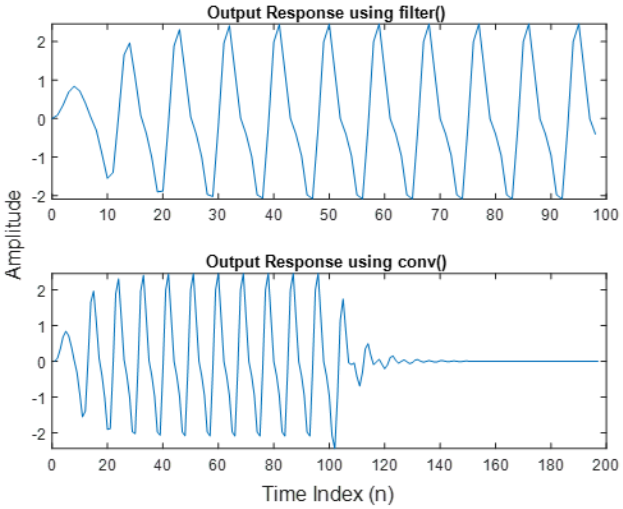

Figure 5c - $y_f(n)$ and $y_c(n)$

Comment: Upon visual inspection, the filter() and conv() function have identical graphs at n < 100. But the conv() function generated extra indices due to N = 1:length(x) + length(h) - 1.

d. Obtain the frequency plots of $x(n)$ and $y(n)$. Comment on the similarities/differences between the plots in relation to the position of the poles and zeros.

MATLAB code:

```
[Hx,wx] = freqz(x,1,1024);
[Hy,wy] = freqz(y,1,1024);

q = tiledlayout('flow');
nexttile, plot(wx/pi,abs(Hx)), title('Frequency Response of x(n)');
nexttile, plot(wy/pi,abs(Hy)), title('Frequency Response of y(n)');
xlabel(q,'Frequency (Hz)');
ylabel(q,'Magnitude');
```

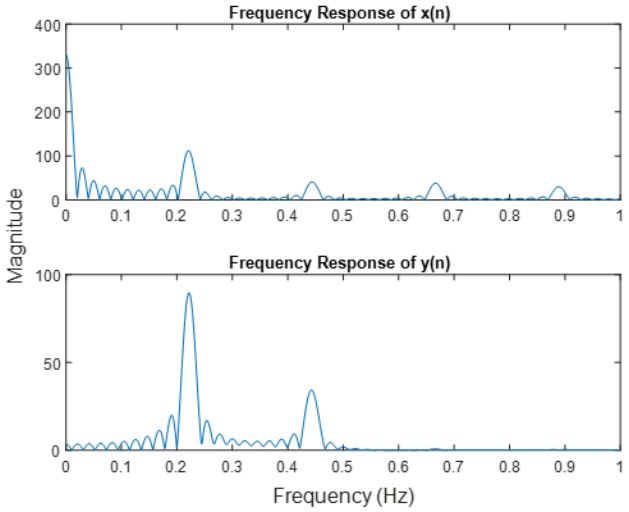

Figure 5d - Frequency response of *x(n)* and *y(n)*

Comment: The graph shows how the system *h(n)* only lets certain frequencies of the input *x(n)* pass through while the rest are attenuated. This shows that a portion of the input's poles are present at the output when frequency increases.

Conclusion: The system *h(n)* acts as a band-pass filter since any frequencies not within a certain range are attenuated.