Efe Tarhan
22002840
EEE321-02

EEE321 LAB WORK 6

1)

My school number is 22002840. Therefore $N_1 = 8$ and $N_2 = 2$. This implies $M_1 = 10$ and $M_2 = 4$. According to the specifications of the lab the passband of the signal is between $\left(\frac{\pi}{10}, \frac{\pi}{4}\right)$ and $\left(-\frac{\pi}{10}, -\frac{\pi}{4}\right)$. Reason of this is because of the following property of the z-transform.

$$X(e^{jw}) = \sum_{k=-\infty}^{\infty} h[k] \cdot e^{-jwk}$$

$$X^*(e^{jw}) = \sum_{k=-\infty}^{\infty} h^*[k] \cdot e^{jwk}$$

$$X^*(e^{-jw}) = \sum_{k=-\infty}^{\infty} h^*[k] \cdot e^{-jwk} = X^*(e^{-jw}) = \sum_{k=-\infty}^{\infty} h[k] \cdot e^{-jwk}$$

$$X(e^{-jw}) = X^*(e^{-jw})$$

This means $X(e^{jw})$ is conjugate symmetric and therefore all the zeros, poles and the pass-band must be symmetric for x axis. After finding this result another specification of the system is satisfied which is to have order of the filter as $N_1 + 5 = 13$. This shows that there must be 13 poles of the z-transform and they must be located inside the unit circle for the causality and stability requirement, but they also must be close to the unit circle to ensure maximum amplification of the passband areas. Poles are distributed uniformly to the passband areas of the discrete time Fourier transform plot. 11 zeros are added uniformly to the stopband regions to increase the quality of filter. After that h[n] is found by using the digital calculations of MATLAB. Coefficients of the impulse response comes from the coefficients of the nominator. Those coefficients can be used for determining the graph of a finite duration of the impulse which will be implemented below.

H(z) can be defined in terms of the polynomials which are A and B. Roots of B are the zeros of the system and roots of A are the poles of the system. Polynomials of A(z) and B(z) can be examined below.

B(z) = (z-H_zeros(1))*(z-H_zeros(2))*(z-H_zeros(3))*(z-H_zeros(4))*(z-H_zeros(5))*(z-H_zeros(6))*(z-H_zeros(7))*(z-H_zeros(8))*(z-H_zeros(9))*(z-H_zeros(10))*(z-H_zeros(11));

A(z) = (z-H_poles(1))*(z-H_poles(2))*(z-H_poles(3))*(z-H_poles(4))*(z-H_poles(5))*(z-H_poles(6))*(z-H_poles(7))*(z-H_poles(8))*(z-H_poles(9))*(z-H_poles(10))*(z-H_poles(11))*(z-H_poles(12));

where,

p0 = pi/10 + 1 * ((3*pi/20)/7);

p1 = pi/10 + 2 * ((3*pi/20)/7);

p2 = pi/10 + 3 * ((3*pi/20)/7);

p3 = pi/10 + 4 * ((3*pi/20)/7);

p4 = pi/10 + 5 * ((3*pi/20)/7);

p5 = pi/10 + 6 * ((3*pi/20)/7);

p6 = -pi/10 -1 * ((3*pi/20)/7);

p7 = -pi/10 -2 * ((3*pi/20)/7);

p8 = -pi/10 -3 * ((3*pi/20)/7);

p9 = -pi/10 -4 * ((3*pi/20)/7);

p10 = -pi/10 -5 * ((3*pi/20)/7);

p11 = -pi/10 -6 * ((3*pi/20)/7);

p12 = 0;

Hpoles = [p0,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12];

%Determination of zero locations

z0 = pi/60;

z1 = pi/4 + 1 * ((3*pi/2)/10);

z2 = pi/4 + 2 * ((3*pi/2)/10);

z3 = pi/4 + 3 * ((3*pi/2)/10);

z4 = pi/4 + 4 * ((3*pi/2)/10);

z5 = pi/4 + 5 * ((3*pi/2)/10);

z6 = -pi/4 -1 * ((3*pi/2)/10);

z7 = -pi/4 -2 * ((3*pi/2)/10);

z8 = -pi/4 -3 * ((3*pi/2)/10);

z9 = -pi/4 -4 * ((3*pi/2)/10);

z10 = -pi/60;

```
Hzeros = [z0,z1,z2,z3,z4,z5,z6,z7,z8,z9,z10];
```

And

```
H_zeros =

[1*exp(1j*Hzeros(1));1*exp(1j*Hzeros(2));1*exp(1j*Hzeros(3));...
        1*exp(1j*Hzeros(4));1*exp(1j*Hzeros(5));1*exp(1j*Hzeros(6));...
        1*exp(1j*Hzeros(7));1*exp(1j*Hzeros(8));1*exp(1j*Hzeros(9));...
        1*exp(1j*Hzeros(10));1*exp(1j*Hzeros(11))];

H_poles = [0.95*exp(1j*Hpoles(1));0.95*exp(1j*Hpoles(2));0.95*exp(1j*Hpoles(3));...
        0.95*exp(1j*Hpoles(4));0.95*exp(1j*Hpoles(5));0.95*exp(1j*Hpoles(6));...
        0.95*exp(1j*Hpoles(7));0.95*exp(1j*Hpoles(8));0.95*exp(1j*Hpoles(9));...
        0.95*exp(1j*Hpoles(10));0.95*exp(1j*Hpoles(11));0.95*exp(1j*Hpoles(12));...
        0*exp(1j*Hpoles(13))];
```

$$H(z) = \frac{B(z)}{A(z)}$$

h[n] is the impulse response of a IIR filter which means it is infinite duration. A finite segment of h[n] can be found by using an impulse function as input to the system and repeatedly updating the values of the impulse response by recursion. A finite part of h[n] obtained by the described procedure can be found as a number array which can be seen below.

```
 1.0e+05 *

Columns 1 through 10

      0       0       0       0       0       0       0       0       0       0

Columns 11 through 20

      0       0  0.0000  0.0001  0.0007  0.0028  0.0087  0.0218  0.0455  0.0807

Columns 21 through 30

  0.1208  0.1486  0.1361  0.0515 -0.1254 -0.3836 -0.6664 -0.8706 -0.8702 -0.5622

Columns 31 through 40
```

0.0778 0.9521 1.8344 2.4165 2.4030 1.6287 0.1558 -1.6947 -3.4024 -4.3940

Columns 41 through 50

-4.2353 -2.8037 -0.3745 2.4223 4.7736 5.9341 5.4752 3.4447 0.3759 -2.8625

Columns 51 through 60

-5.3312 -6.3137 -5.5396 -3.2646 -0.1822 2.8013 4.8498 5.4466 4.5305 2.4816

Columns 61 through 70

-0.0269 -2.2545 -3.6195 -3.8521 -3.0375 -1.5452 0.1169 1.4712 2.2036 2.2279

Columns 71 through 80

1.6711 0.7979 -0.0920 -0.7574 -1.0720 -1.0357 -0.7454 -0.3439 0.0324 0.2912

Columns 81 through 90

0.3985 0.3721 0.2612 0.1228 0.0027 -0.0740 -0.1031 -0.0951 -0.0670 -0.0344

Columns 91 through 100

-0.0079 0.0082 0.0145 0.0141 0.0104 0.0062 0.0029 0.0008 -0.0001 -0.0004

Columns 101 through 110

-0.0004 -0.0003 -0.0002 -0.0001 -0.0001 -0.0001 -0.0001 -0.0001 -0.0001 -0.0002

Columns 111 through 120

-0.0003 -0.0004 -0.0005 -0.0005 -0.0002 0.0005 0.0016 0.0030 0.0042 0.0046

Columns 121 through 130

0.0035 0.0005 -0.0043 -0.0098 -0.0143 -0.0157 -0.0125 -0.0042 0.0077 0.0202

Columns 131 through 140

0.0295 0.0317 0.0247 0.0092 -0.0113 -0.0311 -0.0441 -0.0456 -0.0341 -0.0121

Columns 141 through 150

0.0145 0.0380 0.0514 0.0507 0.0358 0.0113 -0.0158 -0.0377 -0.0483 -0.0452

Columns 151 through 160

-0.0301 -0.0080 0.0143 0.0306 0.0371 0.0330 0.0206 0.0045 -0.0105 -0.0204

Columns 161 through 170

-0.0234 -0.0197 -0.0116 -0.0021 0.0061 0.0109 0.0119 0.0096 0.0054 0.0009

Columns 171 through 180

-0.0026 -0.0045 -0.0047 -0.0037 -0.0020 -0.0004 0.0008 0.0014 0.0014 0.0011

Columns 181 through 190

0.0006 0.0002 -0.0001 -0.0002 -0.0003 -0.0002 -0.0001 -0.0001 -0.0000 0.0000

Columns 191 through 200

0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

Columns 201 through 210

0.0000 0.0000 0.0000 0.0000 0.0000 -0.0000 -0.0000 -0.0000 -0.0000 -0.0000

Columns 211 through 220

-0.0000 -0.0000 -0.0000 0.0000 0.0000 0.0001 0.0001 0.0001 0.0000 -0.0000

Columns 221 through 230

-0.0001 -0.0002 -0.0002 -0.0002 -0.0001 0.0000 0.0002 0.0003 0.0004 0.0003

Columns 231 through 240

0.0002 -0.0000 -0.0002 -0.0004 -0.0004 -0.0004 -0.0002 0.0001 0.0003 0.0004

Columns 241 through 250

0.0004 0.0003 0.0001 -0.0001 -0.0002 -0.0003 -0.0003 -0.0002 -0.0001 0.0001

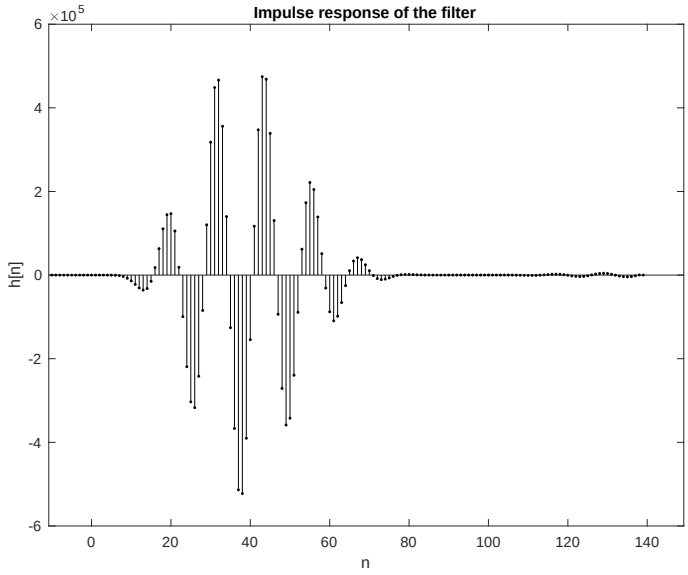
Columns 251 through 260

0.0002 0.0002 0.0002 0.0001 0.0001 -0.0000 -0.0001 -0.0001 -0.0001 -0.0001

Column 261

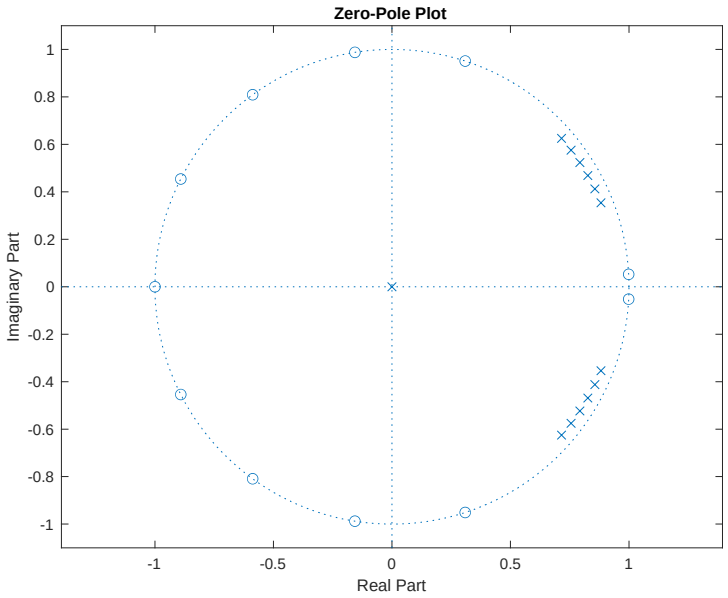
-0.0000

The h[n] is purely real as specified in the lab manual because all zeros and poles have chosen symmetric along the y axis. Plot of h[n] can also be seen below in Fig. 1.



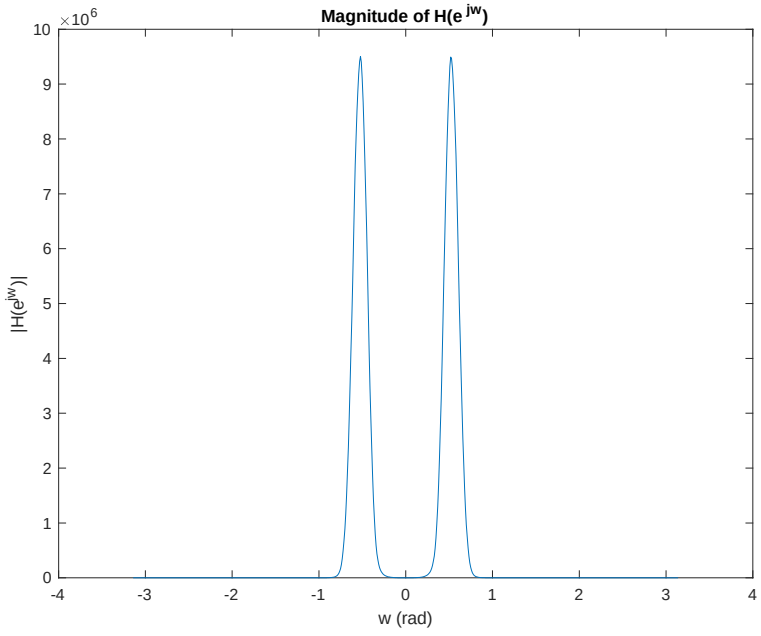
(Fig.1 Plot of h[n])

Poles and zeros chosen to satisfy the specification of the bandpass filter can be seen in Fig.2.

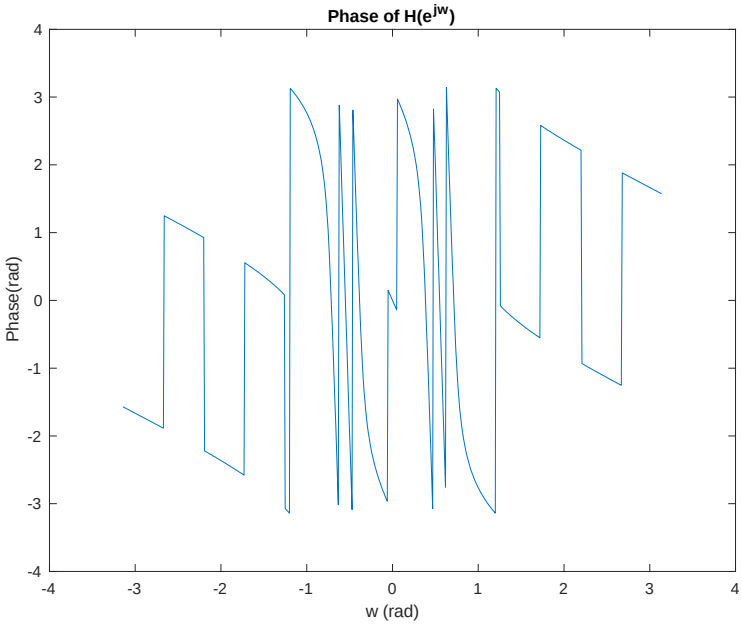


(Fig.2 Poles and Zeros of H(z) chosen for the best separation of stopband and passband region)

Also plot of $\left|H(e^{jw})\right|$ and $\measuredangle H(e^{jw})$ can be found in Fig.3 and Fig.4 respectively. Those plots are found by inserting e^{jw} to the Z transform of the impulse response H(z).



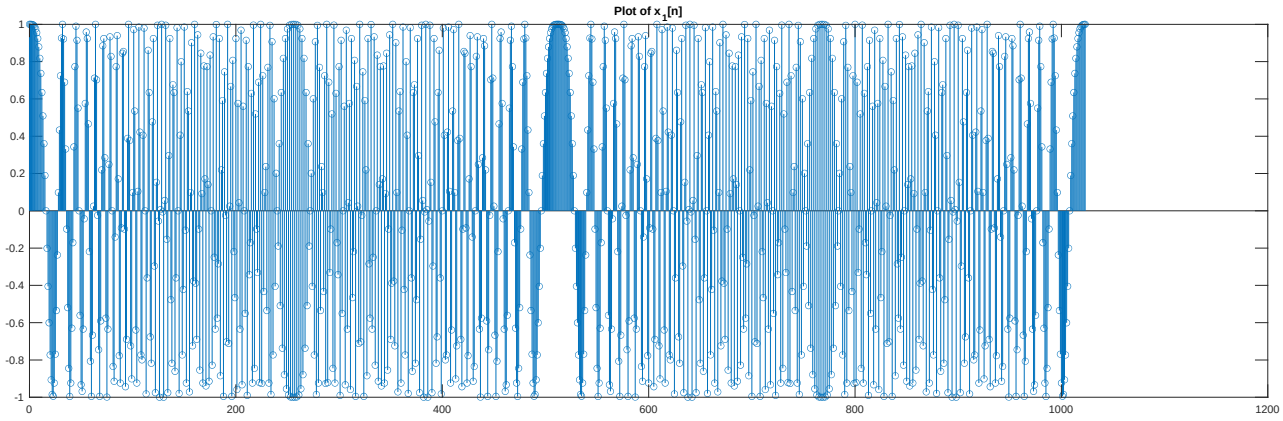
(Fig.3 Magnitude of Discrete Time Fourier Transform of h[n])



(Fig.4 Phase of Discrete Time Fourier Transform of h[n])

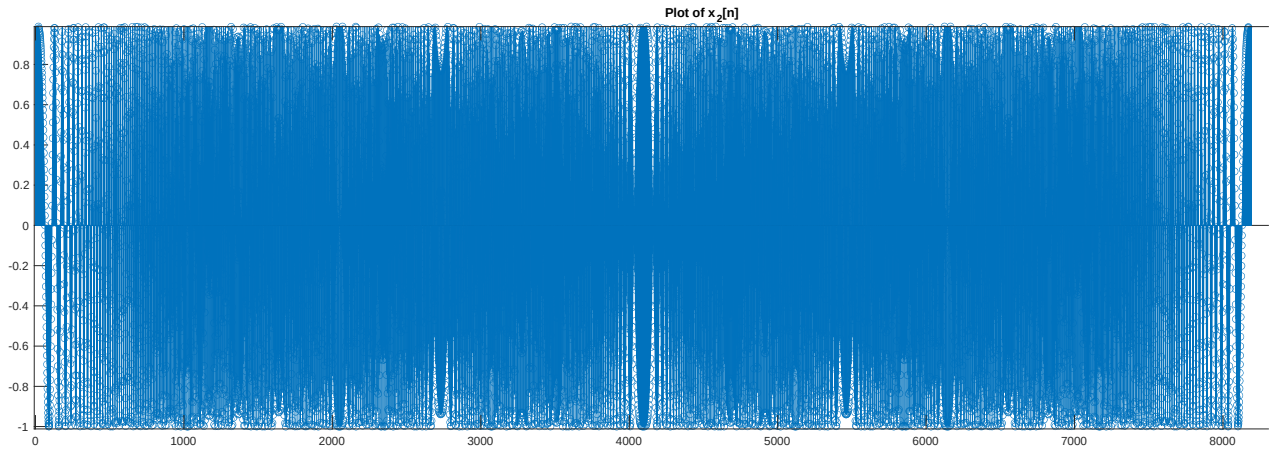
2)

The sampled chirp signal with sampling frequency $\sqrt{\frac{\pi}{\alpha \cdot 512}}$ with $\alpha = 1000$ is denoted as $x_1[n]$. Plot of $x_1[n]$ for $n \in [0,1023]$ can be seen in Fig.5



(Fig.5 Plot of $x_1[n] = \cos\left(\frac{\pi n^2}{512}\right)$ for n∈ [0: 1023])

$x_2[n]$ is the discrete time signal where $x_2[n] = \cos(\alpha(n \cdot T_s)^2)$ and $n \in [0: 8192]$. $T_s = 1000\ rad^{-2}$ for this signal. Plot of this signal can be seen in Fig.6.



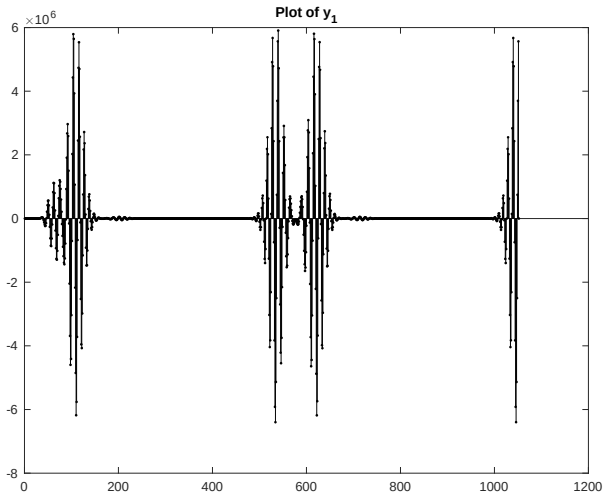
(Fig.6 Plot of $x_2[n] = \cos\left(\frac{\pi n^2}{8192}\right)$ for n∈ [0: 8192])

3)

Then the generated $x_1[n]$ signal is used in the recursion loop which is defined below to find the output for a predefined duration. The whole difference equation representing the IIR filter is shown below:

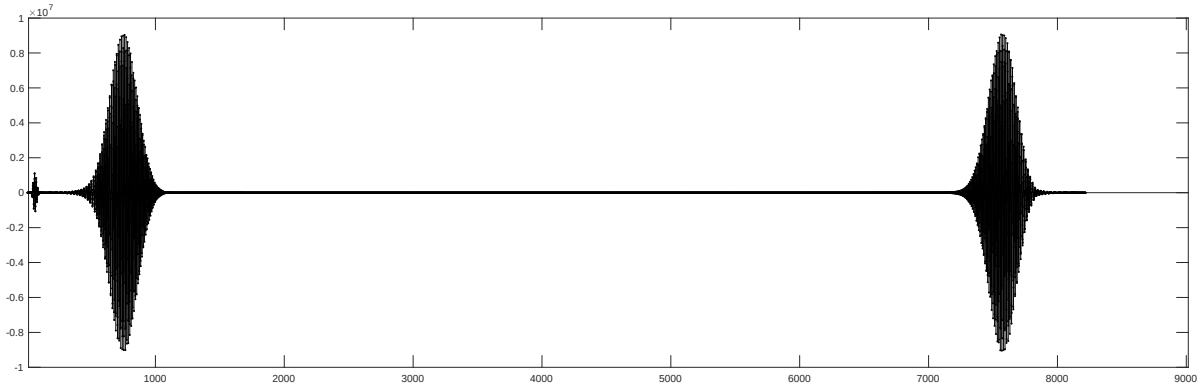
$$y[n] = -\sum_{k=1}^{13} A(k+1) \cdot y[n-k] + \sum_{l=0}^{10} B(k+1) \cdot x[n-k]$$

Output of this IIR filter with input as a sampled chirp signal with sampling period 2.48 ms can be seen in Fig.7.



(Fig.7 Output of the IIR filter ($y_1[n]$) which has input as $x_1[n]$)

Also, the same filter is being used with a chirp signal with higher sampling rate as 0.62 ms. Output of this filtering clearly shows the passband of the filter with the dark areas.



(Fig.8 Output of the IIR filter ($y_2[n]$) which has input as $x_2[n]$)

Output of this filter represents the response of the filter to the frequency values of given input signals because of the nature of the chirp signal. The chirp signal $cos(\alpha t^2)$ has instantaneous frequency of $2\alpha t$ which represents the time instance where the frequency is examined. Therefore, instantaneous frequency of the chirp signal is exactly equal to the point it is examined when $\alpha = \frac{1}{2}$. The output of the filter obtained by using the chirp signal as input basically gives the behavior of the filter for every time instance t. Therefore, $y_1[n]$ represents the frequency response of the IIR filter.

Resolution of the frequency response of the system is determined by the sampling rate used in the chirp signal which can clearly be seen from the respective outputs of the filter to $x_1[n]$ and $x_2[n]$. The sampling rate must be chosen as a high frequency in order to ensure that the frequency response of the filter is being represented accurately.

Chirp signal is not a stable signal. Therefore, convolving it with another signal may result in divergence, so one must be careful when using chirp signal as the input to the system. The system representing the IIR filter must be stable to be convolved with the chirp signal.

4)

$x_a(t)$ is not a periodic signal but $x_r(t)$ is generated by using a periodic sampling of the signal $x_a(t)$ so, the sounds that are being heard are not the same since one is periodic and the other is not. This can also be understood when listening both signals as audio. The original chirp signal has a constant increasing frequency therefore the generated sound gets higher toned through time. But the signal generated by sampling the original signal is periodic therefore the pitch of the sound generated by that signal increases and decreases periodically.

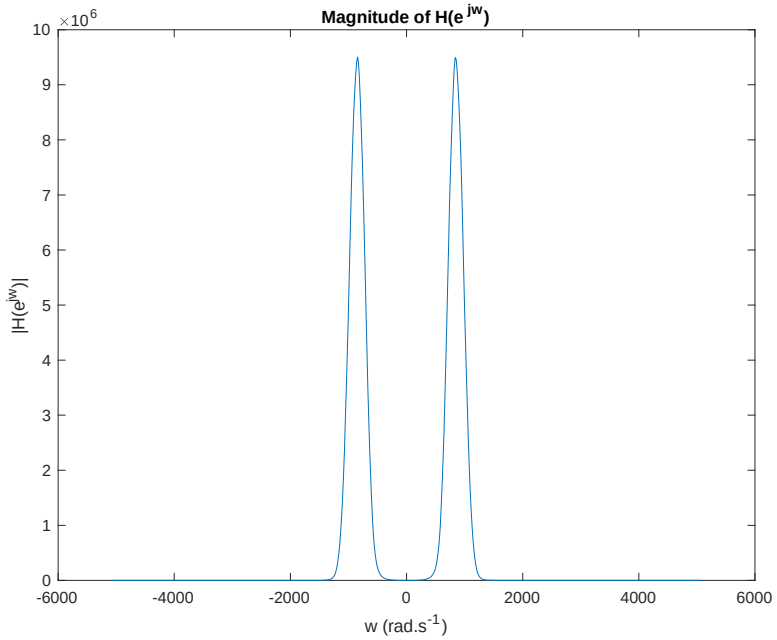
5)

$y_r(t)$ and $y_2[n]$ are similar signals since both are periodic. $y_r(t)$ is generated by using the discrete version of the signal and using a sample rate. Each instance of the discrete signal lasts for the sampling rate and therefore a continuous can be generated by this procedure on D/A converter of the which is also called as interpolation. Both $y_r(t)$ and $y_2[n]$ are signals that has repeated sounds which makes them similar.

6)

Cut-off frequencies of the impulse response of the system that produces $y_r(t)$ from $x_a(t)$ are the same with the designed filter at the first. So, cut-off frequencies are $\frac{\pi}{10}$, $\frac{\pi}{4}$ and the symmetric of them according to the y axis that are $-\frac{\pi}{10}$ and $-\frac{\pi}{4}$. Those cutoff frequencies can

also be seen from the frequency response plot of the analog system which is available as Fig. 9.



(Fig.9 Plot of $|H(jw)|$ for the signal $y_r(t)$)

7)

This filter allows the pass of components of a signal that lay in the bandpass domain of the filter. Therefore, only sounds within a small range of frequency band can be heard from the generated output sound file. This makes the output sound sharper and cleaner since many frequencies are removed from the recording by suppression effect of the filter on the stopbands. Therefore, most of the information that the input signal carries remain in the output but there are slight changes in the sound of the signal which result from the properties of the designed IIR filter.

MATLAB CODES:

```
%School number is 22002840
% N1 =  8  || N2 =  2
% M1 = 10  || M2 =  4
% Order = 13
% Cutoff's between pi/10 and pi/4
% Causal

% PART 1

%i) Determination of the pole locations:

p0 = pi/10 + 1 * ((3*pi/20)/7);
p1 = pi/10 + 2 * ((3*pi/20)/7);
p2 = pi/10 + 3 * ((3*pi/20)/7);
p3 = pi/10 + 4 * ((3*pi/20)/7);
p4 = pi/10 + 5 * ((3*pi/20)/7);
p5 = pi/10 + 6 * ((3*pi/20)/7);

p6 = -pi/10 -1 * ((3*pi/20)/7);
p7 = -pi/10 -2 * ((3*pi/20)/7);
p8 = -pi/10 -3 * ((3*pi/20)/7);
p9 = -pi/10 -4 * ((3*pi/20)/7);
p10 = -pi/10 -5 * ((3*pi/20)/7);
p11 = -pi/10 -6 * ((3*pi/20)/7);
p12 = 0;

Hpoles = [p0,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12];

%Determination of zero locations
z0 = pi/60;
z1 = pi/4 + 1 * ((3*pi/2)/10);
z2 = pi/4 + 2 * ((3*pi/2)/10);
z3 = pi/4 + 3 * ((3*pi/2)/10);
z4 = pi/4 + 4 * ((3*pi/2)/10);
z5 = pi/4 + 5 * ((3*pi/2)/10);
z6 = -pi/4 -1 * ((3*pi/2)/10);
z7 = -pi/4 -2 * ((3*pi/2)/10);
z8 = -pi/4 -3 * ((3*pi/2)/10);
z9 = -pi/4 -4 * ((3*pi/2)/10);
z10 = -pi/60;

Hzeros = [z0,z1,z2,z3,z4,z5,z6,z7,z8,z9,z10];



H_zeros = [1*exp(1j*Hzeros(1));1*exp(1j*Hzeros(2));1*exp(1j*Hzeros(3));...
           1*exp(1j*Hzeros(4));1*exp(1j*Hzeros(5));1*exp(1j*Hzeros(6));...
           1*exp(1j*Hzeros(7));1*exp(1j*Hzeros(8));1*exp(1j*Hzeros(9));...
           1*exp(1j*Hzeros(10));1*exp(1j*Hzeros(11))];

H_poles = [0.95*exp(1j*Hpoles(1));0.95*exp(1j*Hpoles(2));0.95*exp(1j*Hpoles(3));...
           0.95*exp(1j*Hpoles(4));0.95*exp(1j*Hpoles(5));0.95*exp(1j*Hpoles(6));...
           0.95*exp(1j*Hpoles(7));0.95*exp(1j*Hpoles(8));0.95*exp(1j*Hpoles(9));...
           0.95*exp(1j*Hpoles(10));0.95*exp(1j*Hpoles(11));0.95*exp(1j*Hpoles(12));...
           0*exp(1j*Hpoles(13))];

figure(4)
zplane(H_zeros, H_poles);
title('Zero-Pole Plot');
syms z
```

```
B(z) = (z-H_zeros(1))*(z-H_zeros(2))*(z-H_zeros(3))*(z-H_zeros(4))*(z-H_zeros(5))*(z-H_zeros(6))*(z-H_zeros(7))*...
    (z-H_zeros(8))*(z-H_zeros(9))*(z-H_zeros(10))*(z-H_zeros(11));

A(z) = (z-H_poles(1))*(z-H_poles(2))*(z-H_poles(3))*(z-H_poles(4))*(z-H_poles(5))*(z-H_poles(6))*(z-H_poles(7))*...
    (z-H_poles(8))*(z-H_poles(9))*(z-H_poles(10))*(z-H_poles(11))*(z-H_poles(12));


H(z) = B(z)/A(z);

w = -pi:0.01:pi;
figure(2)
plot(w,abs(double(H(exp(1j*w)))));
xlabel('w (rad)');
ylabel('|H(e^{jw})|');
title('Magnitude of H(e^{jw})');

figure(3)
plot(w,angle(double(H(exp(1j*w)))));
xlabel('w (rad)');
ylabel('Phase(rad)');
title('Phase of H(e^{jw})');

nom = round(double(coeffs(B(z),z)),8);
denom = round(double(coeffs(A(z),z)),8);

[A_coeff, B_coeff] = zp2tf(H_zeros, H_poles,1/0.95);

imps = zeros(1,250);
imps(11) = 1;
y_1 = zeros(1,250);
for i=-10:238
    if(i>=3)
```

```
    y_1(i+11) = -B_coeff(2)*y_1(i+10) - B_coeff(3)*y_1(i+9) - B_coeff(4)*y_1(i+8) - B_coeff(5)*y_1(i+7)...
    -B_coeff(6)*y_1(i+6) - B_coeff(7)*y_1(i+5) - B_coeff(8)*y_1(i+4) - B_coeff(9)*y_1(i+3) - B_coeff(10)*y_1(i+2)...
    -B_coeff(11)*y_1(i+1)-B_coeff(12)*y_1(i)-B_coeff(13)*y_1(i-1)-B_coeff(14)*y_1(i-2)+...
    A_coeff(4)*imps(i+11) + A_coeff(5)*imps(i+10) + A_coeff(6)*imps(i+9) + A_coeff(7)*imps(i+8) + A_coeff(8)*imps(i+7) + ...
    A_coeff(9)*imps(i+6) + A_coeff(10)*imps(i+5) +A_coeff(11)*imps(i+4) +A_coeff(12)*imps(i+3) +A_coeff(13)*imps(i+2) +A_coeff(14
    else
    y_1(i+11) = 0;
    end
end

hn = y_1;
save('hn.mat','hn')
load("hn.mat",'hn')
figure(1.)
stem(-10:239,hn,'filled','.k')
title('Impulse response of the filter')
```

```
%PART 2

syms t

xa(t) = cos(1000*t^2);
n1 = 0:1023;
T1 = sqrt(pi/(1000*512));
n2 = 0:8191;
T2 = sqrt(pi/(1000*8192));
x1 = xa(n1.*T1);
x2 = xa(n2.*T2);

figure(5)
stem(n1,x1)
title('Plot of x_{1}[n]')

figure(6)
stem(n2,x2)
title('Plot of x_{2}[n]')

%PART 3
%{
x1_n = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 x1];
y_1 = zeros(1,1052);
for i=-10:1040
    if(i>=3)
    y_1(i+11) = -B_coeff(2)*y_1(i+10) - B_coeff(3)*y_1(i+9) - B_coeff(4)*y_1(i+8) - B_coeff(5)*y_1(i+7)...
    -B_coeff(6)*y_1(i+6) - B_coeff(7)*y_1(i+5) - B_coeff(8)*y_1(i+4) - B_coeff(9)*y_1(i+3) - B_coeff(10)*y_1(i+2)...
    -B_coeff(11)*y_1(i+1)-B_coeff(12)*y_1(i)-B_coeff(13)*y_1(i-1)-B_coeff(14)*y_1(i-2)+...
    A_coeff(4)*x1_n(i+11) + A_coeff(5)*x1_n(i+10) + A_coeff(6)*x1_n(i+9) + A_coeff(7)*x1_n(i+8) + A_coeff(8)*x1_n(i+7) +
    A_coeff(9)*x1_n(i+6) + A_coeff(10)*x1_n(i+5) +A_coeff(11)*x1_n(i+4) +A_coeff(12)*x1_n(i+3) +A_coeff(13)*x1_n(i+2) +A_
    else
    y_1(i+11) = 0;
```

```
end
save('y_1.mat','y_1')
%}
load("y_1.mat",'y_1')
figure(7)
stem(y_1,'filled','.k')
title('Plot of y_{1}')


x2_n = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 x2];
%{
y_2 = zeros(1,8208);
for i=-10:8208
    if(i>=3)
    y_2(i+11) = -B_coeff(2)*y_2(i+10) - B_coeff(3)*y_2(i+9) - B_coeff(4)*y_2(i+8) - B_coeff(5)*y_2(i+7)...
    -B_coeff(6)*y_2(i+6) - B_coeff(7)*y_2(i+5) - B_coeff(8)*y_2(i+4) - B_coeff(9)*y_2(i+3) - B_coeff(10)*y_2(i+2)...
    -B_coeff(11)*y_2(i+1)-B_coeff(12)*y_2(i)-B_coeff(13)*y_2(i-1)-B_coeff(14)*y_2(i-2)+...
    A_coeff(4)*x2_n(i+11) + A_coeff(5)*x2_n(i+10) + A_coeff(6)*x2_n(i+9) + A_coeff(7)*x2_n(i+8) + A_coeff(8)*x2_n(i+7) +
    A_coeff(9)*x2_n(i+6) + A_coeff(10)*x2_n(i+5) +A_coeff(11)*x2_n(i+4) +A_coeff(12)*x2_n(i+3) +A_coeff(13)*x2_n(i+2) +A_
    else
    y_2(i+11) = 0;
    end
end
save('y_2.mat','y_2')
%}
load('y_2.mat','y_2')
figure(8)
stem(y_2,'filled','.k')
x2_n = double(x2_n);
save('x2_n.mat','x2_n')
```

```
%PART 6

figure(9)
w = -pi:0.01:pi;
plot(w/(T2),abs(double(H(exp(1j*w)))));
xlabel('w (rad.s^{-1})');
ylabel('|H(e^{jw})|');
title('Magnitude of H(e^{jw})');
```

```
load("y_2.mat","y_2");
a = 1000;
Ts=sqrt(pi/(a*8207));
samplerate = 1/Ts;

player = audioplayer((y_2), samplerate);
period = Ts.*length(y_2);
 while(1)
     play(player);
     pause(period);
     stop(player);
 end
```

```
[selfaudio, samplerate] = audioread('selfaudio.m4a');

x2_n = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 selfaudio'];

y_2 = zeros(1,340928);
for i=-10:340928
    if(i>=3)
    y_2(i+11) = -denom_coeff(2)*y_2(i+10) - denom_coeff(3)*y_2(i+9) - denom_coeff(4)*y_2(i+8) - denom_coeff(5)*y_2(i+7)...
    -denom_coeff(6)*y_2(i+6) - denom_coeff(7)*y_2(i+5) - denom_coeff(8)*y_2(i+4) - denom_coeff(9)*y_2(i+3) - denom_coeff(1
    -denom_coeff(11)*y_2(i+1)-denom_coeff(12)*y_2(i)-denom_coeff(13)*y_2(i-1)-denom_coeff(14)*y_2(i-2)+...
    nom_coeff(4)*x2_n(i+11) + nom_coeff(5)*x2_n(i+10) + nom_coeff(6)*x2_n(i+9) + nom_coeff(7)*x2_n(i+8) + nom_coeff(8)*x2_
    nom_coeff(9)*x2_n(i+6) + nom_coeff(10)*x2_n(i+5) +nom_coeff(11)*x2_n(i+4) +nom_coeff(12)*x2_n(i+3) +nom_coeff(13)*x2_n
    else
    y_2(i+11) = 0;
    end
end

figure(10)
plot(selfaudio)
audiowrite('IIRself.m4a',y_2*10^(-6),samplerate)
```