EE 313 Linear Signals & Systems

Solution Set for Mini-Project #2 on Digital Audio Processing

By: Anyesha Ghosh & Prof. Brian L. Evans

1.1. The code below is used to record sound from the microphone. It is directly taken from the problem statement for Mini-Project #2.

```
fs = 44100;
numBits = 16;
numChannels = 1;
recordingTime = 5;
recObj = audiorecorder(fs, numBits, numChannels);
disp('Start recording...');
recordblocking(recObj, recordingTime);
disp('End recording.');
% Store data in double-precision floating-point array
myRecording = getaudiodata(recObj);
% Play back the recording with automatic scaling
soundsc(myRecording, fs);
% Plot the waveform in the time domain
figure; plot(myRecording);
% Plot the spectrogram
figure;
spectrogram(myRecording, hamming(16384), 8192, 16384, fs, 'yaxis');
```

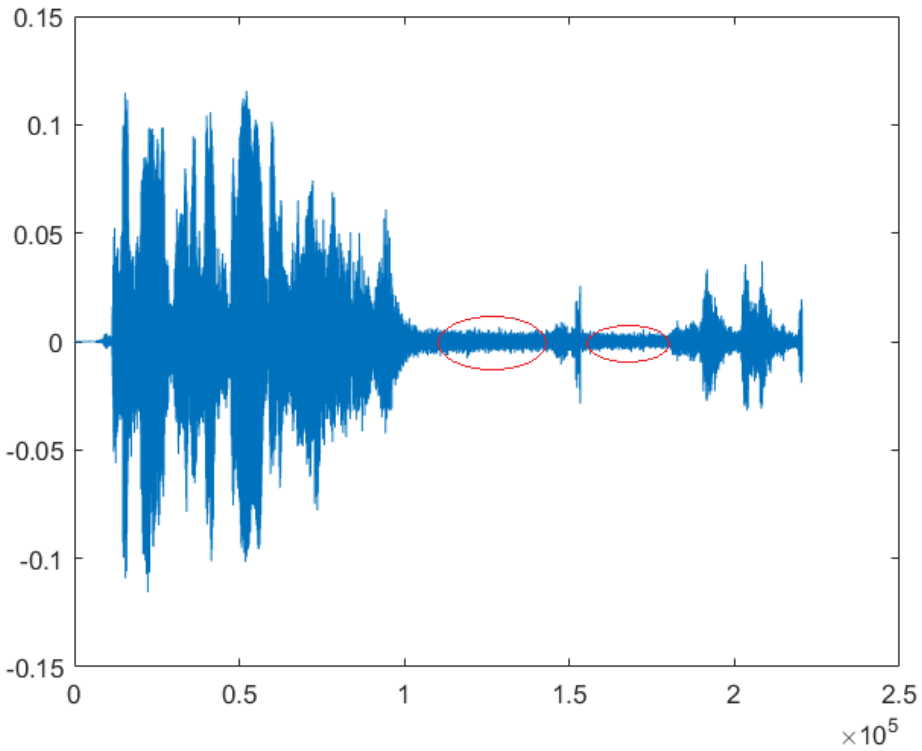


Fig. 1. Time domain plot of recorded voice

As can be seen in Fig. 1, the average value of the signal is approximately 0. This is confirmed by taking the mean of the signal. The quiet portions of the recording are circled in red.

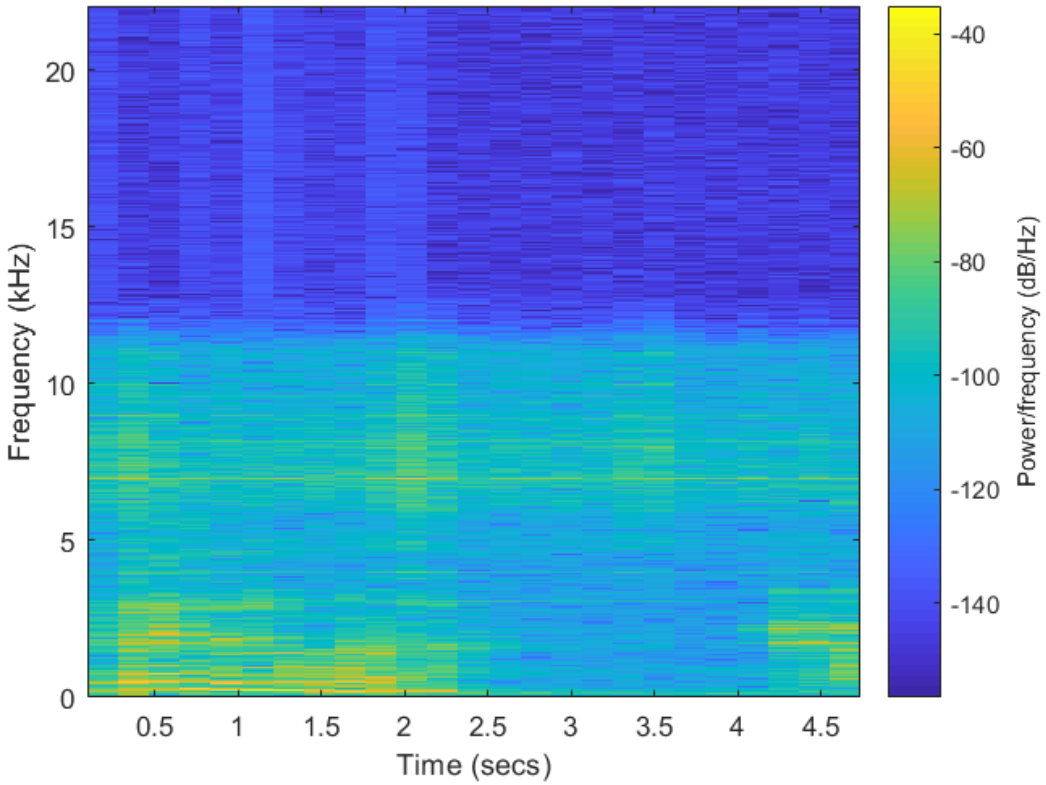


Fig. 2. Spectrogram generated from the code above.

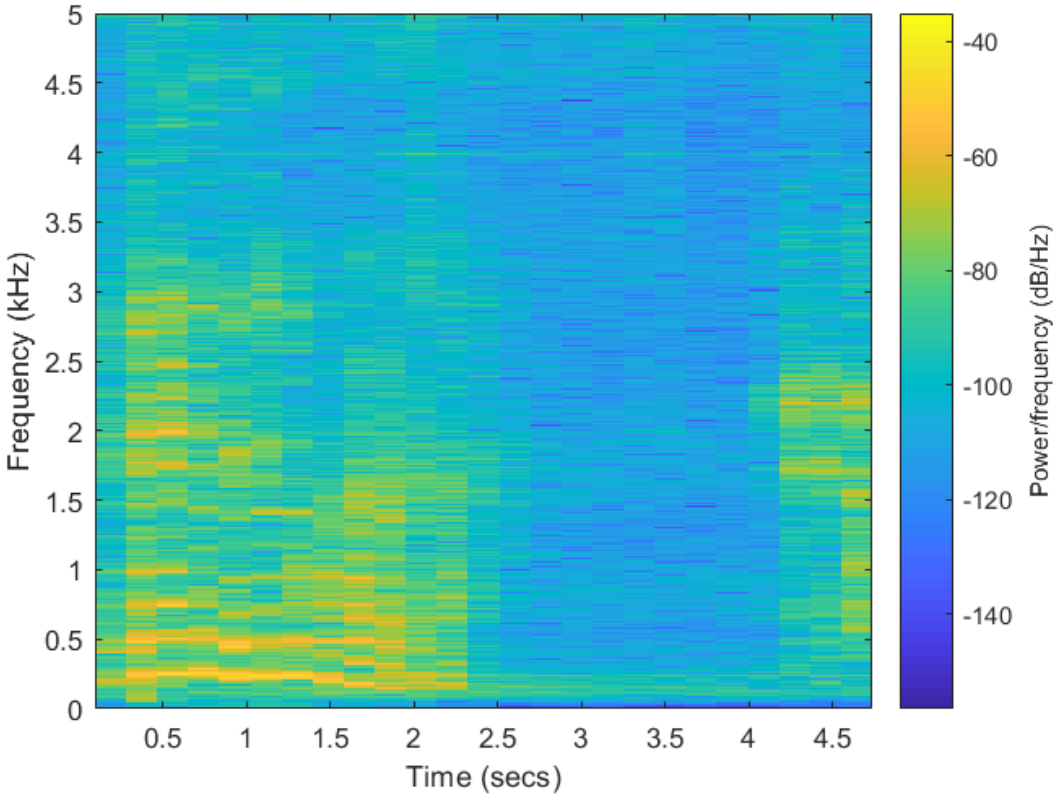


Fig.3. Magnified View (1) of the above spectrogram.

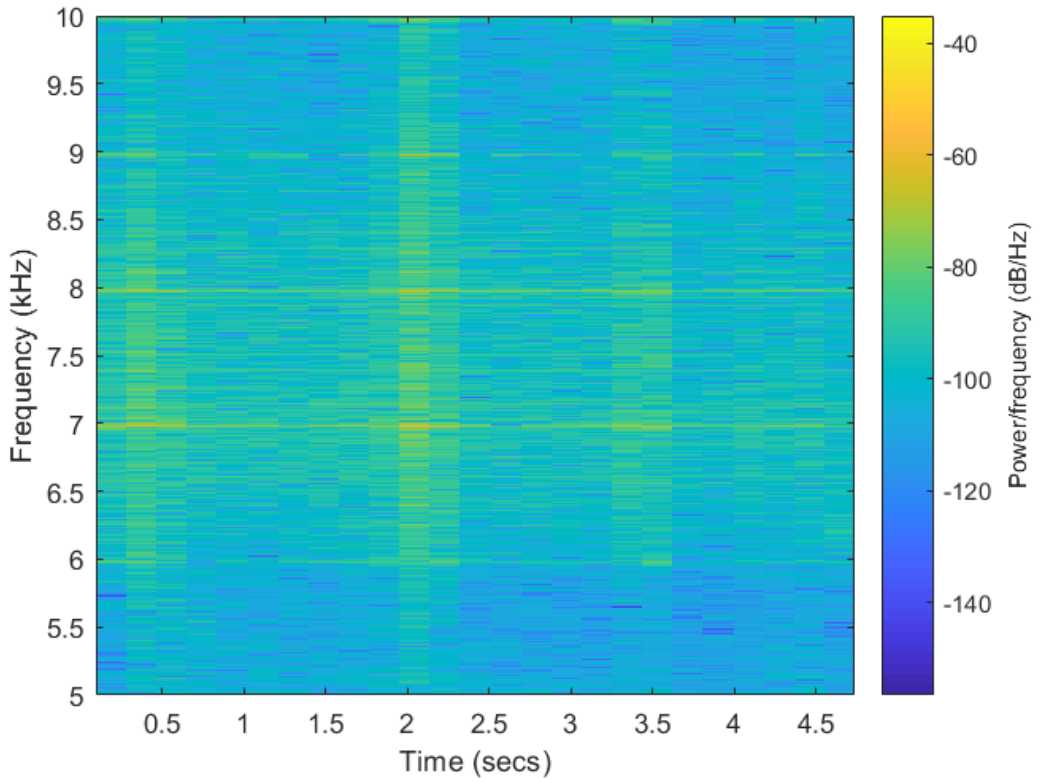


Fig. 4. Magnified view (2) of the above spectrogram.

In the spectrogram, the principal frequencies are those with the highest power density, which are represented by bright yellow in the spectrogram. We expect to see bright bands at the principal frequencies & their harmonics. Here, we see the principal frequencies to be (approximately): 250Hz, 400Hz, 750Hz, 2kHz, 7kHz. We also see the entire spectrum dropping to the blue regions (low power) at t = [2.5s, 4s], which corresponds to the quiet regions observed in the time domain plot.

In English, vowel sounds generally have a harmonic structure. We had seen this with the 'ah' sound in Section 3-3.1 and lecture slide 3-6. An opposite effect can occur from pronouncing 's' which sounds noise-like; i.e. it has a wide band of frequencies in it without a harmonic structure. There are many phonemes in English that have a mixture of the two extremes.

1.2. The code used for this part is taken from the Mini-Project #2 problem statement. The audio track is "Nightbook" by Ludovico Einaudi (https://www.youtube.com/watch?v=OB3wgiaOOvA).

```
waveFilename = 'nightbook_einaudi.mp3'; % Pick an audio filename
[y,Fs] = audioread(waveFilename);
soundsc(y, fs);
spectrogram(y(:,1), hamming(16384), 8192, 16384, fs, 'yaxis');
```

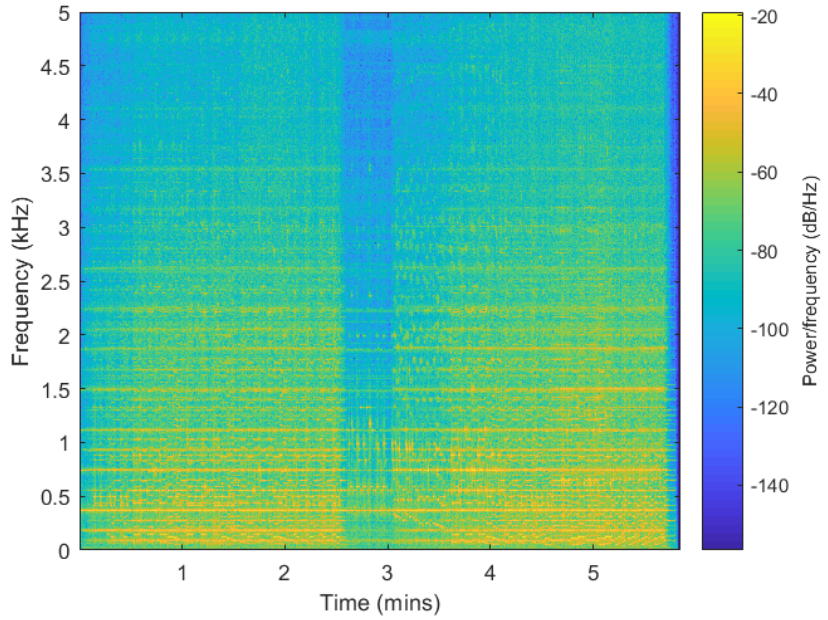


Fig.5. Magnified spectrogram for the music signal.

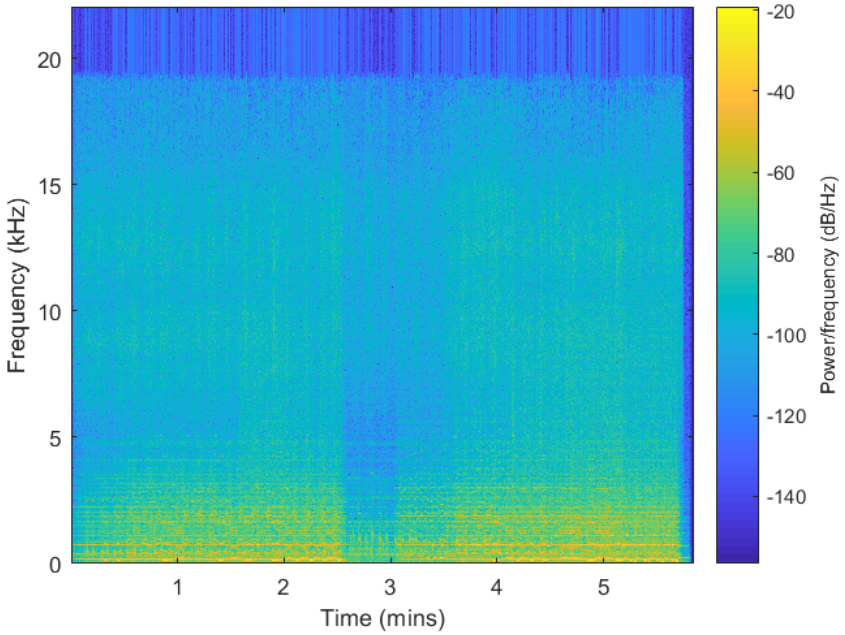


Fig. 6. Complete spectrogram for the music signal.

We see that the spectrogram repeats about halfway through the sound, which makes sense, considering that a large part of the song is repeated in the file. We see many principal frequencies, which are mostly clustered in the [0, 1.5kHz] band. Look for the bright yellow lines in Fig. 5.

1.3. The code used for this part is (taken directly from the problem statement):

```
% Sampling rate and time
fs = 44100;
Ts = 1/fs;
% Generate a chirp signal
tmax = 5;
t = 0 : Ts : tmax;
```

```
fstart = 20;
fend = 4000;
y = chirp(t, fstart, tmax, fend);
% Play back the chirp signal
soundsc(y, fs);
% Spectrogram
spectrogram(y, hamming(16384), 8192, 16384, fs, 'yaxis');
```

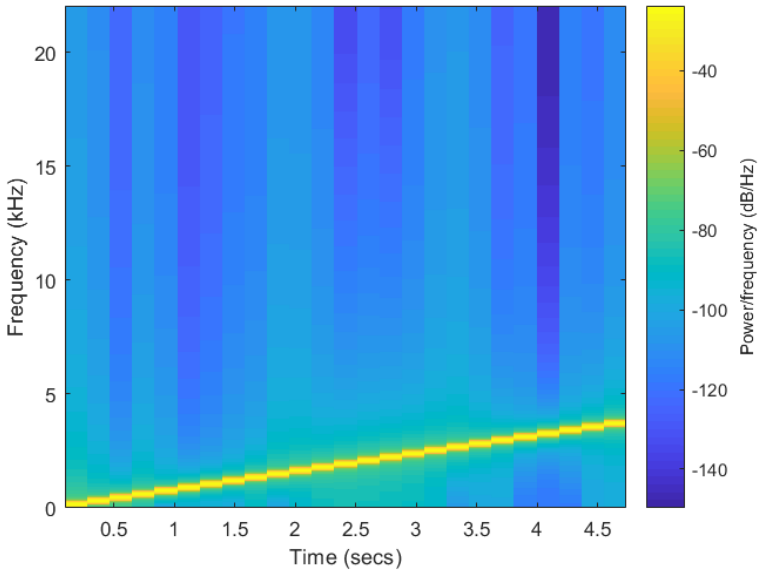


Fig. 7. Spectrogram of a chirp signal.

Here, we see a pattern of increasing principal (and sole, in this case) frequency. This is exactly what is expected as we are dealing with a chirp signal.

2. I'm using the song above for this as well- since most of the class seems to be doing this.

a) A delay of one sample corresponds to a delay of $T_s = \frac{1}{f_s}$. So, for a delay of 0.15s and a sampling rate of 11025 Hz, we need to solve for

$$0.15s = \frac{P}{f_s} \Rightarrow P = f_s * 0.15 = 11025 \frac{\text{samples}}{\text{s}} * 0.15s = 1653.75 \sim 1654 \text{ samples:}$$

$$y[n] = \frac{1}{1+\alpha} * x[n] + \frac{\alpha}{1+\alpha} * x[n-p] = \frac{1}{1.95} * x[n] + \frac{0.95}{1.95} * x[n-P]$$

$$\therefore y[n] = 0.5128\, x[n] + 0.4872\, x[n-1654]$$

So, $h[n] = 0.5128\delta * \delta[n] + 0.4872 * \delta[n-1654]$. Hence, the filter coefficients are:

$$h[n] = \begin{cases} 0.5128, & n = 0 \\ 0.4872, & n = 1654 \\ 0, & else \end{cases}$$

The stem and frequency response plots are generated using the following code:

```
P = 1654;
H = zeros(P+1,1);
H(1) =  0.5128;
H(P+1)= 0.4872;
stem(H);
figure;
freqz(H);
```

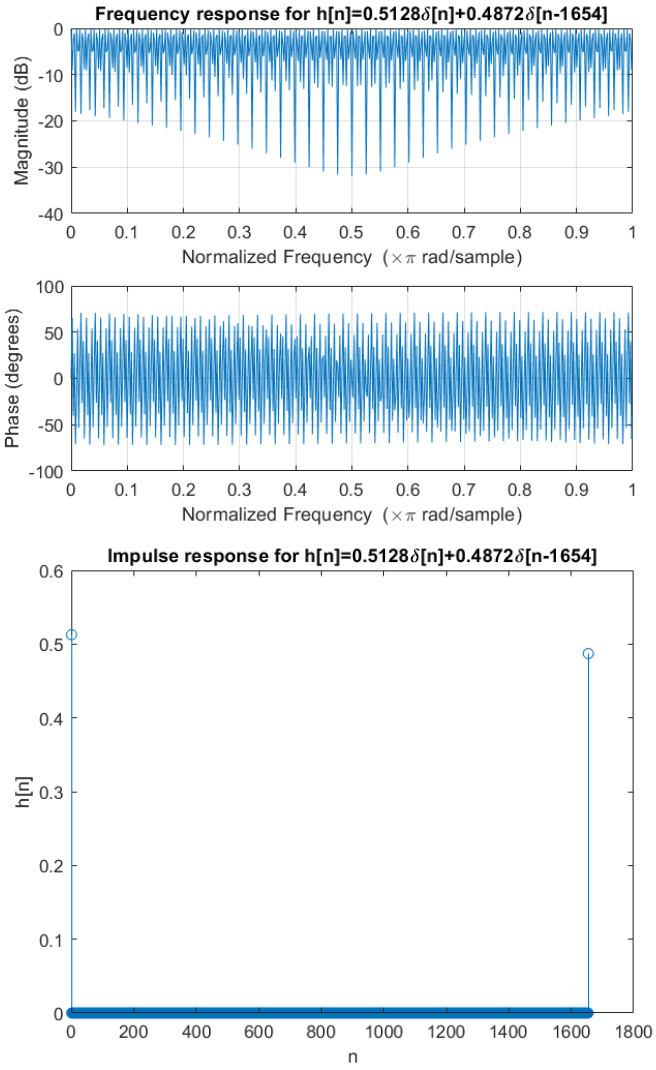


Fig.8. Frequency & impulse responses for h[n] (single echo filter).

b)

```
filename = 'nightbook_einaudi.mp3';
% Reading in only part of the file. This is useful when dealing with
% really long files.  x has two columns, representing the two audio
% channels (left and right). Either of the channels will work for us.
[x,fs] = audioread(filename,[1,2250000]);
p = ceil(fs*0.15); %Number of samples to produce a delay of 0.15s.
alpha = 0.95;
% Construct a single echo filter.
h = zeros(p+1,1);
h(1) = 1/(1+alpha);
h(p+1) = alpha/(1+alpha);
% Output of the single echo filter.
y = conv(h,x(:,1)); %Taking just one of the channels.
soundsc(y, fs);
audiowrite('einaudi_echo1.wav',y/abs(max(y)),fs)
```

The filtered signal sounds like the original overlapped with a loud echo. This makes sense, as the filter adds a significantly delayed version of the original signal to itself, which sounds like an echo.

c) $h[n] = 0.5128\,\delta[n] + 0.4872\,\delta[n - 1654]$

$$= a\,\delta[n] + b\,\delta[n-P]\,, for\ a = 0.5128, b = 0.4872, P = 1654.$$

Let $h_1[n]$:= impulse response of cascade of two filters, $h_f[n]$:= impulse response of entire cascade.

$$h_1[n] = h[n] * h[n] = \sum_m (a\,\delta[m] + b\,\delta[m-P])(a\,\delta[n-m] + b\delta\,[n-m-P])$$

$$= \sum_m a\,\delta[m]\,(a\,\delta[n-m] + b\,\delta[n-m-P]) + \sum_m b\delta\,[m-k](a\,\delta[n-m] + b\,\delta[n-m-P])$$

$$= a\,(a\,\delta[n] + b\,\delta[n-P]) + b\,(a\,\delta[n-P] + b\,\delta[n-2P])$$

$$= a^2\,\delta[n] + 2ab\,\delta[n-P] + b^2\,\delta[n-2P]$$

$$h_f[n] = h_1[n] * h_1[n]$$

$$= \sum_m (a^2\delta[m] + 2ab\delta[m-P] + b^2\delta[m-2P])(a^2\,\delta[n-m] + 2ab\,\delta[n-m-P] + b^2\delta[n-m-2P])$$

$$= \sum_m a^2\delta[m](a^2\delta[n-m] + 2ab\delta[n-m-P] + b^2\delta[n-m-2P]) + \sum_m 2ab\delta[m-P](a^2\delta[n-m] + 2ab\delta[n-m-P] + b^2\delta[n-m-2P]) + \sum_m b^2\delta[m-2P](a^2\delta[n-m] + 2ab\delta[n-m-P] + b^2\delta[n-m-2P])$$

$$= a^2(a^2\delta[n] + 2ab\delta[n-P] + b^2\delta[n-2P]) + 2ab(a^2\delta[n-P] + 2ab\delta[n-2P] + b^2\delta[n-3P]) + b^2(a^2\delta[n-2P] + 2ab\delta[n-3P] + b^2\delta[n-4P])$$

$$= a^4\,\delta[n] + 4a^3b\,\delta[n-P] + 6a^2b^2\,\delta[n-2P] + 4ab^3\,\delta[n-3P] + b^4\,\delta[n-4P].$$

So, the net impulse response of the cascaded system is:

$$h_f[n] = a^4\delta[n] + 4a^3b\,\delta[n-P] + 6a^2b^2\,\delta[n-2P] + 4ab^3\,\delta[n-3P] + b^4\,\delta[n-4P]$$

$\therefore h_f[n] = 0.06915\,\delta[n] + 0.2628\,\delta[n-1654] + 0.3745\,\delta[n-3308] + 0.2372\,\delta[n-4962] + 0.05634\,\delta[n-6616]$.

We can see that the coefficients of the component terms of $h_f[n]$ are identical to the terms in the expansion of $(a+b)^4$. This is not a coincidence, and you can see that by calculating $h_f[n]$ by going through the *z*-domain, and noting that convolution in the time domain corresponds to multiplication in the *z*-domain.

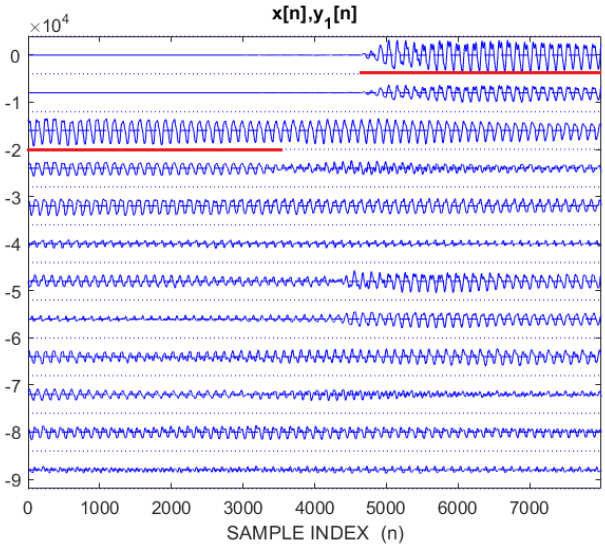
d)

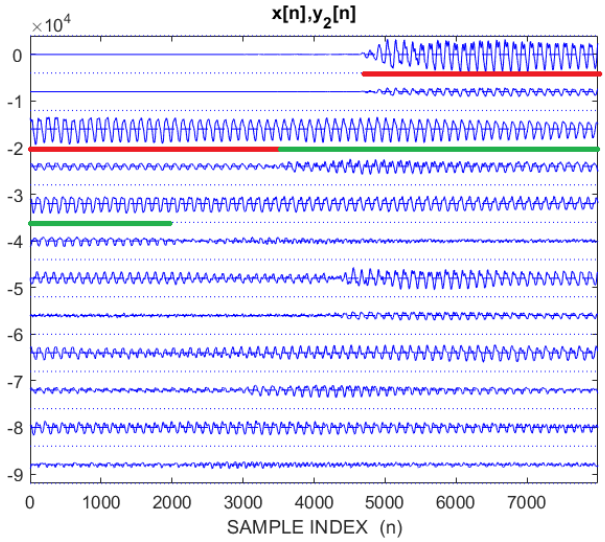
```
filename = 'nightbook_einaudi.mp3';
[x,fs] = audioread(filename,[1,2250000]);

p = ceil(fs*0.15); %Number of samples to produce a delay of 0.15s.
alpha = 0.95;
%Using explicit time domain equation instead of filter() or conv().
```

```
y_prev = x(:,1);
for i=1:1:4
    y1 = circshift(y_prev,p);
    y1(1:p) = 0;
    y = (1/(1+alpha))*y_prev+(alpha/(1+alpha))*y1;

    figure;
    inout(x(:,1),y,30000,8000,6);
    y_prev = y;
end

soundsc(y, fs);
audiowrite('einaudi_echo2.wav',y/abs(max(y)),fs)
```

The final $y[n]$ has very strong echoes (because of the large value of alpha, and the fact that we're adding 4 echoes to the initial signal). That comes across in the sound file, which can barely be identified as the original song.

e)





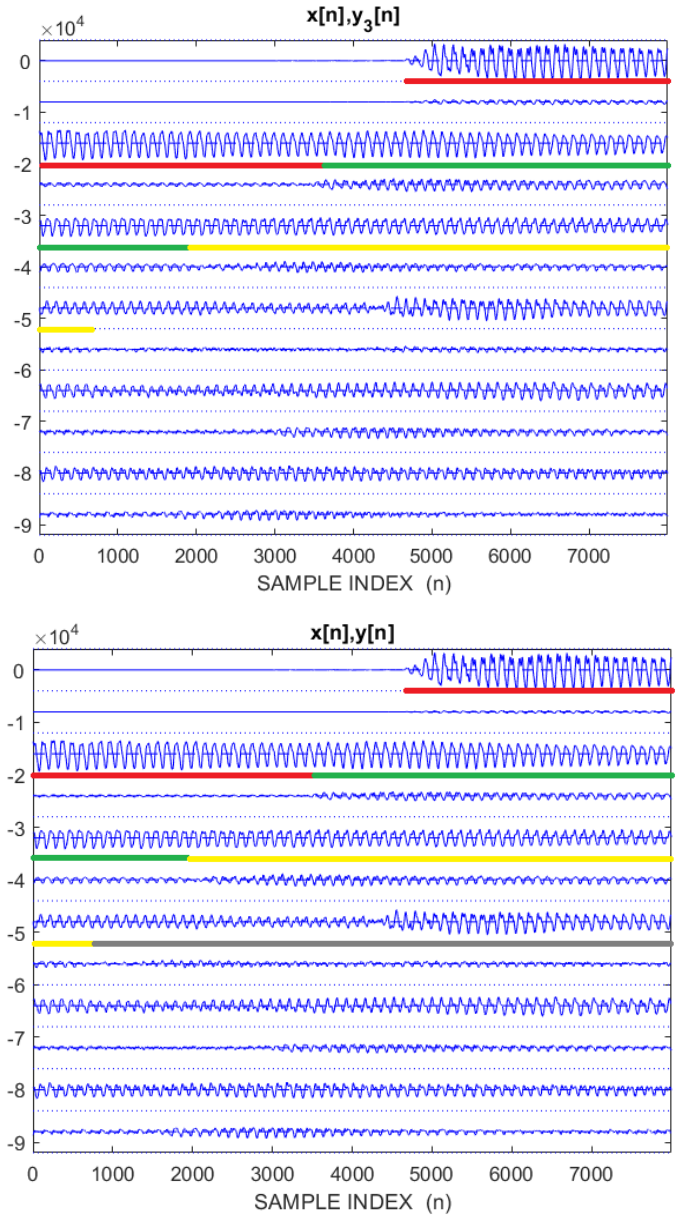


Fig.9.Plots showing effect of echo filters, plotted at the output of each stage in the cascade. The colour lines represent the time duration before each delayed signal component starts. The colour coding is: Red:= time before start of 1st delayed component, Green:= Time between start of 1st & 2nd delayed components, Yellow:= Time between start of 2nd & 3rd delayed components, Grey:= Time between start of 3rd & 4th delayed components.

The plots above show the initial section of the music signal. We can see the effect of the echoes in the time plots. Since each delayed component occurs with a delay of *P* with respect to the previous component, we can see this visually by searching for regions where the shape of the output signal changes due to addition of a new component. Using this method, the start times of the delayed components are marked in the plots above.