CA2 - Filtering

Friday, January 21, 2022 8:34 PM

FIR Filtering

1. Complete the FIR filtering function and turn in a printout of your code. Do not use built-in functions or vector operations. Use `for` loops where needed. Write the filtering function using a single for loop.

```
function y = myFIRfilter(b,x)
persistent z L;

if isempty(z)
  L = length(b);
  z = zeros(size(b));
end

y = 0;

for n = L:-1:2
  z(n) = z(n-1);
  y = y + b(n) * z(n);
end
z(1) = x;
y = y + b(1) * z(1);
end
```

2. Let the impulse response be $\mathbf{h} = [1, 2, 3, 4, 4, 3, 2, 1]$ and prove that the code is correct by inputting an impulse sequence $[1, 0, 0, 0, 0, \cdots, 0]$ and verifying that the output is the impulse response $h[n]$. You will need to write a script that calls your filtering code for each sample in the impulse sequence.

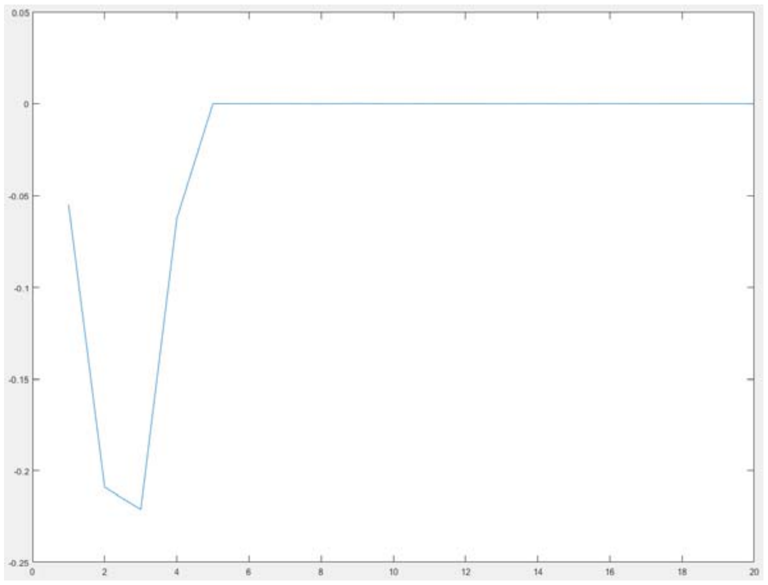
```
x = [1,0,0,0,0,0,0,0];

out = [0,0,0,0,0,0,0,0];

for n = 1:length(x)
  out(n) = myFIRfilter([1,2,3,4,4,3,2,1],x(n));
end

out
```

3. Let the impulse response be $\mathbf{h} = [0.0625, 0.2764, 0.4182, 0.2764, 0.0625]$ and the input be $x[n] = \cos(2\pi 0.422 n)$.
   a. Compute the filter output $y[n]$ for $n = 0, 1, 2, \cdots, 20$ and show that after an initial transient the steady state value of the output is zero.

```
clear myFIRfilter;
```

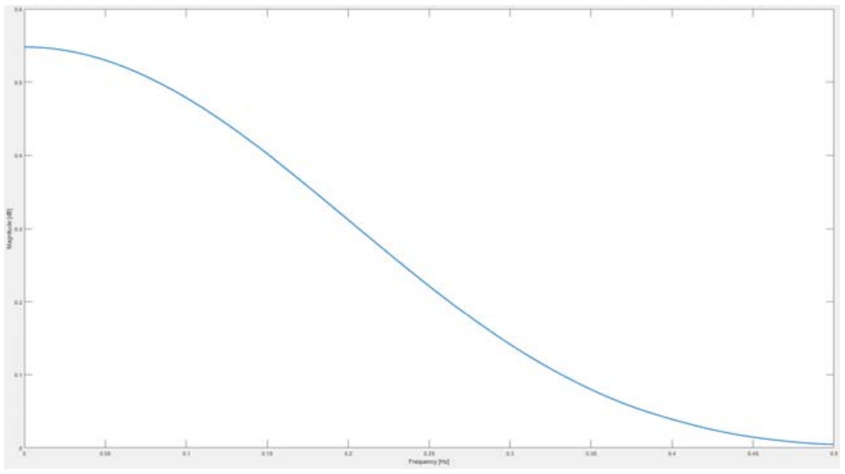
a. Compute the filter output $y[n]$ for $n = 0, 1, 2,$, 20 and show that after an initial transient the steady state value of the output is zero.

```
clear myFIRfilter;

out = [];
x = [];
for t = [1:1:20]
  x(t) =  cos(2*pi*.422*t);
end

h = [.0625,.2764,.4182,.2764,.0625];
out = zeros(size(x));


for n = 1:length(x)
  out(n) = myFIRfilter(h,x(n));
end

out
plot(out)
```



b. Using the `freqz` function, plot the magnitude response assuming the sample rate is $F_s = 1$ sample/second.

```
Fs = 1; % samples/sec
[H,F] = freqz(out,1,2^14,Fs);
plot(F,abs(H),'LineWidth',2);
xlabel('Frequency [Hz]');
ylabel('Magnitude [dB]');
```

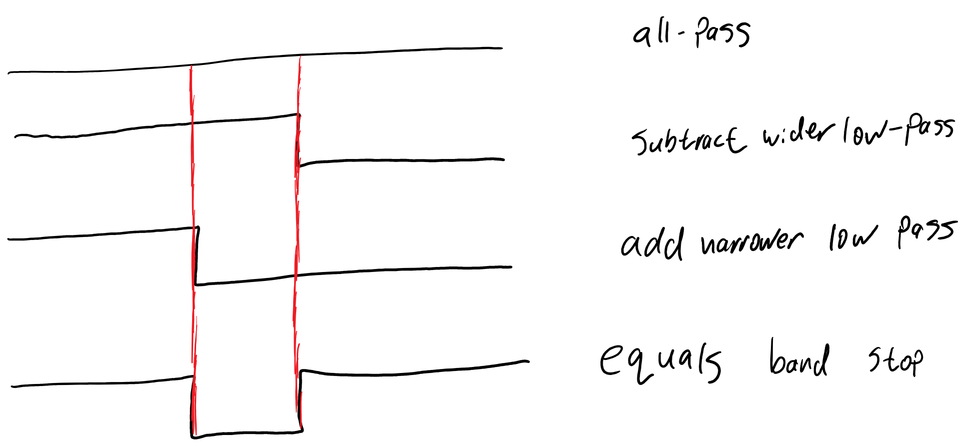


c. Using the magnitude response, explain why the steady state output is zero.

The magnitude response shows gain in the low frequencies and a 0 or nearly 0 output in the higher frequencies, meaning the result of the filter will settle to 0 after a time.

4. Design a band-stop filter by combining all-pass and low-pass filters.

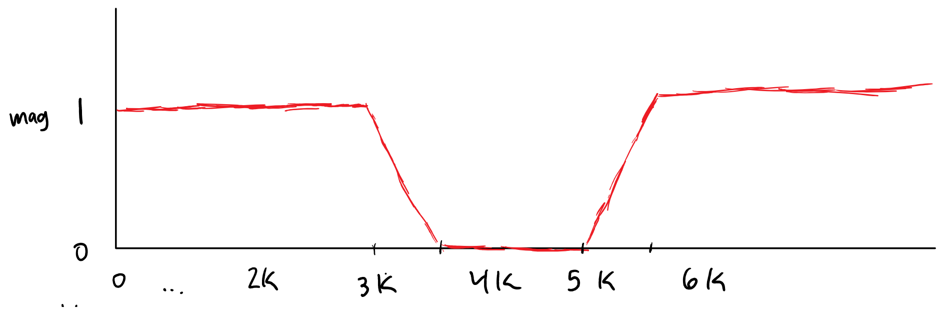
a. Show how to compute the impulse response.



$$\delta[n] - \left[(2\cdot 5K)\frac{\sin(2\pi(5K)n)}{2\pi(5K)n} \cdot \frac{\sin(2\pi(5.5K)n)}{2\pi(5.5K)n}\right]$$

$$+\left[(2\cdot 3K)\frac{\sin(2\pi(3K)n)}{2\pi(3K)n} \quad \frac{\sin(2\pi(3.5K)n)}{2\pi(3.5K)n}\right]$$

b. Make a plot of the filter magnitude response.



H_2

IIR Filtering

4. Complete the IIR filtering function and turn in a printout of your code. Do not use built-in functions or vector operations. Use `for` loops where needed. Write the filtering function using a single for loop for the feed forward section and a single for loop for the feedback section.

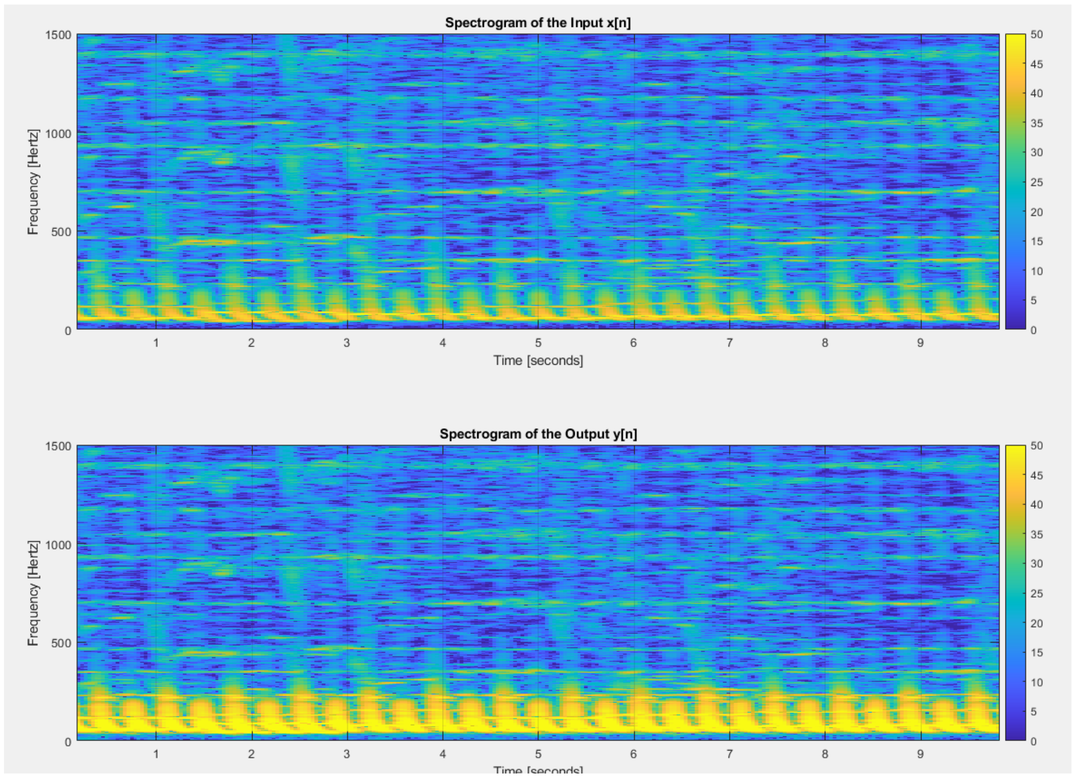
```
function y = myIIRfilter(b,a,x)
persistent zx zy Lx Ly;

if isempty(zx)
  Lx = length(b);
  Ly = length(a);
  zx = zeros(size(b));
  zy = zeros(size(a));
end



y = 0;

for n = Lx:-1:2
  zx(n) = zx(n-1);
  y = y + b(n) * zx(n);
end
zx(1) = x;
y = y + b(1) * zx(1);


for n = Ly:-1:2
  zy(n) = zy(n-1);
  y = y - a(n) * zy(n);
end
zy(1) = y;

end
```

5. Use the filter coefficients for the shelving filter and process the signal on the shelving filter web page. Use the original audio as the input signal rather than the whole song.

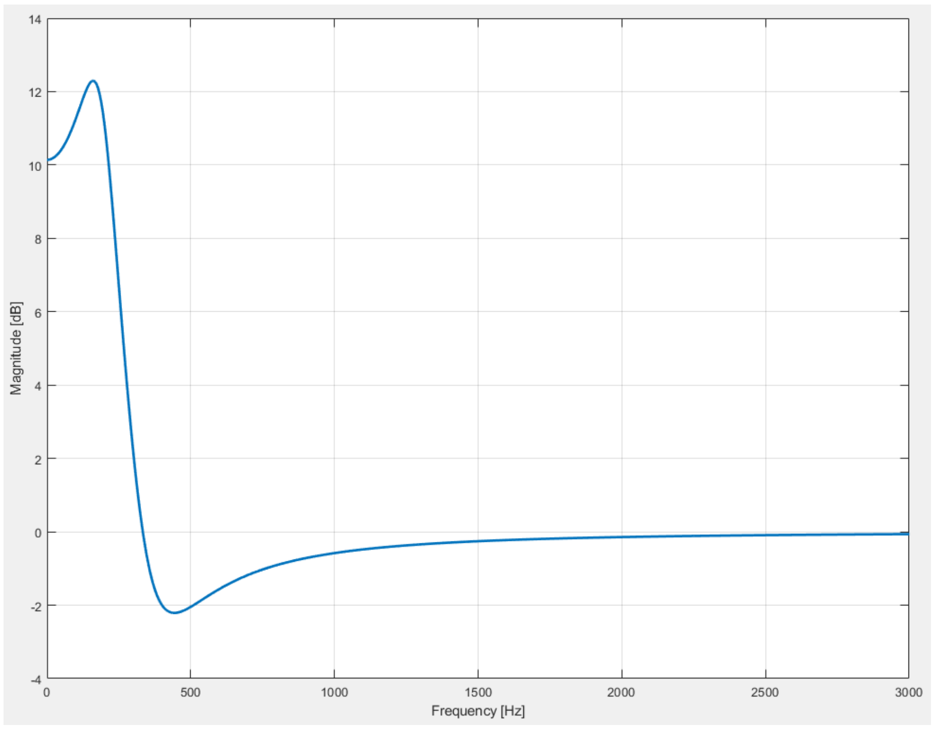
a. Make spectrogram plots of the input and output to illustrate that the shelving filter performs as expected.

```
clear myIIRfilter;

b = [1.0082,-1.9784,0.97277];
a = [1,-1.9793,.9801];
```

```
[x,Fs] = audioread('rainbow_road_snip_original.wav');

out = zeros(size(x));

for n = 1:length(x)
  out(n) = myIIRfilter(b,a,x(n));
end

audiowrite('rainbow_road_output.wav',out,Fs);
```



b. Using the `freqz` function, plot the filter magnitude response using the sample rate in the audio file.

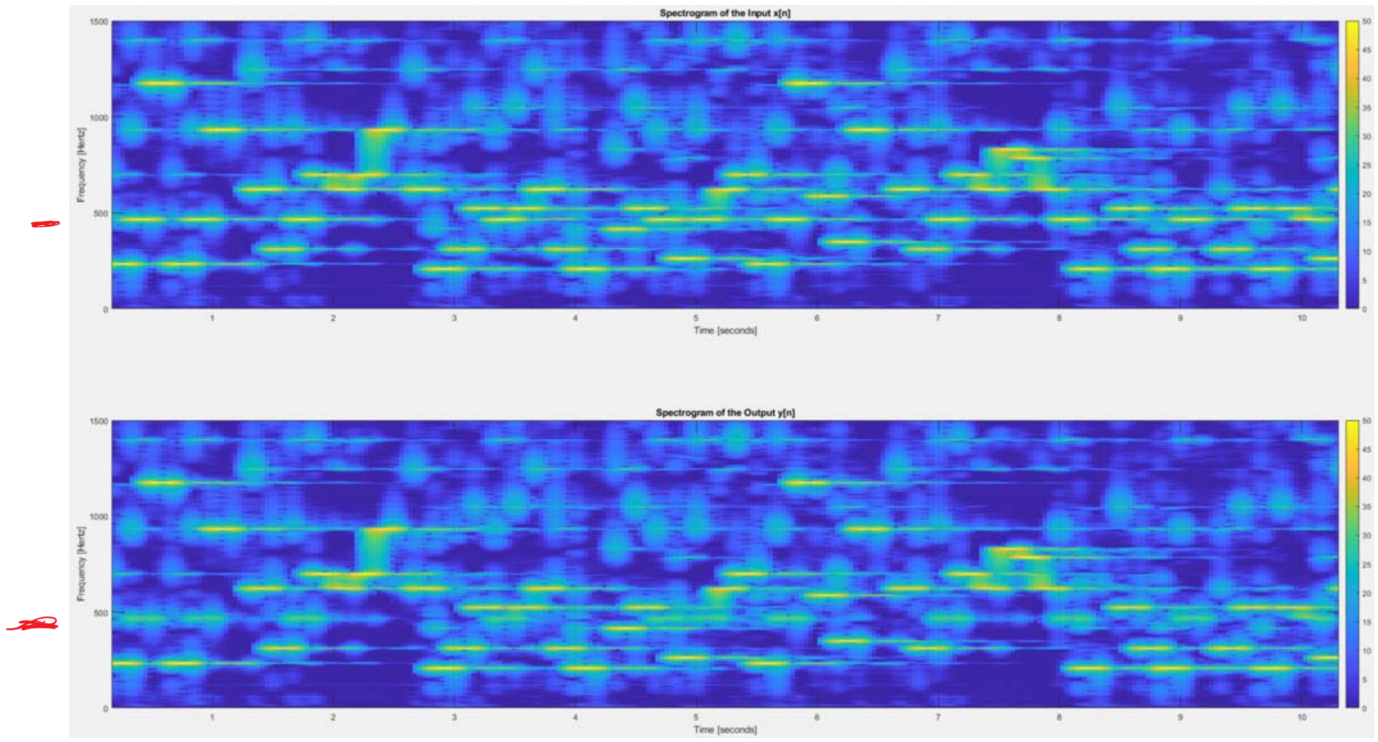


```
[H,F] = freqz(b,a,2^12,Fs);
plot(F,20*log10(abs(H)),'LineWidth',2);
xlim([0,3000]);
grid on;
xlabel('Frequency [Hz]');
ylabel('Magnitude [dB]');
shg;
```

c. What audible effect does the shelving filter impart to the signal? How does the magnitude response of the shelving filter predict this result?

It amplifies the lower frequencies without changing the higher frequencies. It shows a positive magnitude at low frequencies and a steady state of 0 gain.

6. Design a notch filter to cut out the note Bb4=A#4 which has a frequency of 466.16 Hz (https://pages.mtu.edu/~suits/notefreqs.html). This note occurs multiple times in the introduction to the song "Fireflies" by Owl City. Here is an audio clip. To design the notch filter, you will need the audio sample rate and that can be obtained from the audio file using the Matlab command shown below. Process the audio clip using your IIR filtering code and make spectrogram plots of the input signal $x[n]$ and the output signal $y[n]$. Can you see that the notch filter removed the notes at 466.16 Hz?



As can be seen above by the red marks, the filter did a pretty good (though not perfect) job of removing the notes at 466.16 Hz.

```
[x,Fs] = audioread('fireflyintro.wav');

q = exp(-20*pi/Fs);
Fn = 466.16;
fn = Fn/Fs;
w = 2*pi*fn;

b = [1, -2*cos(w), 1];
a = [1, -2*q*cos(w), q^2];

clear myIIRfilter;

% [H,F] = freqz(b,a,2^12,Fs);
% plot(F,20*log10(abs(H)),'LineWidth',2);
% xlim([0,3000]);
% grid on;
% xlabel('Frequency [Hz]');
% ylabel('Magnitude [dB]');
% shg;
```

```
out = zeros(size(x));

for n = 1:length(x)
    out(n) = myIIRfilter(b,a,x(n));
%    out(n) = exampleIIRfilter(b,a,x(n));
end

audiowrite('firefly_notch_output.wav',out,Fs);

NFFT = 2^14;

% Plot spectrograms of input and output signals
figure();
subplot(211);
[S,F,T] = spectrogram(x,hamming(NFFT),round(0.9
*NFFT),NFFT,Fs);
imagesc(T,F,20*log10(abs(S)),[0 50]);
colorbar;
set(gca,'YDir','normal');
ylim([0 1500]);
xlabel('Time [seconds]');
ylabel('Frequency [Hertz]');
title('Spectrogram of the Input x[n]');
grid on;

subplot(212);
[S,F,T] = spectrogram(out,hamming(NFFT),round(0.9
*NFFT),NFFT,Fs);
imagesc(T,F,20*log10(abs(S)),[0 50]);
colorbar;
set(gca,'YDir','normal');
ylim([0 1500]);
xlabel('Time [seconds]');
ylabel('Frequency [Hertz]');
title('Spectrogram of the Output y[n]');
grid on;
shg;
orient tall;
print -dpng audio_spectrograms.png
```