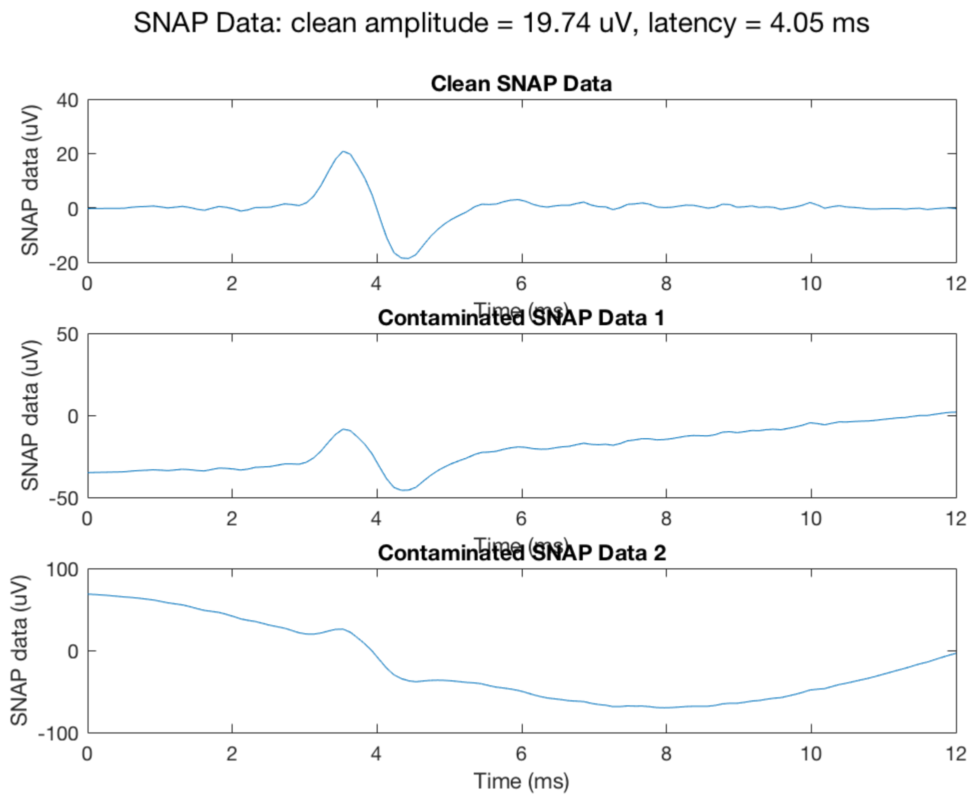
Table of Contents

Part 1

Setting the baseline by finding the info for the clean SNAP signal and plotting all the data

```
% Load data file, sampled at fs over ts
load NerveDataProject3.mat; % uV
fs = 10; % kHz
ts = 12; % ms
t = linspace(0,ts,fs*ts); % time vector for data

% Creating dummy variables to make typing easier
s1 = cleanSnap_uV; % uV
s2 = contamSnap1_uV; % uV
s3 = contamSnap2_uV; % uV

% Plot the dummy variables
figure(1);
subplot(3,1,1);
plot(t,s1);
title('Clean SNAP Data')
ylabel('SNAP data (uV)')
xlabel('Time (ms)')
subplot(3,1,2);
plot(t,s2);
title('Contaminated SNAP Data 1')
ylabel('SNAP data (uV)')
xlabel('Time (ms)')
subplot(3,1,3);
plot(t,s3);
title('Contaminated SNAP Data 2')
ylabel('SNAP data (uV)')
xlabel('Time (ms)')

% Calling the function
[ampl_Clean,latency_Clean] = analyzeSNAP(s1,fs);

% Display latency and amplitude in title
supTitle = sprintf('SNAP Data: clean amplitude = %.2f uV, latency =
 %.2f ms', ampl_Clean, latency_Clean);
suptitle(supTitle)
```

```
% Print the shell function
type('analyzeSNAP.m')


function [ampl,latency] = analyzeSNAP(x,Fs)
% function [ampl,latency] = analyzeSNAP(x,Fs)
% This is a function that takes in the SNAP (Sensory nerve action
 pulse)
% data (uV) and the sample rate (kHz) for the data and returns the
 latency
% (ms) and the amplitude (uV) for the set

[MAX,iMax] = max(x);
[MIN,iMin] = min(x);
ampl = (MAX-MIN)/2;
latency = (iMax+iMin)/(2*Fs);

return
```

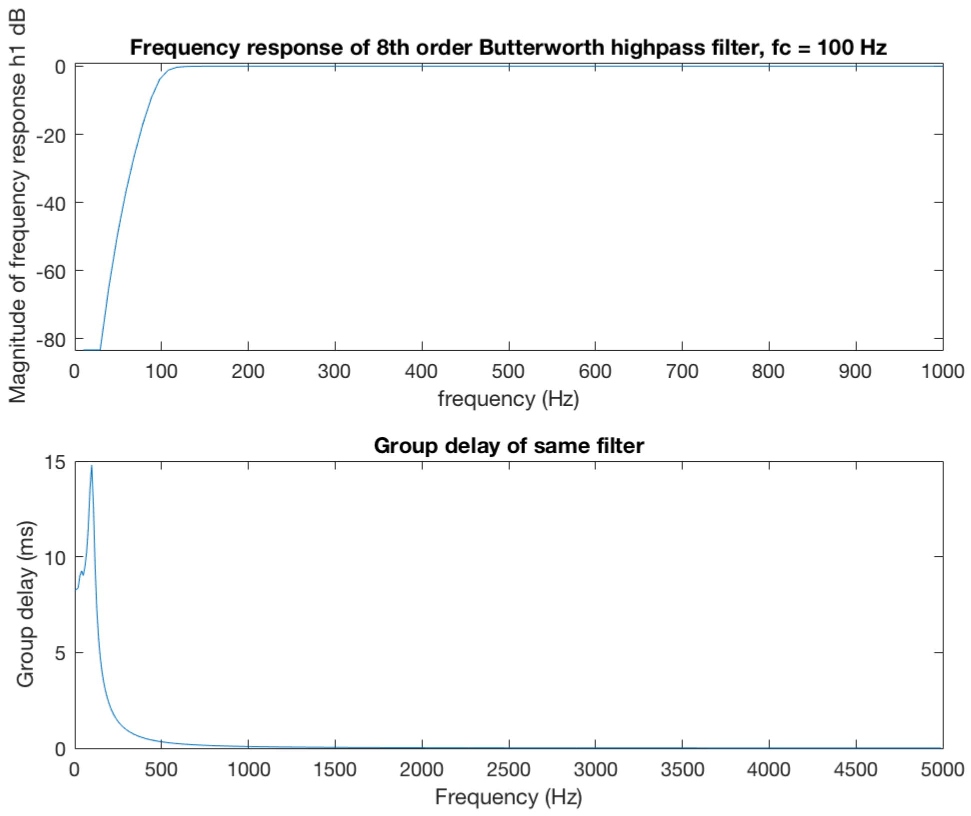


Part 2

Applying an 8th order butterworth (IIR) filter to the signal and analyze the filters frequency response

```
% Create the butterworth filter
[b1,a1] = butter(8,100/(10000/2),'high');
```

```
[gd1,w1] = grpdelay(b1,a1);
n = length(w1);
[gd1ms,w1] = grpdelay(b1,a1,n,fs*1000);

[h1,w1] = freqz(b1,a1,n);
h1dB = mag2db(abs(h1));
f1 = fs*1000*w1/(2*pi);

figure(2)
subplot(2,1,1)
plot(f1,h1dB)
title('Frequency response of 8th order Butterworth highpass filter, fc
 = 100 Hz')
ylabel('Magnitude of frequency response h1 dB')
xlabel('frequency (Hz)')
axis([0 1000 min(h1dB)-1 max(h1dB)+1])

subplot(2,1,2)
plot(f1,gd1ms/10)
title('Group delay of same filter')
xlabel('Frequency (Hz)')
ylabel('Group delay (ms)')
```



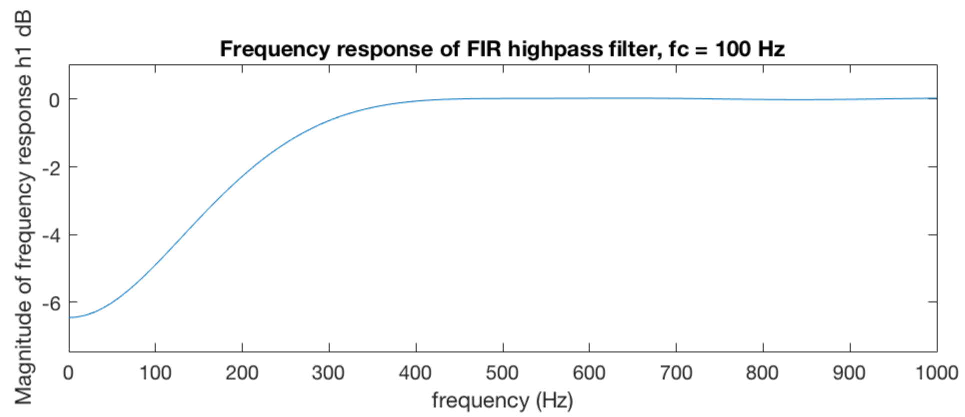
a) Taken from the graph: the filter will affect the signal by -37.2dB at ~60 Hz b) Right around the cutoff frequency at 100Hz there will be some attentuation of the signal, but it will be less than or equal to 3dB. At around 200 Hz there will still be some attenuation of the signal, but on a decibel scale it will basically be negligible. I don't know if it counts as signal distortion, but in that frequency range there will be a group delay affecting the signal which may cause frequencies in that range come later than expected.

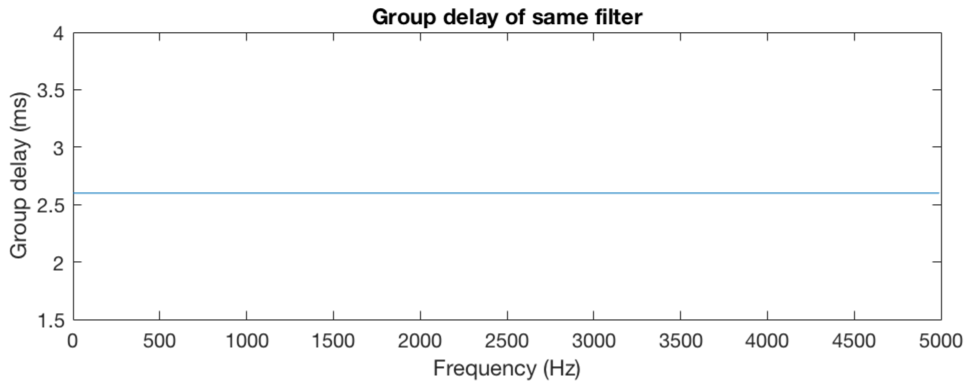
Part 3

Creating a convolutional FIR filter for cleaning up the data

```
[b2,a2] = fir1(51,100/(10000/2),'high');
[gd2,w2] = grpdelay(b2,a2);
n = length(w2);
[gd2ms,w2] = grpdelay(b2,a2,n,fs*1000);

[h2,w2] = freqz(b2,a2,n);
h2dB = mag2db(abs(h2));
f2 = fs*1000*w2/(2*pi);

figure(3)
subplot(2,1,1)
plot(f2,h2dB)
title('Frequency response of FIR highpass filter, fc = 100 Hz')
ylabel('Magnitude of frequency response h1 dB')
xlabel('frequency (Hz)')
axis([0 1000 min(h2dB)-1 max(h2dB)+1])

subplot(2,1,2)
plot(f2,gd2ms/10)
title('Group delay of same filter')
xlabel('Frequency (Hz)')
ylabel('Group delay (ms)')
```

```
Warning: Odd order symmetric FIR filters must have a gain of zero at
 the Nyquist
frequency. The order is being increased by one.
```





a) at 60Hz the filter will suppress the signal by 5.85dB b) yes, there will be magnitude loss in the frequency range for the SNAP signal, but the signal will not be distorted because the group delay is constant.

Part 4

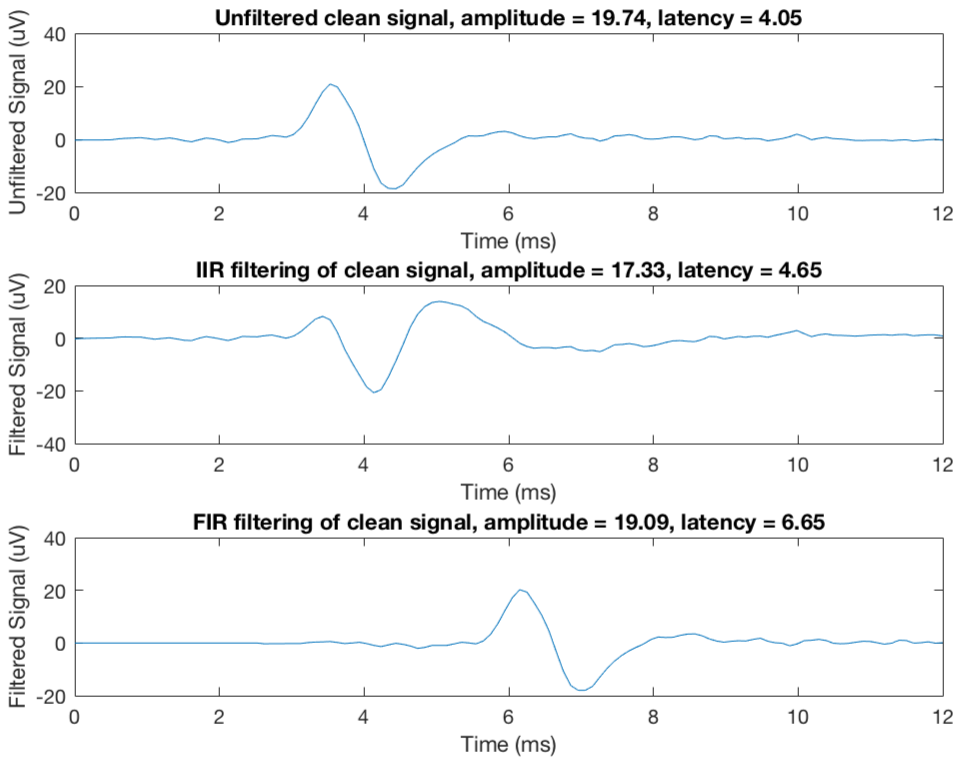
What estimates can you make about the filters in terms of computational load? The computational load for the IIR filter will probably be much lower than the load for the FIR filter because the number of coefficients for the filter are much greater, so the amount of work needing to be done in the convolution is much much greater.

Part 5

Applying each of the filters to the clean signal to see the affect on latency and amplitude.

```
% First the IIR filter
clean_IIR_Filter = filter(b1,a1,cleanSnap_uV);
[ampl_Clean_IIR,latency_Clean_IIR] = analyzeSNAP(clean_IIR_Filter,fs);
title_IIR = sprintf('IIR filtering of clean signal, amplitude = %.2f,
 latency = %.2f',ampl_Clean_IIR,latency_Clean_IIR);

% Now the FIR filter
clean_FIR_Filter = filter(b2,a2,cleanSnap_uV);
[ampl_Clean_FIR,latency_Clean_FIR] = analyzeSNAP(clean_FIR_Filter,fs);
title_FIR = sprintf('FIR filtering of clean signal, amplitude = %.2f,
 latency = %.2f',ampl_Clean_FIR,latency_Clean_FIR);
```

```
% Plot everything
figure(4)
subplot(3,1,1);
plot(t,cleanSnap_uV);
xlabel('Time (ms)');
ylabel('Unfiltered Signal (uV)')
title(sprintf('Unfiltered clean signal, amplitude = %.2f, latency =
 %.2f',ampl_Clean,latency_Clean))

subplot(3,1,2);
plot(t,clean_IIR_Filter);
xlabel('Time (ms)');
ylabel('Filtered Signal (uV)')
title(title_IIR)

subplot(3,1,3);
plot(t,clean_FIR_Filter);
xlabel('Time (ms)');
ylabel('Filtered Signal (uV)')
title(title_FIR)
```



a) In the IIR filter you can see significant distortion between the unfiltered and filtered signal, where the waveform is changed significantly, but there is not very much of a delay in the signal, where the FIR filter barely changes the signal at all (aside from some minor attenuation) but has a larger delay in the signal.

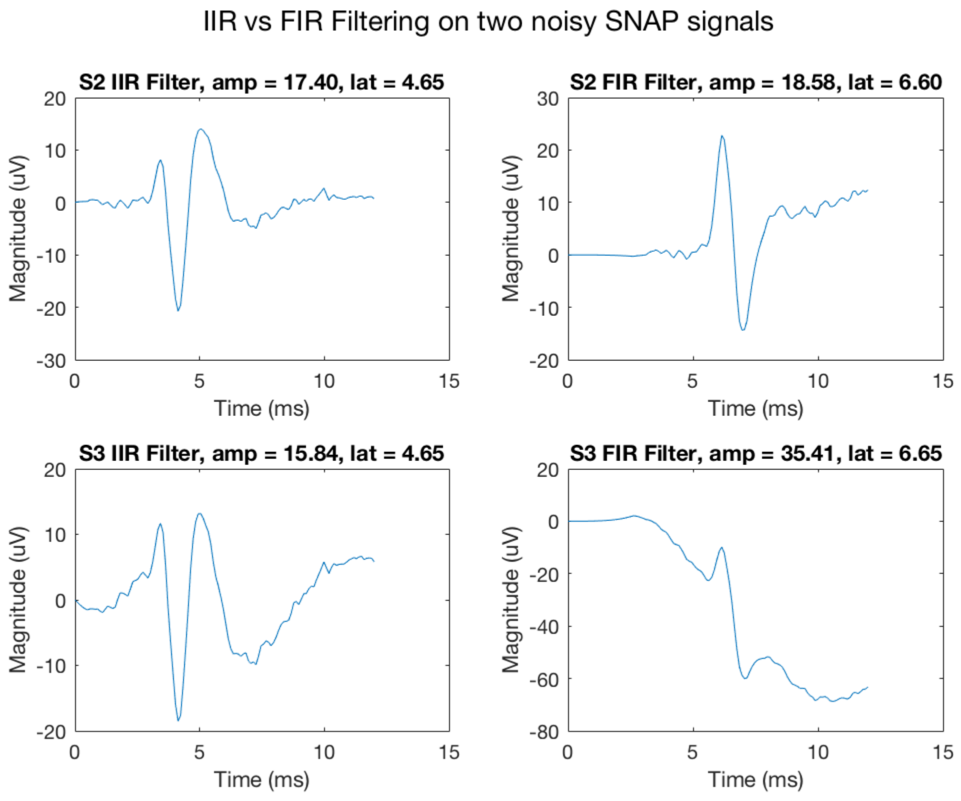
b) The latency shift in the filter is going to be 2.6ms. This is because the group delay across the frequency spectrum is uniform at 2.6ms. The group delay for an FIR filter is going to mean that the output is delayed by half of the order of the filter in samples (i.e. for a 52nd order filter it will be delayed by 26 samples, at fs = 10k, 1 sample equates to 0.1ms. For a 21 point filter the delay would be 11 samples or 1.1ms. If I were going to present the data to a doctor who doesn't care about the latency caused by the filter but only the latency in the signal, I would take the order of the filter (if odd add 1) and divide by two times the sample frequency and then subtract this number from the overall latency. The I would remove the predictable error in the data.

Part 6

In this section s1 = original snap, s2 = contaminated snap 1, s3 = contaminated snap 2

```
S2_IIR = filter(b1,a1,s2);
S2_FIR = filter(b2,a2,s2);
S3_IIR = filter(b1,a1,s3);
S3_FIR = filter(b2,a2,s3);

% Comparing the shifted and unshifted filtering
% figure(5)
% subplot(4,2,1)
% plot(t,S2_IIR)
% title('s2 IIR')
% xlabel('Time (ms)')
% ylabel('Magnitude (uV)')
%
% subplot(4,2,3)
% plot(t,S2_FIR)
% title('s2 FIR')
% xlabel('Time (ms)')
% ylabel('Magnitude (uV)')
%
% subplot(4,2,5)
% plot(t,S3_IIR)
% title('s3 IIR')
% xlabel('Time (ms)')
% ylabel('Magnitude (uV)')
%
% subplot(4,2,7)
% plot(t,S3_FIR)
% title('s3 FIR')
% xlabel('Time (ms)')
% ylabel('Magnitude (uV)')

s22 = s2 - s2(1);
s32 = s3 - s3(1);
S22_IIR = filter(b1,a1,s22);
S22_FIR = filter(b2,a2,s22);
S32_IIR = filter(b1,a1,s32);
S32_FIR = filter(b2,a2,s32);

% Plots to compare the
% subplot(4,2,2)
% plot(t,S22_IIR)
% title('s2 shifted IIR')
% xlabel('Time (ms)')
% ylabel('Magnitude (uV)')
%
% subplot(4,2,4)
% plot(t,S22_FIR)
% title('s2 shifted FIR')
% xlabel('Time (ms)')
% ylabel('Magnitude (uV)')
%
% subplot(4,2,6)
% plot(t,S32_IIR)
% title('s3 shifted IIR')
% xlabel('Time (ms)')
% ylabel('Magnitude (uV)')
%
% subplot(4,2,8)
% plot(t,S32_FIR)
% title('s3 shifted FIR')
% xlabel('Time (ms)')
% ylabel('Magnitude (uV)')
%
% suptitle('Testing to compare DC shift of signals')

[ampl_S2_IIR,lat_S2_IIR] = analyzeSNAP(S22_IIR,fs);
[ampl_S2_FIR,lat_S2_FIR] = analyzeSNAP(S22_FIR,fs);
[ampl_S3_IIR,lat_S3_IIR] = analyzeSNAP(S32_IIR,fs);
[ampl_S3_FIR,lat_S3_FIR] = analyzeSNAP(S32_FIR,fs);

figure(5)
subplot(2,2,1)
plot(t,S22_IIR)
title(sprintf('S2 IIR Filter, amp = %.2f, lat =
 %.2f',ampl_S2_IIR,lat_S2_IIR))
xlabel('Time (ms)')
ylabel('Magnitude (uV)')

subplot(2,2,2)
plot(t,S22_FIR)
title(sprintf('S2 FIR Filter, amp = %.2f, lat =
 %.2f',ampl_S2_FIR,lat_S2_FIR))
xlabel('Time (ms)')
ylabel('Magnitude (uV)')

subplot(2,2,3)
plot(t,S32_IIR)
title(sprintf('S3 IIR Filter, amp = %.2f, lat =
 %.2f',ampl_S3_IIR,lat_S3_IIR))
xlabel('Time (ms)')
ylabel('Magnitude (uV)')

subplot(2,2,4)
plot(t,S32_FIR)
```

```
title(sprintf('S3 FIR Filter, amp = %.2f, lat =
 %.2f',ampl_S3_FIR,lat_S3_FIR))
xlabel('Time (ms)')
ylabel('Magnitude (uV)')

suptitle('IIR vs FIR Filtering on two noisy SNAP signals')
```



a) Removing the initial offset gets rid of the DC bias in the signal, cleaning the signal a little bit further so that when the signal is convolved with the filter more accurate values are used and the error does not propogate through the signal.

b) In the second contaminated signal there is some sort of exponentially decreasing DC noise. the IIR filter handles this much better, yeilding a similar result to the other contaminated signal (this is because the cutoff for the highpass IIR filter is significantly sharper than that of the FIR filter so it removes the DC components better, but the group phase delay makes it so that the signal is not as clean as that of the FIR filter. So basically, the IIR filter handles low frequency noise better but introduces a delay in the frequency messing with the quality of the signal, where the IIR offers a constant phase delay but does not do as good of a job removing the low frequency noise because the filter is a smoother cutoff.

Part 7

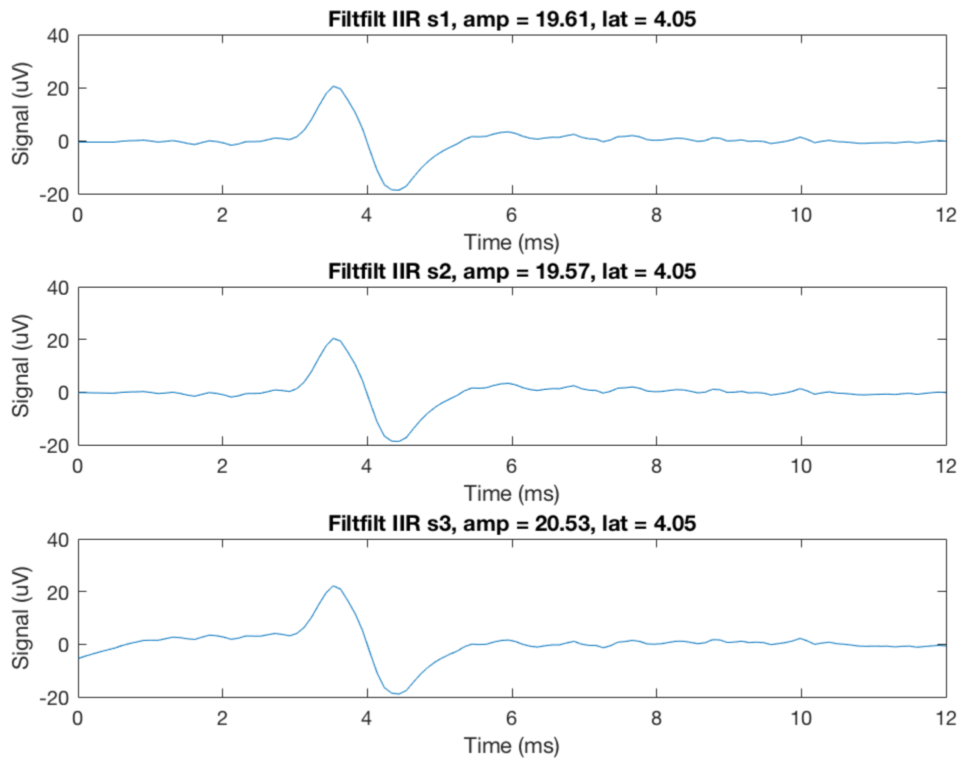
Can you see any start up transients or edge effects? Yes, for the FIR filter toward the end of the signal there was a significant trend up and a slight delay to begin taking affect. For the IIR there was an initial slight downward trend in the data but at the end there was a large growth in the signal.

For the FIR filter calculate the relationship between the start up region and the filter length. The FIR filter is going to have a start up region that is one half of the length of the filter coefficient vector plus one because of the way the convolution works, in terms of samples. Divide this by the sampling frequency and you get the overlap in terms of milliseconds and is the same as the group phase delay, 2.6ms.

Part 8

Use filtfilt with the IIR filter on all the signals

```
S13 = filtfilt(b1,a1,s1);
S23 = filtfilt(b1,a1,s2);
S33 = filtfilt(b1,a1,s3);

[amp1,lat1] = analyzeSNAP(S13,fs);
[amp2,lat2] = analyzeSNAP(S23,fs);
[amp3,lat3] = analyzeSNAP(S33,fs);

figure(6)
subplot(3,1,1)
plot(t,S13)
title(sprintf('Filtfilt IIR s1, amp = %.2f, lat = %.2f',amp1,lat1))
xlabel('Time (ms)')
ylabel('Signal (uV)')

subplot(3,1,2)
plot(t,S23)
title(sprintf('Filtfilt IIR s2, amp = %.2f, lat = %.2f',amp2,lat2))
xlabel('Time (ms)')
ylabel('Signal (uV)')

subplot(3,1,3)
plot(t,S33)
title(sprintf('Filtfilt IIR s3, amp = %.2f, lat = %.2f',amp3,lat3))
xlabel('Time (ms)')
ylabel('Signal (uV)')
```



I had trouble actually being able to implement the filtfilt command because the dimensions of the signal matrix said they were not compatible the with dimensions required to use the command (signal is 120 samples, requires 156 length vector, and I tried using a zero buffer but it failed (horribly), but from my understanding of what the filter does, using the filtfilt command runs the filter across the signal twice, attenuating a signal by the amplitude of the filter squared (or doubled in dB).

Published with MATLAB® R2016b