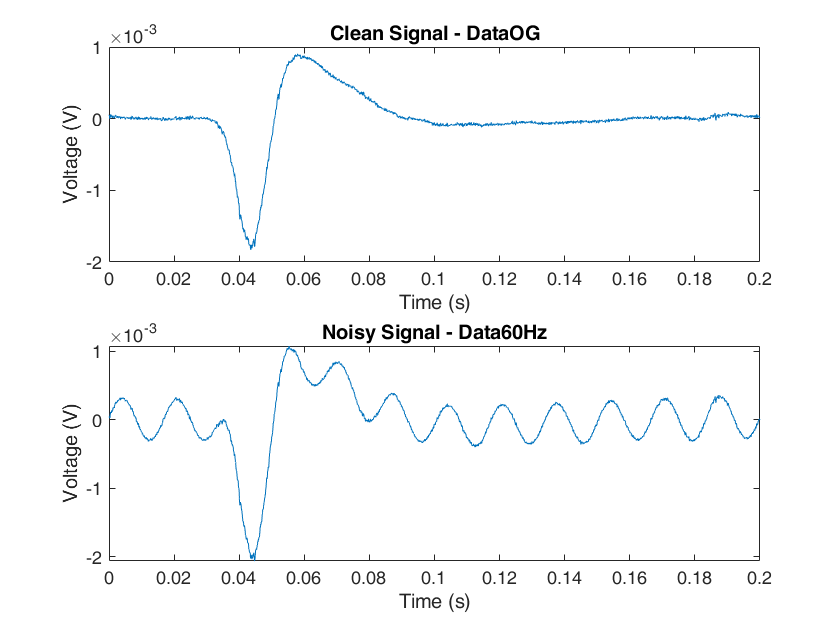
**BMEN 8101:** HW4

Rithika Varma Dandu  
Dept. of Biomedical Engineering

**PART – I**

**1.**

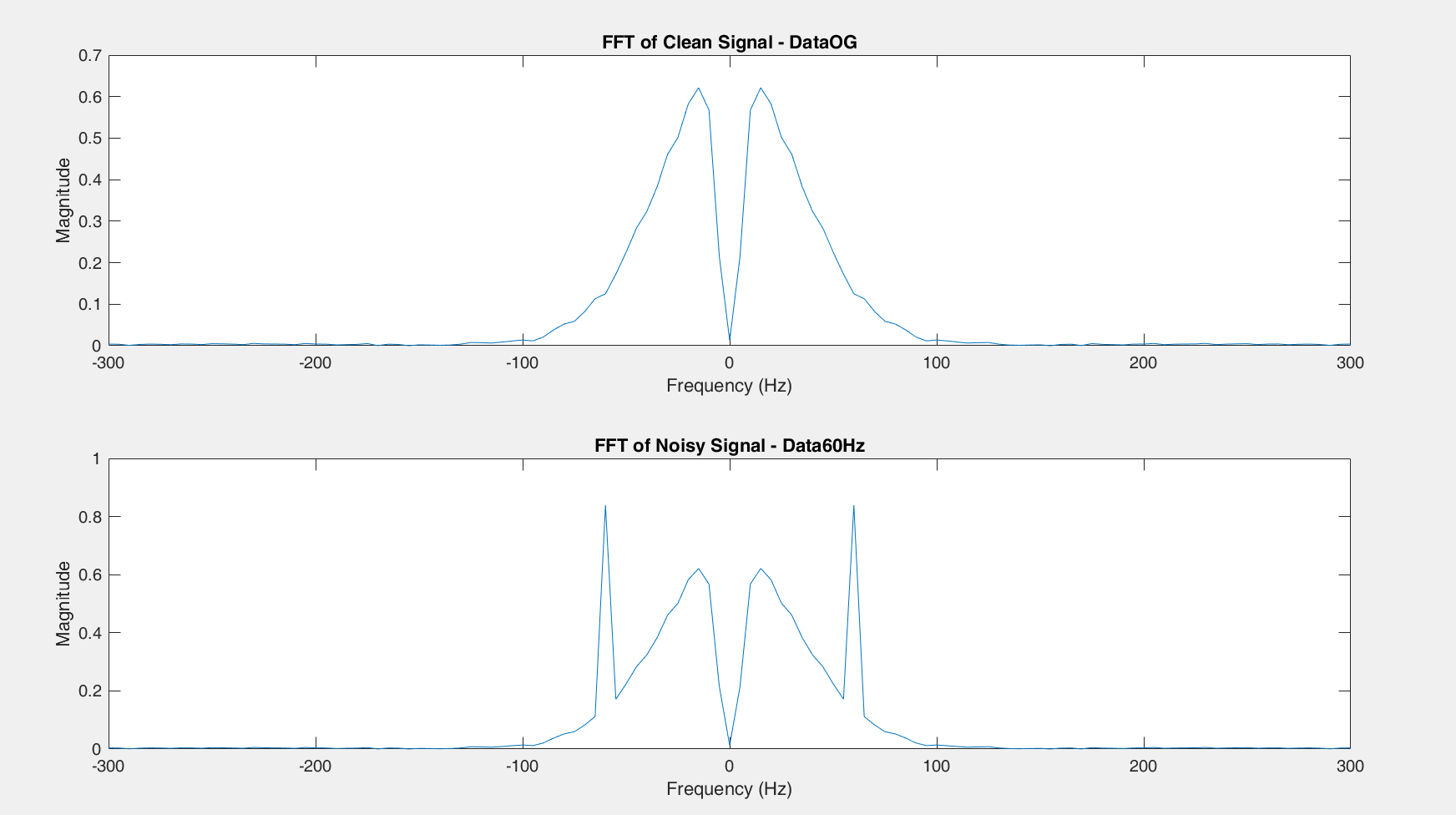
****

**Fig 1**: Voltage vs Time plots of clean and noisy signals

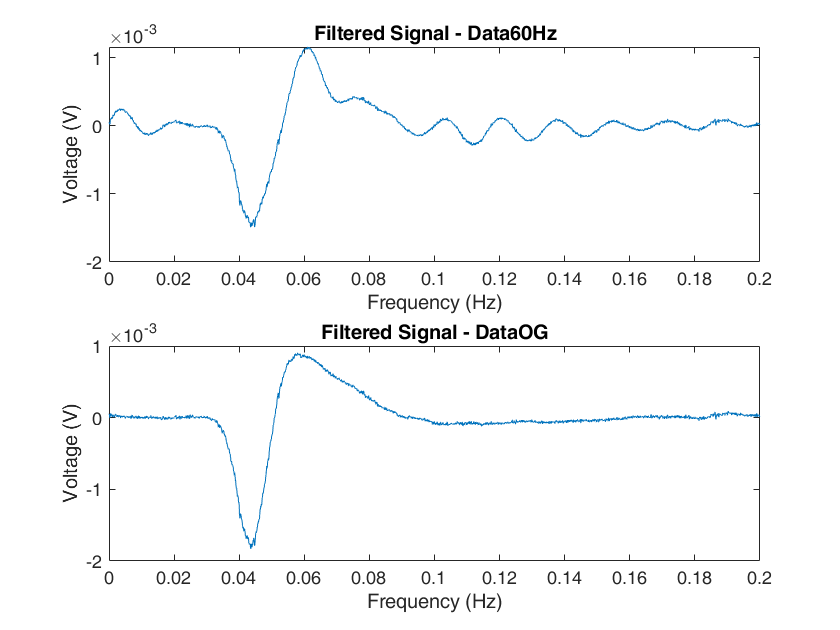
The last sample is 200msec or 0.2 seconds.

**2.**

The FFT of the clean and the noisy signals are shown below in Fig 2. I was able to visually verify that the noise was 60Hz noise by zooming into this curve. I also used MATLAB code to extract the exact frequency and found it to be 59.9854 Hz.



**Fig 2**: FFT of the Clean and Noisy Signals

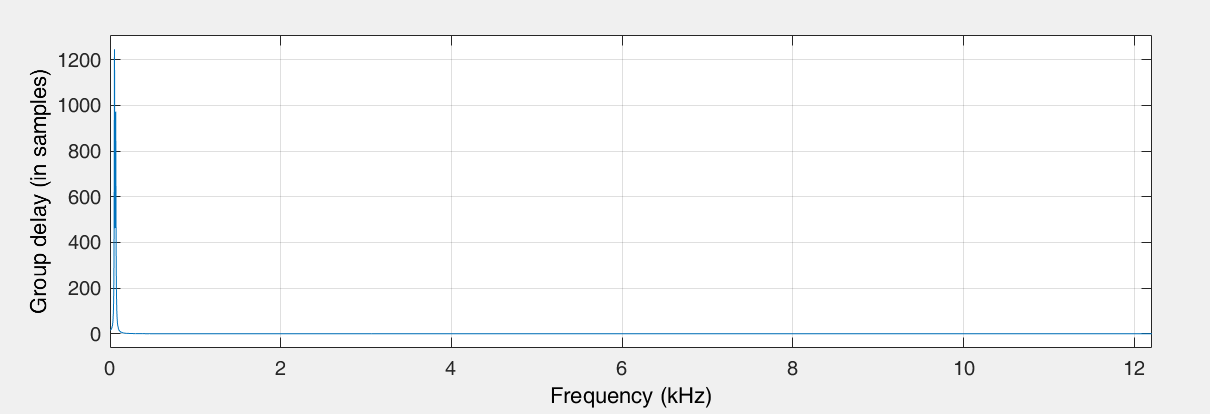


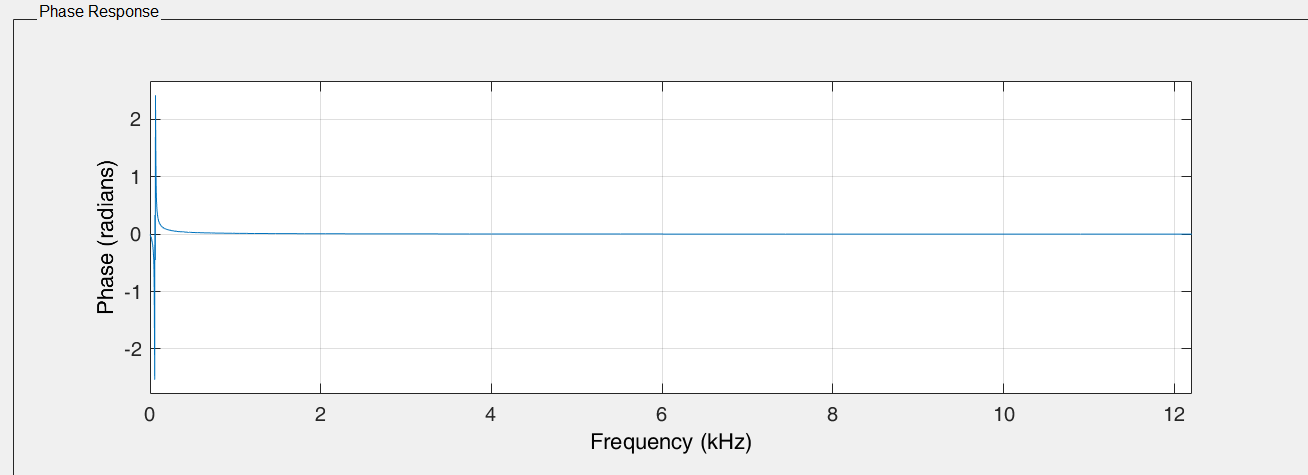
**Fig 3:** Best filter performance on Data60Hz and DataOG

My best filter was an IIR Elliptic filter of order 4 with Fpass values at 50 and 70Hz, and Fstop values at 56 and 64Hz to filter out the 60Hz noise. My Apass values were 1 and Astop value was 12 to introduce significant attenuation in the stop band. We can see from **Fig 3** that while my filter was able to flatten some ripples, it did not entirely get rid of them. Filtering as a standalone method did not prove to be sufficient.

**3.**

IIR filters typically (and unsurprisingly) had lower orders, and were more computationally efficient. For the earlier filter shown in (2), the filter order was only 4.





**Fig 4:** Group delay vs Frequency (above), Phase vs Frequency (below) for the filter depicted in Q2

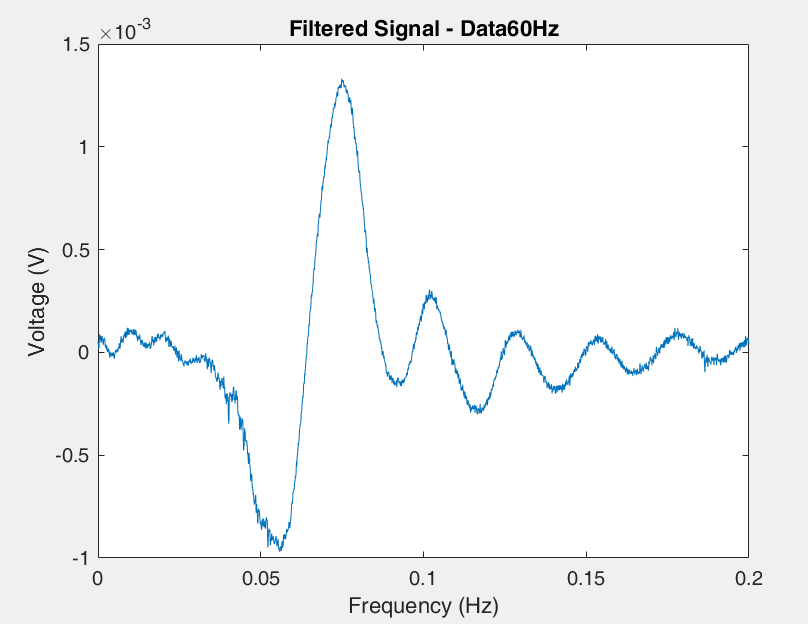
As we can see, there isn’t significant group delays or phase shifts introduced by the best filter shown in question 2 that would cause significant distortion either.

I preferred using the IIR filters for computational efficiency and ease of implementation as FIR filters often took longer to compute and yielded errors. I did this while keeping an eye on group delays.

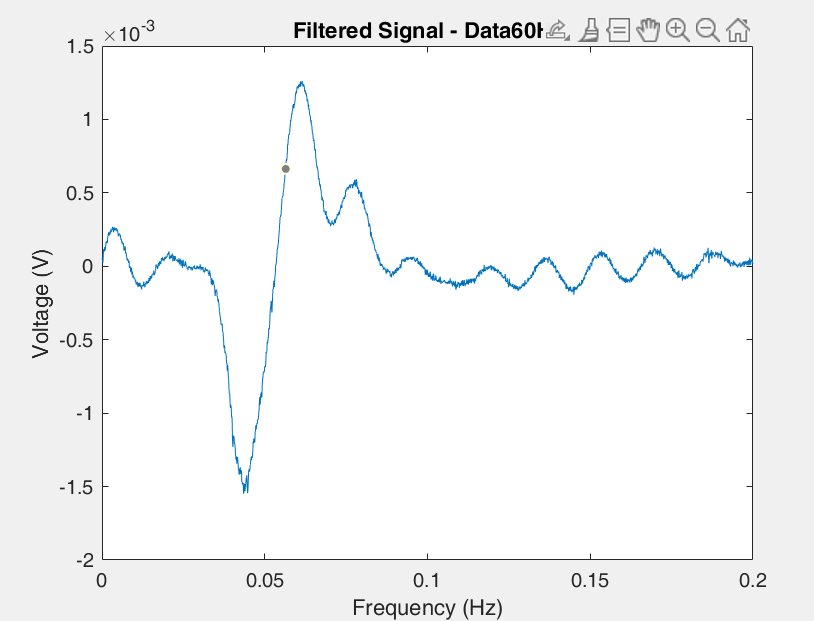
Regarless of the kind of response, I used bandstop filters in various configurations (Elliptic IIR, Butterworth IIR, Equiripple FIR etc) because I was trying to design something similar to a notch filter – with the stop band centered around the 60Hz noise, while letting other frequencies pass through.

I did try different filter widths at each iteration, by increasing or decreasing the notch width by 10-20Hz at each iteration. For example, one filter had its stop band between 58 and 62Hz, while another would have it at 50 and 70Hz. My widths did not vary much beyond this because I did not want to filter out parts of the signal, nor did I want to permit noise to remain.

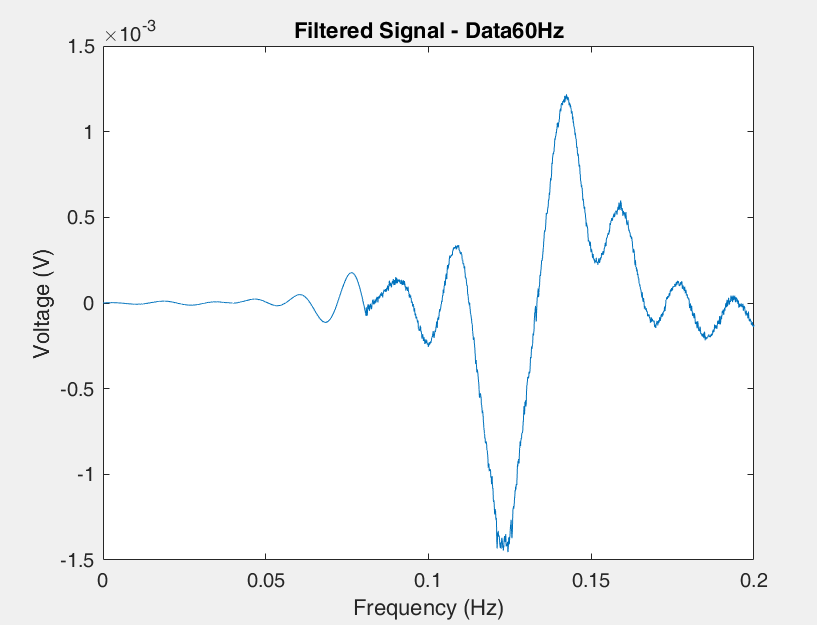
Some of the other filters I tried were:



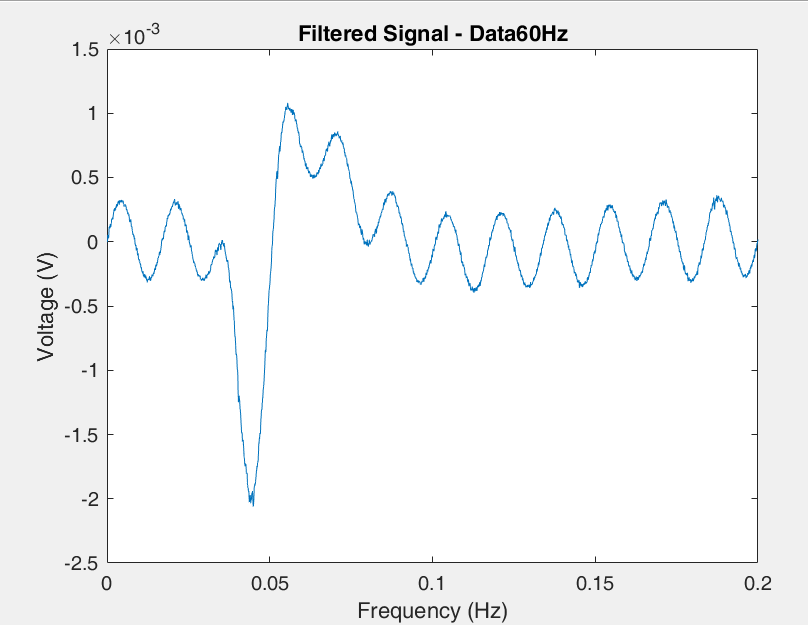
**Fig 5**: A Butterworth Bandstop IIR Filter with order 14.

****

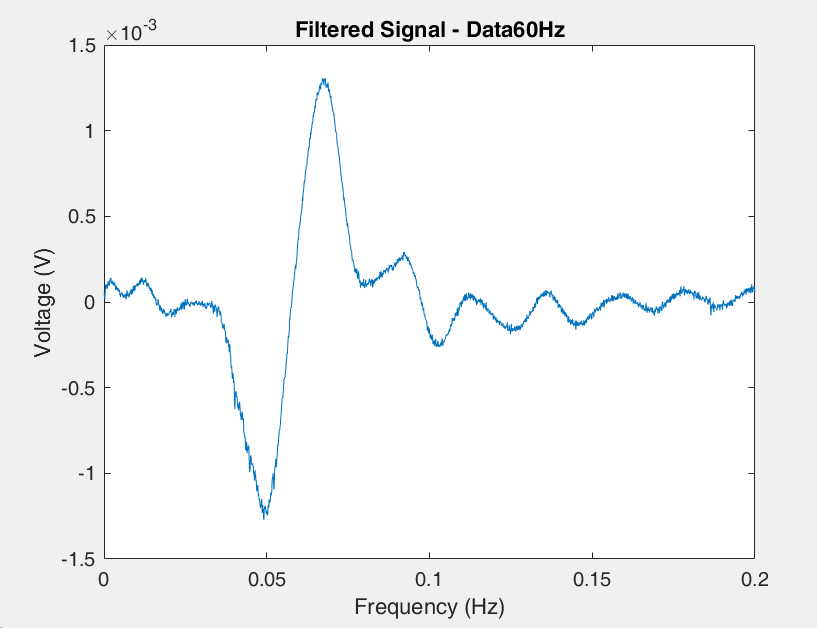
**Fig 6:** Butterworth Bandstop IIR Filter with order 6



**Fig 7:** FIR Equiripple filter with order 3950



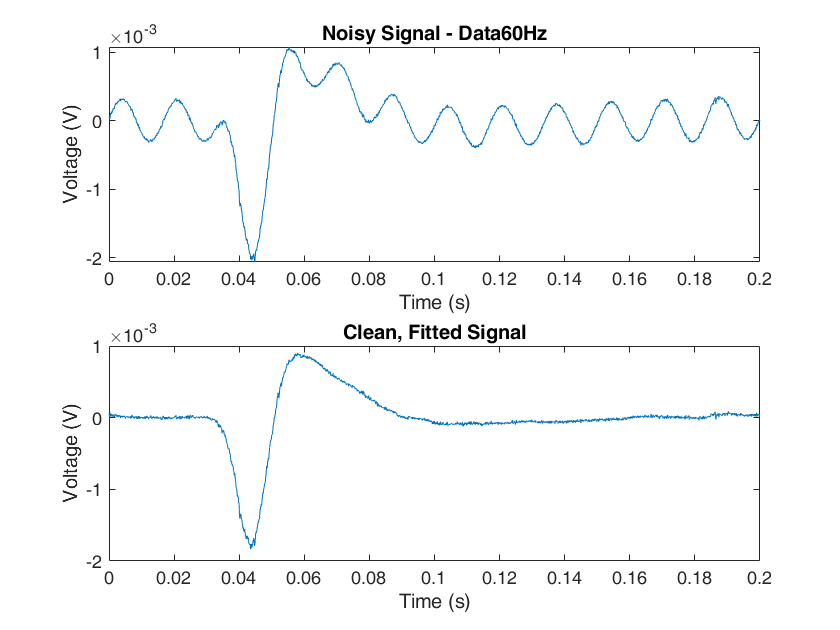
**Fig 8:** FIR Window (Hamming) filter with order 10

. 

**Fig 9**: IIR Chebyshev Type 1 Filter with order 10

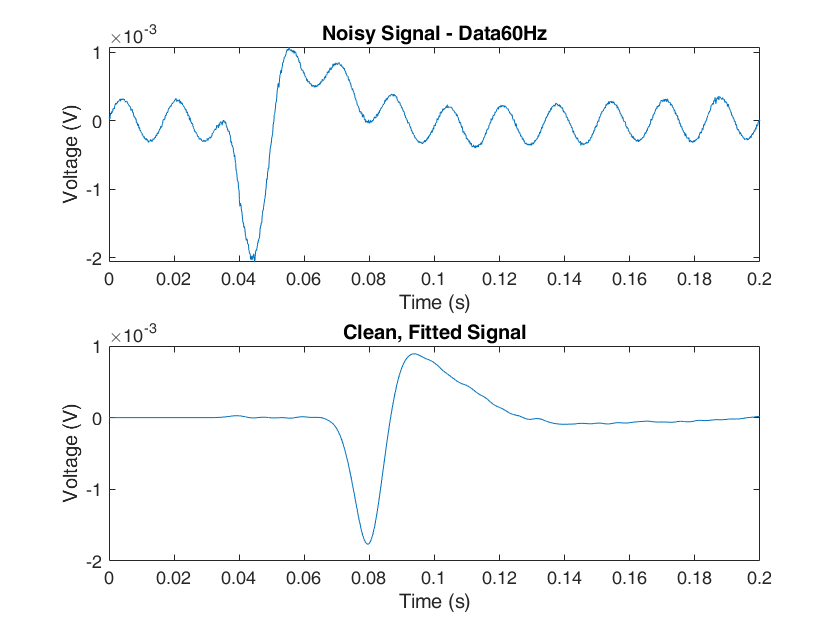
**4.**

The exact frequency of the spike was determined to be 59.9854 Hz. The function fit60 (defined in fit60.m) attempts to model 60Hz noise by creating a sinusoid at the same frequency and then subtracting it from the noise signal. I found this method to be a lot more straightforward and it yielded good results. I’m attributing the quality of results to the fact that fit60 is able to more or less directly model the noise and subtract it, while the filters I had previously designed attempt to capture an accurate range for which the noise exists. There is also no attenuation taking place here, which simplifies things a lot more.



**Fig 10**: Noisy Signal and Fitted Signal (time domain)

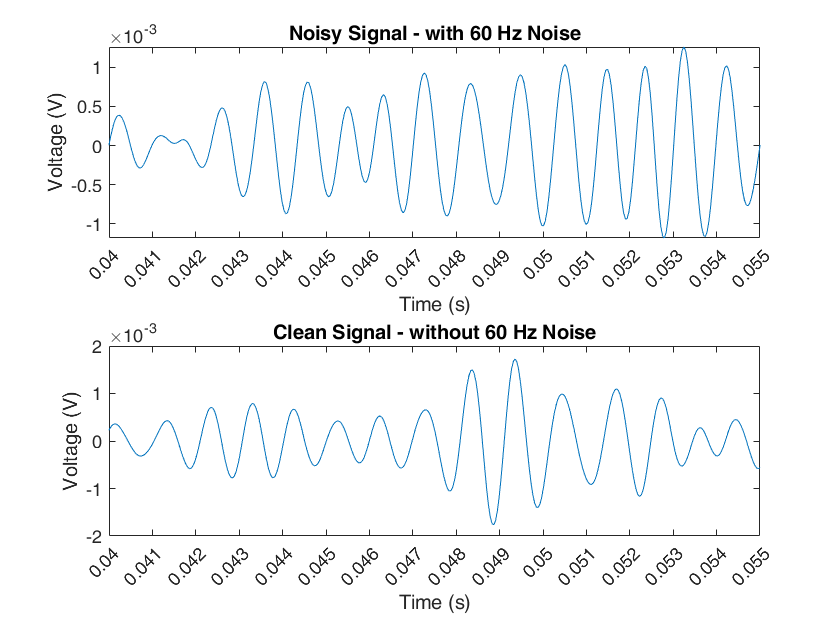
Most of the major ripples were eliminated, but the signal still had some noise. I plotted the FFT and noticed spikes outside the 200Hz range as the question indicated. I designed a low pass filter to get rid of these spikes, and passed the previously fitted 60Hz signal through it. This yielded a much cleaner result.



**Fig 10**: Noisy Signal and Clean, Fitted Signal AFTER filtering

**5.**

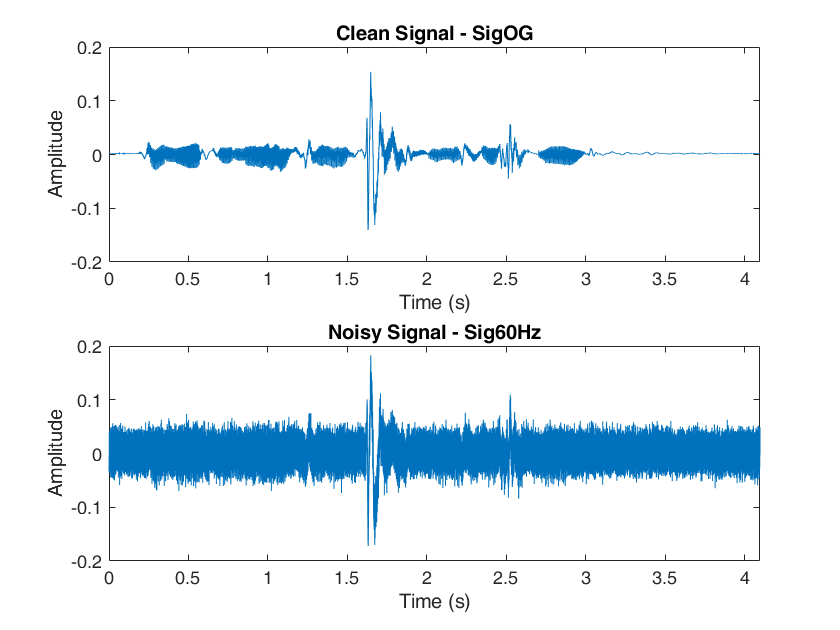
The best filter that was able to detect neural spikes of roughly 1msec each was a bandpass IIR filter. I used the frequency ranges provided in the question and had Fpass1 and Fpass2 set to 290 and 310Hz. Fstop1 and Fstop2 were 275 and 3025Hz.



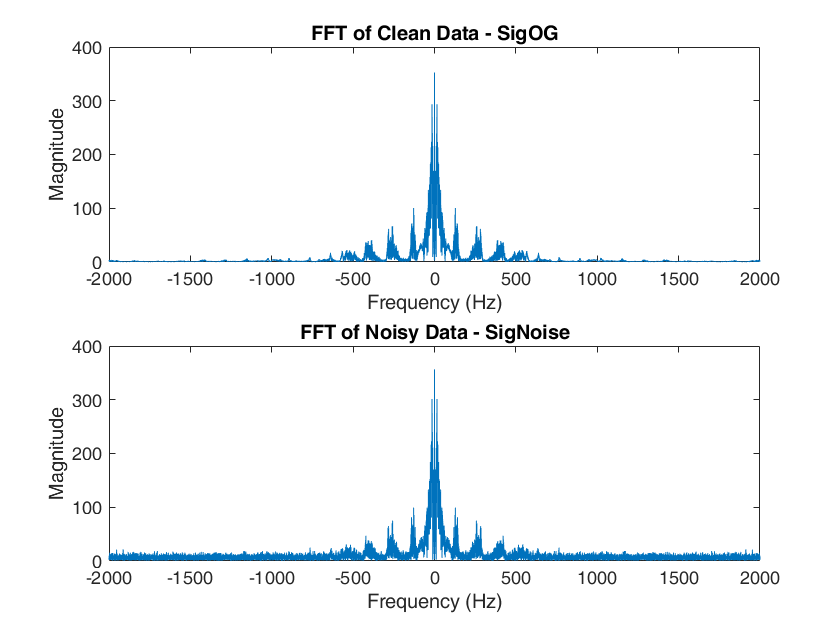
**Fig 11:** Neural spikes in 300 – 3000Hz

**PART – II**

1.



**Fig 12:** Clean and Noisy Speech signals



**Fig 12:** FFT Magnitude plots of Clean and Noisy Speech signals

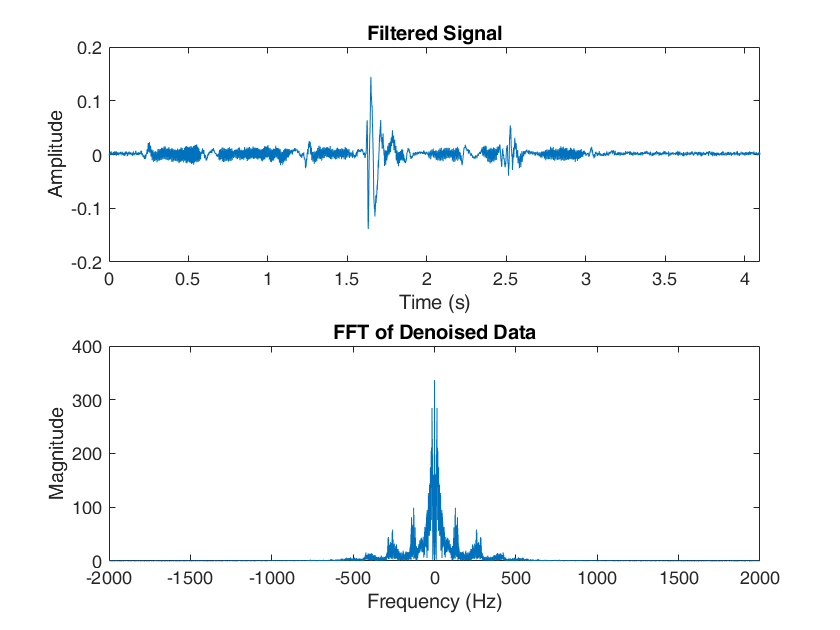
Note: these FFT plots are zoomed in to [-2000 2000] to provide better representation of the spectrum. The noise outside of this range was resembled the noise we’re seeing outside of the speech signature.

By plotting the FFT of both signals and the time domain signals, I could see that the noise is distributed across frequencies unlike in Part – 1.

At first I created filters by taking zoomed in plots of the FFT and attempting to estimate where most of the noise frequencies generally peak. Using these frequencies, I designed a few low pass filters. While designing the low pass filters, I experimented with using high Astop and low Apass values because we require minimum attenuation to the speech signal but the noise needs to be removed. These generally yielded better results.

While it was helpful in filtering some noise, a second approach I tried worked better. I started to create filters centered around human speech frequencies 20 – 200kHz.

I found that this method was extremely good at denoising, but made Dr. Lim’s voice quite muffled. I did not amplify the voice as the question states we cannot do this, but despite this – a significant portion of the noise was removed and the message is clear.



**Fig 13**: Time domain and FFT plot of the filtered signal post filtering

When plotted and compared to the previous time and FFT plots of the original signal, we can observe some overcorrection in some regions which is possibly contributing to the muffling. This is especially visible in the FFT plot – where some part of the true signal in the 750 – 1500 Hz regions are no longer present in my denoised FFT. While I was able to retrieve the sound signature, this came at the cost of some elements of the signal without which muffling occurs.

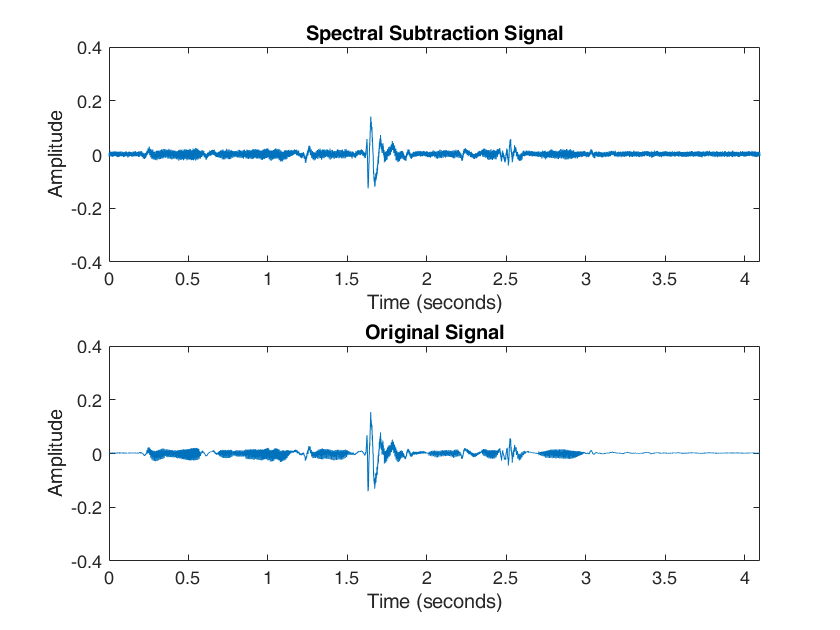
My best audio file is saved as ‘RithikaVarma\_Denoised.wav’. I also used the function audiowrite because wavwrite isn’t compatible with my version of MATLAB.

**2.**

For this part, a pipeline of two methods. I first divided the signal into intervals at which different thresholds would be applicable, and then utilized an IIR low pass filter to do some additional filtering.

I used the original time signal to estimate which regions required different thresholds. I ended up using one threshold Thres2 for all indices between 0.18s and 0.3s and threshold Thres1 for all indices outside of that range.

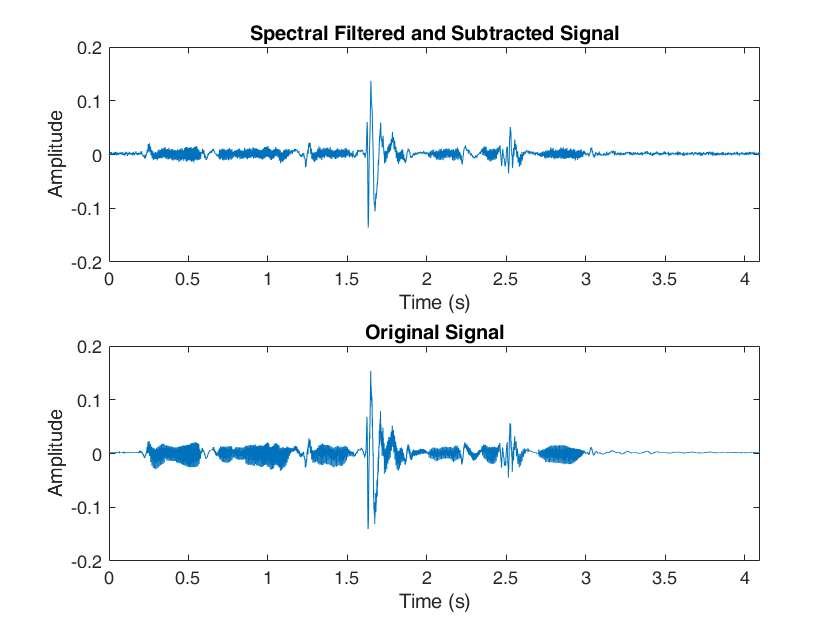
I experimented with different thresholds for the best spectral subtraction output I could manage. It was clear that Thres1 had to be significantly higher than Thres2. High Thres2 would cause overfiltering of the sound signature, but since there was relatively low information in the region of Thres1 I could subtract more noise here without worrying about losing important information. The output was respectable, but I wanted to try using a low pass filter to see if I could get rid of some of the additional noise.



**Fig 14:** Time domain speech signal post spectral subtraction compared to the original speech signal

I had to use lower Astop values (1dB) and Apass values (2dB) this time because higher values were causing the signal to start shrinking due to attentuation. Fpass was 190 Hz and Fstop was 19090Hz.

There is still some noise that can be observed by comparing it to the original signal, but the overall sound signature is retrieved and using another low pass filter helped clean up some of the additional noise that was left over from spectral subtraction alone.



**Fig 14:** Time domain speech signal post spectral subtraction and filtering compared to the original speech signal