Yeison Rodriguez

UTD ID:202139026

Signals and Systems for Biomedical Engineering

Mini Project #3

April 30, 2020

Mini Project #3 Report

For this project I began part one by loading the signal given to us for part a. I then plotted the signal vector in this file against time to graph our Noisy ECG signal in the time domain. Then I created two variables for the signal, one for sampling time (Ts) and the other for sampling frequency (Fs). I set my sampling frequency equal to the number of samples over the length of time that the signal was sampled. I found a sampling frequency of 200.0111 Hz and a sampling time of .005 seconds. Then I used my sampling frequency variable to create a frequency vector so that I could begin plotting in the frequency domain. To get my signal in the frequency domain for part c, I used the two functions fft() and fftshift() to get the fourier transform of the noisyECG signal. Then I used the abs() function to get the magnitude spectrum of our signal and plotted it against frequency.

The following are the two graphs obtained in part a and c of part one:

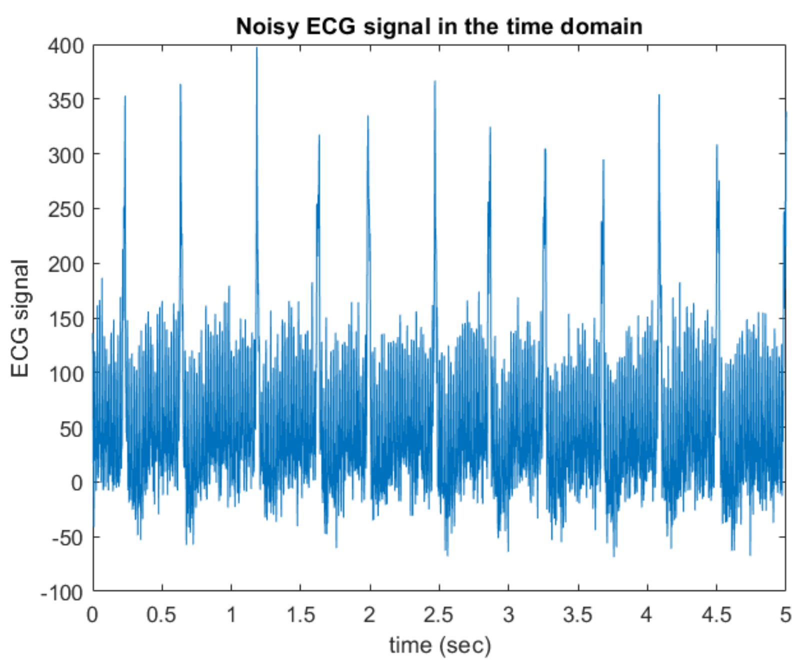

Figure 1. Noisy ECG signal in the time domain

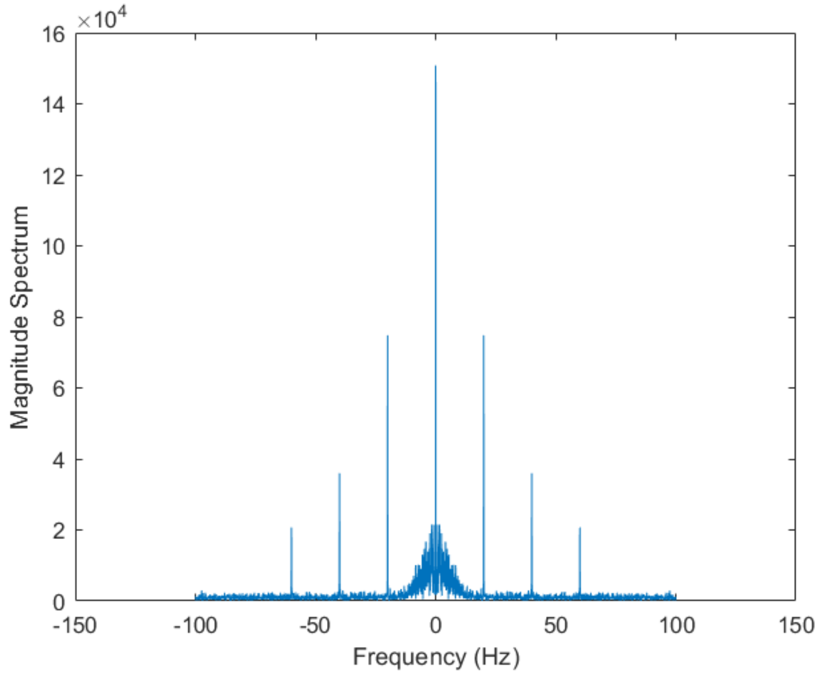

Figure 2. Magnitude Spectrum of the noisy ECG signal in the frequency domain

Based on these two plots, I decided that I would need to design a low pass filter to remove the noise. As the magnitude spectrum plot reveals that the low frequencies are the important frequencies, with most frequencies falling between 15 Hz of zero. Therefore, for part d and e, I used the filterDesigner toolbox to design a low pass filter. After some modification, I decided that a lowpass filter with Fpass = 9 Hz, Fstop = 15 Hz, and Astop = 80 dB would best accomplish the task of cleaning the signal.

The following is the filter I designed using the filterDesigner toolbox:

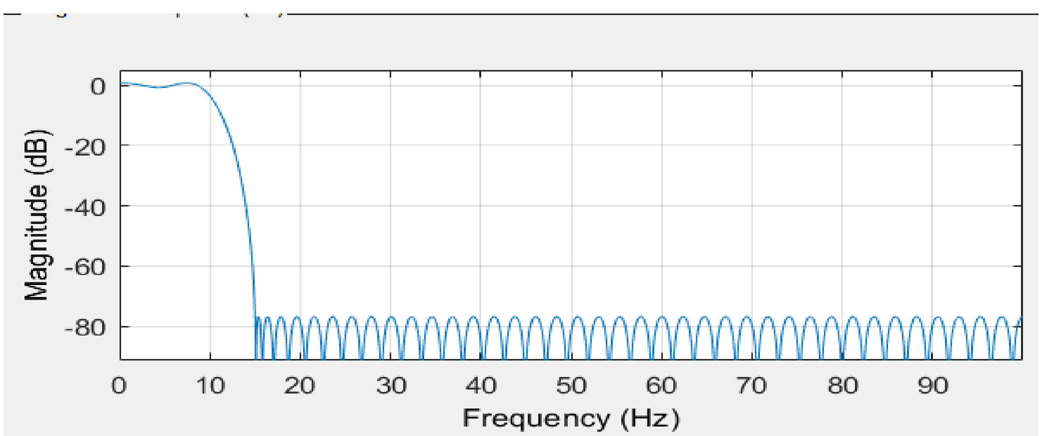

Figure 3. Filter with Fpass = 9 Hz, Fstop = 15 Hz & Astop = 80 dB. Designed using filterDesigner

Once I designed this filter, I saved it into a variable called num2. Then I used the filter to filter my signal using the filter() function. Once my signal was filtered, I plotted my now clean ECG signal into the time and frequency domain using the previously described methods.

The following is the resulting plot of the cleaned ECG signal:

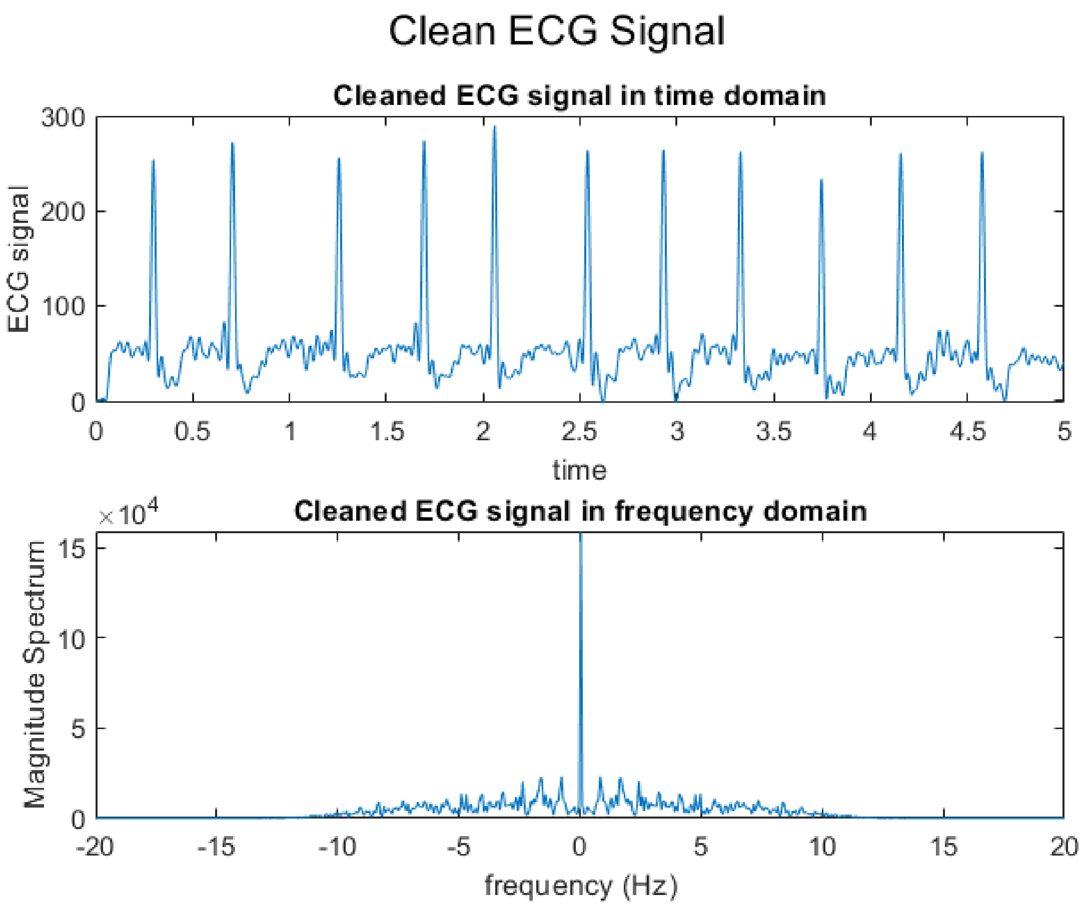

Figure 4. Cleaned ECG signal in the time and frequency domain

Although the signal was not fully cleaned, the filter still managed to remove most of the noise that was affecting our signal. In this form, we are now able to read the peaks of our signal for further analysis.

For part two of the mini project, I began by choosing an appropriate sampling frequency. I chose a sampling frequency of 400 Hz and a sampling time of .0025 seconds. This sampling frequency is substantially higher than the Nyquist frequency, 2 Hz, and so I don't believe there was any issue with aliasing in my program. Once I chose the sampling time I created a vector for time that went from -3 to 3 seconds in a step size of .0025 seconds. Then I also created a frequency vector using linspace() that went from -200Hz to 200Hz with length(t) samples.

Then I used my time vector alongside the given formulas to create the signals x(t), p(t), and h(t). I created y(t) and w(t) from multiplying the signals x(t) with p(t) and y(t) with p(t) respectively. I created the signal for v(t) by using the conv() function to convolve the signal w(t) with h(t). I then had to create new time (tv) and frequency vectors (fv) for my signal v(t) because convolution creates a new function in matlab with length = length of signal one + length of signal two -1.

Once I created my functions in the time domain, I needed to get their Fourier transforms to plot them in the frequency domain. I once more used the fft() and fftshift() function to get the Fourier transform of all my signals, and then used the abs() function to get the magnitude spectrum of every signal.

Once I had my signal in both the time domain the frequency domain, I created six separate plots for each of my signal. The following are my resulting plots:

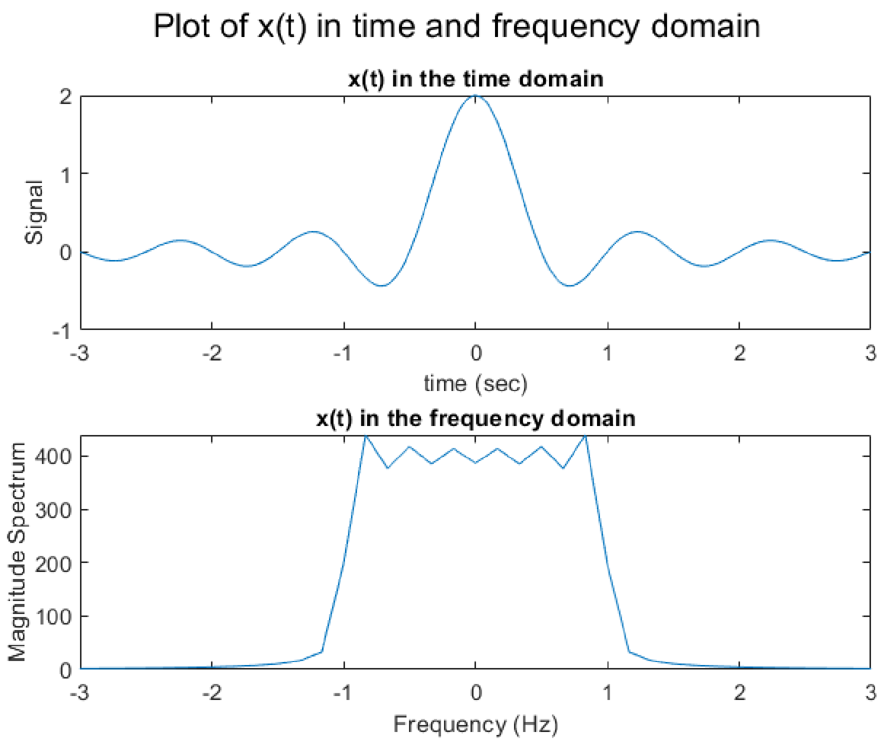

Figure 5. x(t) in the time and frequency domain

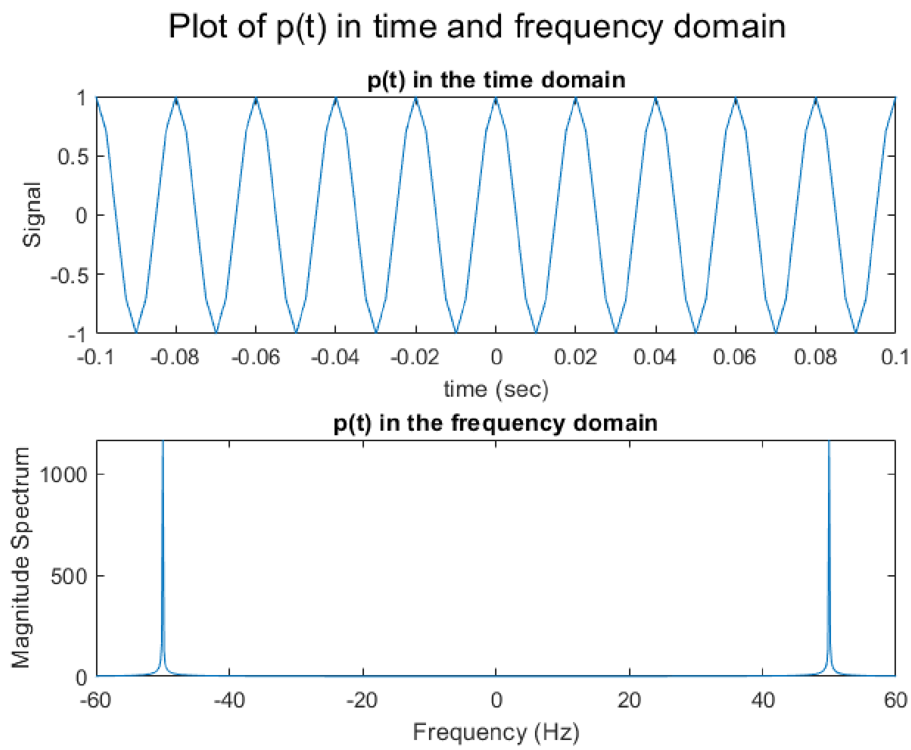

Figure 6. p(t) in the time and frequency domain

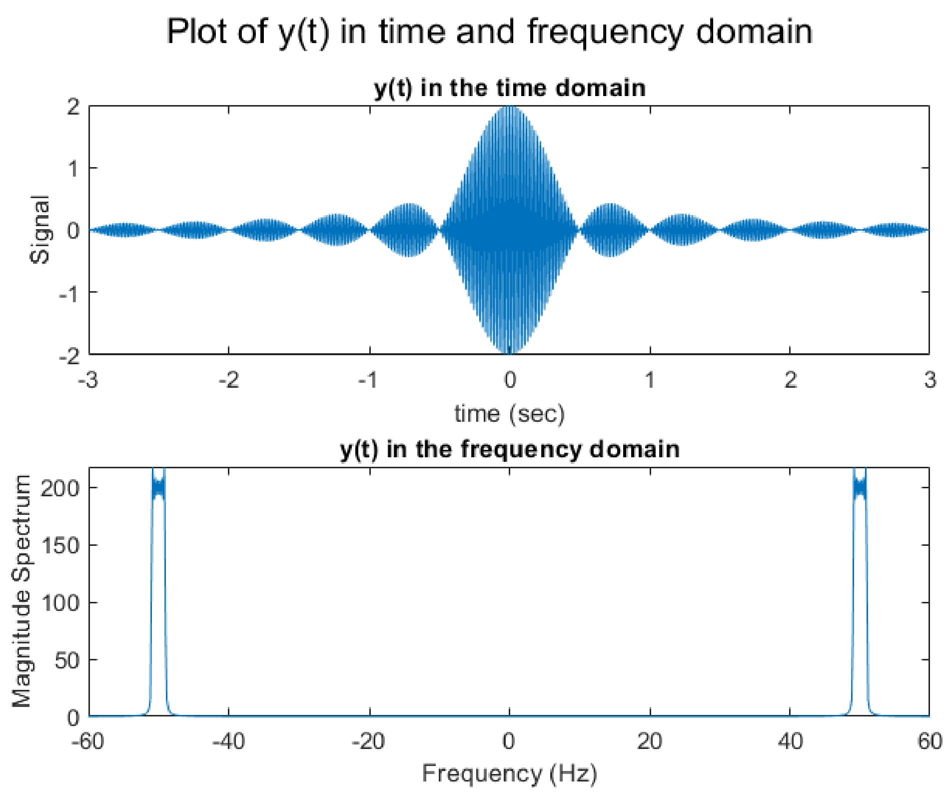

Figure 7. y(t) in the time and frequency domain

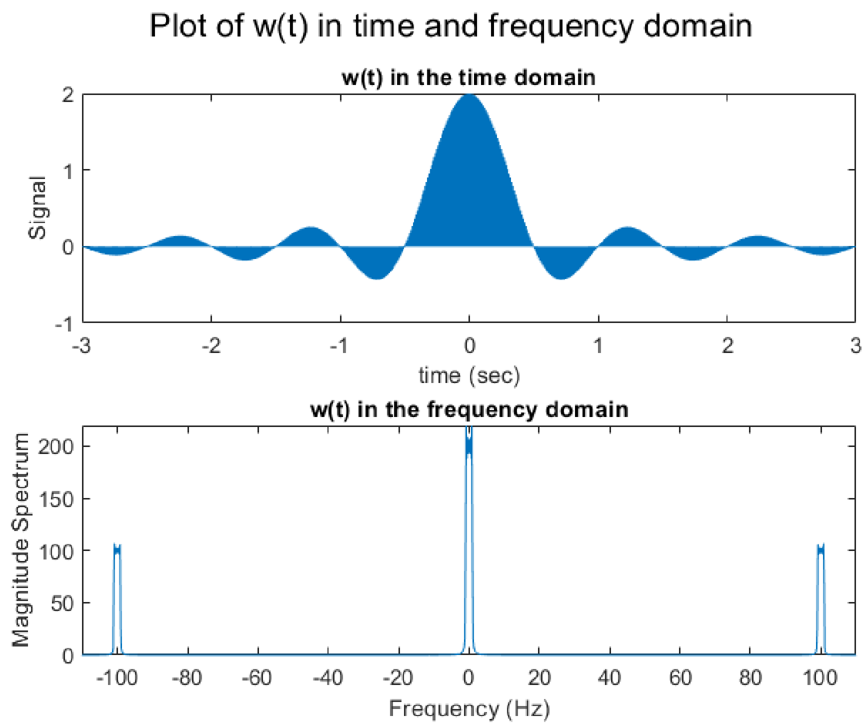

Figure 8. w(t) in the time and frequency domain

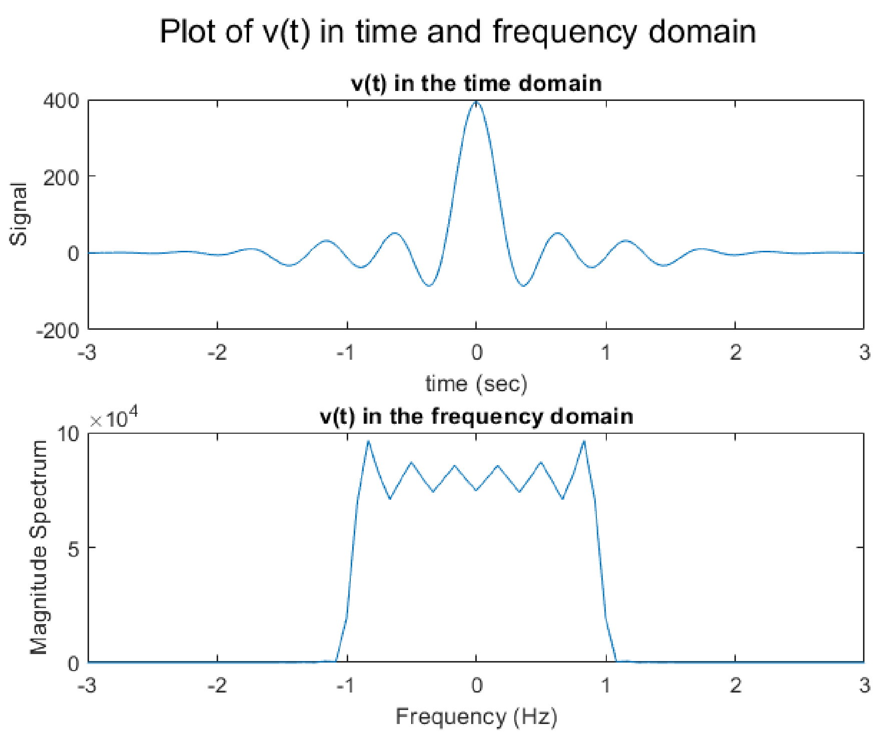

Figure 9. v(t) in the time and the frequency domain.

The resulting plots are a good approximation of what I expected. As x(t) and p(t) did create the correct version of their Fourier transforms, and y(t) did create an envelope of a sinc function around an impulse train. w(t) was also what I expected as the frequency domain showed that the rect signal at f = 0 Hz was double the amplitude of the two rect signals at -100 and 100 Hz. Finally, v(t) is also what I expected, as the low pass filter h(t) successfully removed the two rect signals at -100 and 100 Hz and left us with an amplified version of our signal x(t). Therefore, my resulting plots successfully displayed the process of amplitude modulation through a modulation/demodulation system.