extract_nomotion

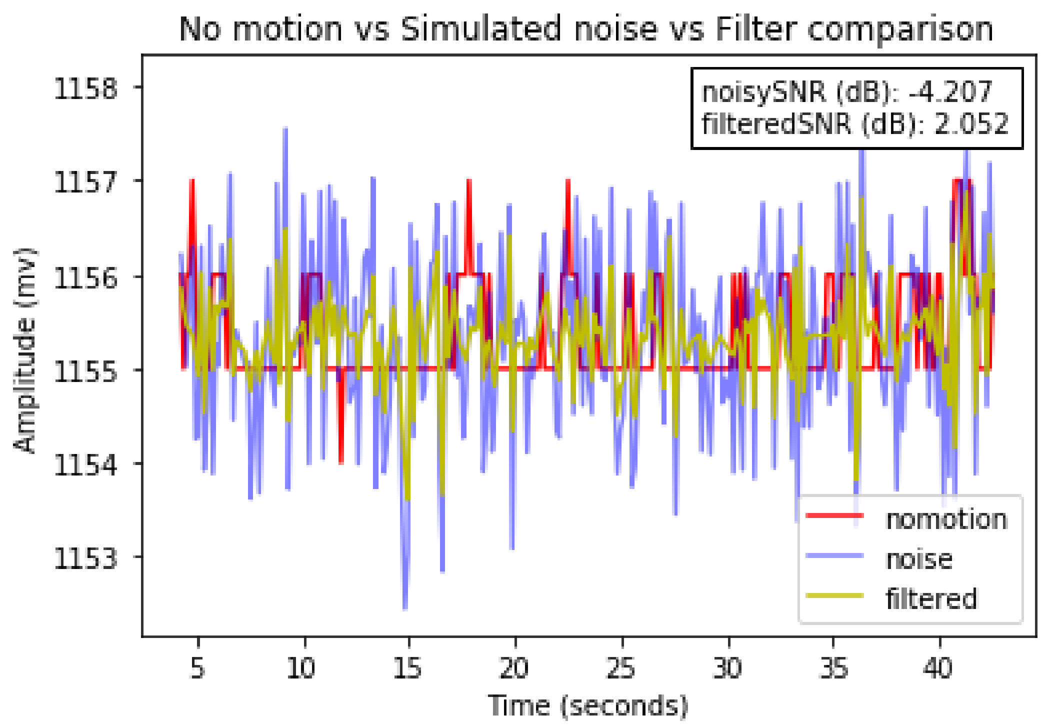
April 27, 2017

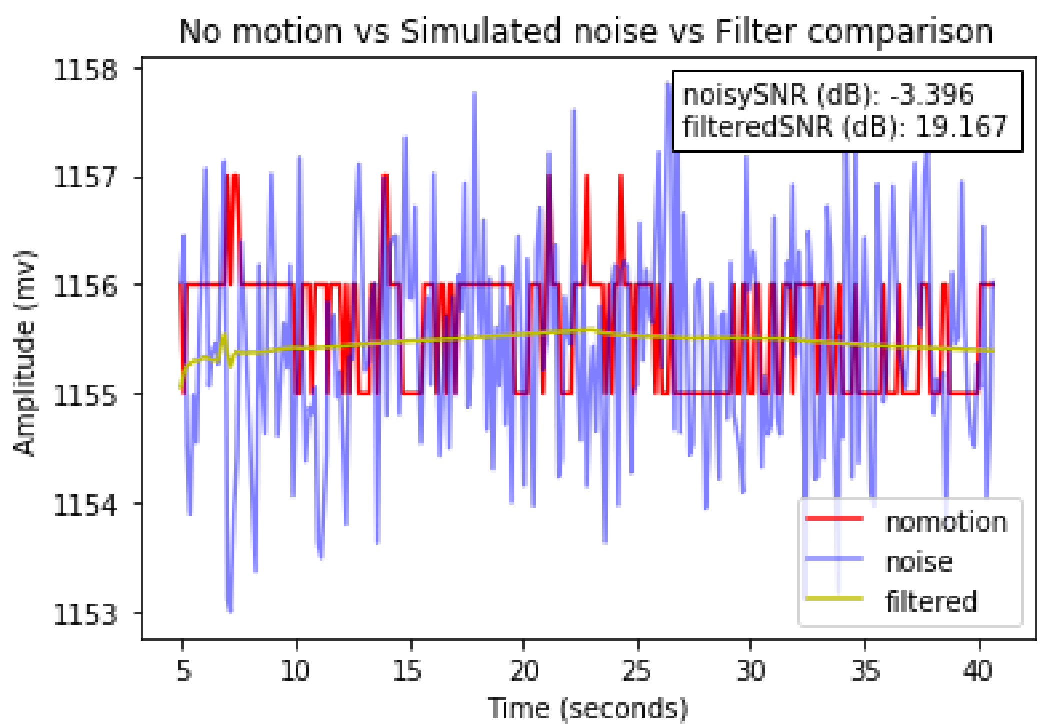
In [167]:
```
%matplotlib inline

import os
import sys
import pywt
import math
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.offsetbox import AnchoredText


# filtering function definitions
def mad(data, axis=None):
    # median absolute deviation
    return np.median(np.absolute(data - np.median(data,axis)), axis)

def waveletSmooth(signal, wavelet, level=1):
    # returns y a rectified and smoothed signal

    # multilevel wavelet decomposition generates coefficients
    coeff = pywt.wavedec(signal, wavelet, mode="per") #by default last axis is used

    # calc a threshold to exclude outliers beyond one median absolute deviation of gau
    sigma = mad(coeff[-level])
    signal_len = len(signal)
    threshold = sigma * np.sqrt(2*np.log(signal_len))
    # Note: alternative distance metrics can be used to vary the threshold
    coeff[1:] =(pywt.threshold(i , value=threshold, mode="soft") for i in coeff[1:])
    #reconstruct signal
    y = pywt.waverec(coeff, wavelet, mode="per")
    return y

def rms(data):
    # returns the root mean squared amplitude of the data
    baseline = np.median(data)
    return np.sqrt(((data - baseline)**2).mean())

def signal_to_noise(signal, noise):
```

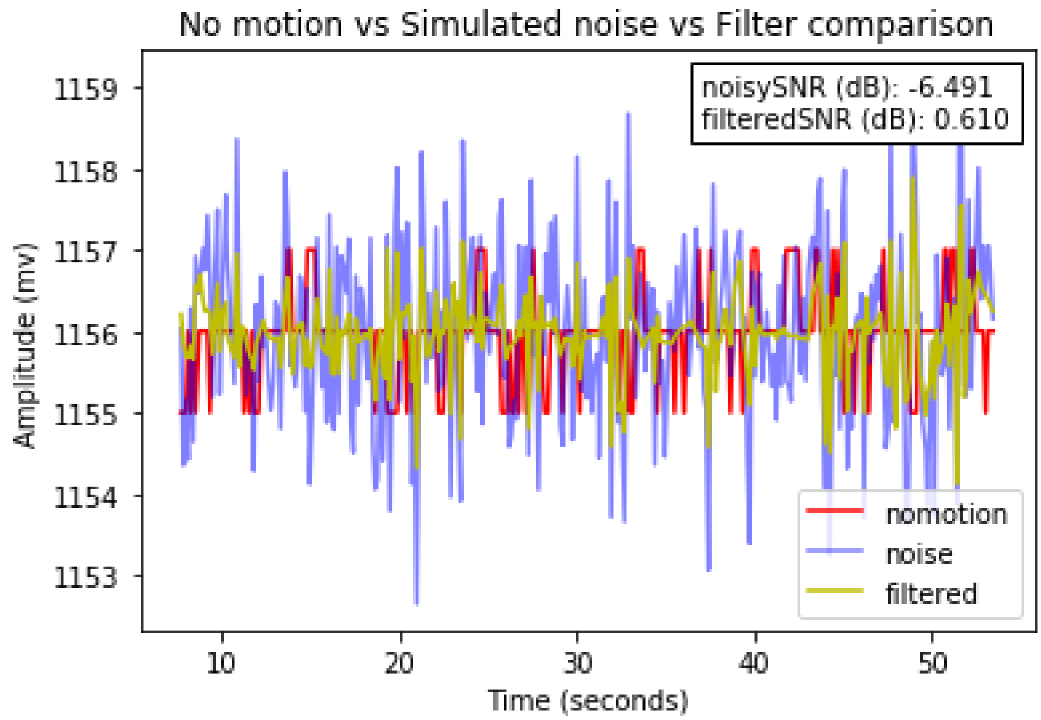
```
    # returns the signal to noise ratio
    # assumption: Equal impedance

    Asignal = rms(signal)
    Anoise = rms(noise)

    if Anoise == 0:
        SNRdb = float('nan')
    else:
        SNRdb = 10*math.log10((Asignal/Anoise)**2)
    return SNRdb
```

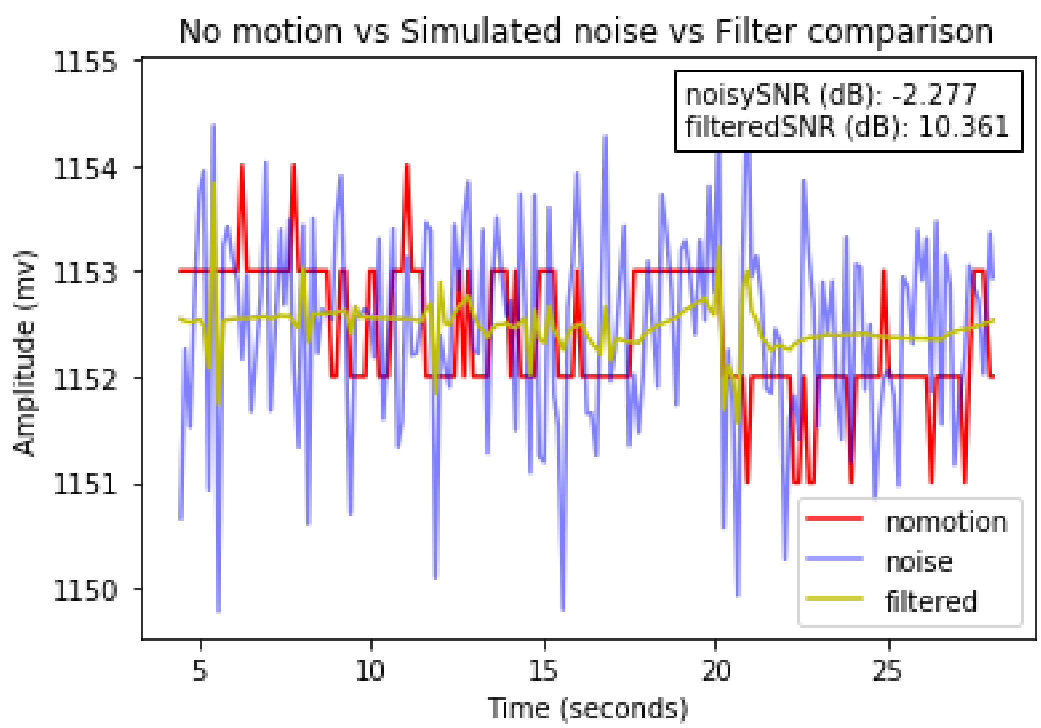
In [175]:
```
# no motion signal vs simulated noise added to signal vs filtered signal analysis
# Author: Anna Lu
# Modified: April 27, 2017

cwd = os.getcwd() + "\\"
nomotion_signal_path = cwd

for filename in os.listdir(nomotion_signal_path):
    signal = []
    if (filename.endswith('.nir')):

        ### fetch data as matrix from file
        data = np.genfromtxt(cwd + filename, delimiter=',')

        num_col = (data.shape[1])
        data = np.array(data[0:].T, dtype=np.float64)
        time = data[0]

        ### simulate addition of gaussian noise

        # select arbitrary nir column from a single wavelength to analyze
        col = 18

        datamean = np.nanmean(data[col])

        noise = np.random.normal(datamean, 1, np.size(data[1:]))
            # datamean is the mean of the normal distribution scaled to center of data
            # 1 is the standard deviation of the normal distribution
            # sqauared size of data = number of elements

        # reshape and add to data except time
        noise = np.reshape(noise, data[1:].shape)
        # scale noise by small number epsilon
        noisydata = data[1:] + noise #* sys.float_info.epsilon

        ### wavlet filter
```

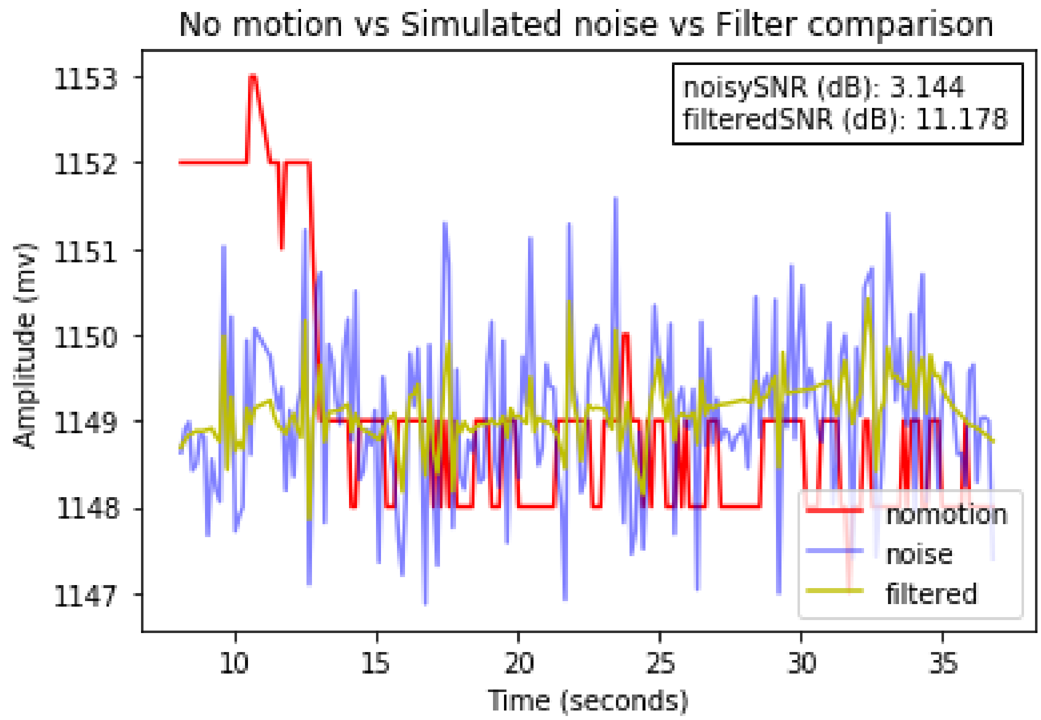
```
wavelet_type = 'db2' # Two decomposition discrete Daubechies wavelet mapping
#noisydataCOL = np.array(noisydata[col], dtype=np.float64)
filtdata = waveletSmooth(noisydata[col], wavelet_type, level=5) # smoothing le

# correct dimension mismatch
if(np.shape(filtdata) != np.shape(noisydata[col])):
    filtdata = filtdata[:-1]

# Signal to noise ratio calculations
# raw no motion data with simulated gaussian noise
noisySNR = signal_to_noise(data[col], noisydata[col])
# filtered signal
filtSNR = signal_to_noise(data[col], filtdata)
SNR_report = ('noisySNR (dB): %.3f\nfilteredSNR (dB): %.3f' %(noisySNR, filtSN


# visualize no motion vs simulated noise vs filter
comp = 'No motion vs Simulated noise vs Filter comparison'
f, ax = plt.subplots(1,1)
plt.title(comp)
plt.ylabel('Amplitude (mv)')
plt.xlabel('Time (seconds)')
p1, = plt.plot(time, data[col], color='r', label='nomotion')
p2, = plt.plot(time, noisydata[col], color='b', alpha=0.5, label='noise')
p3, = plt.plot(time, filtdata, color='y', label='filtered')
#plt.annotate(SNR_report, xy=(1.05, 0.8))
txt = AnchoredText(SNR_report, loc=1)
ax.add_artist(txt)
plt.legend(handles=[p1, p2, p3], loc=4)
plt.show()
```

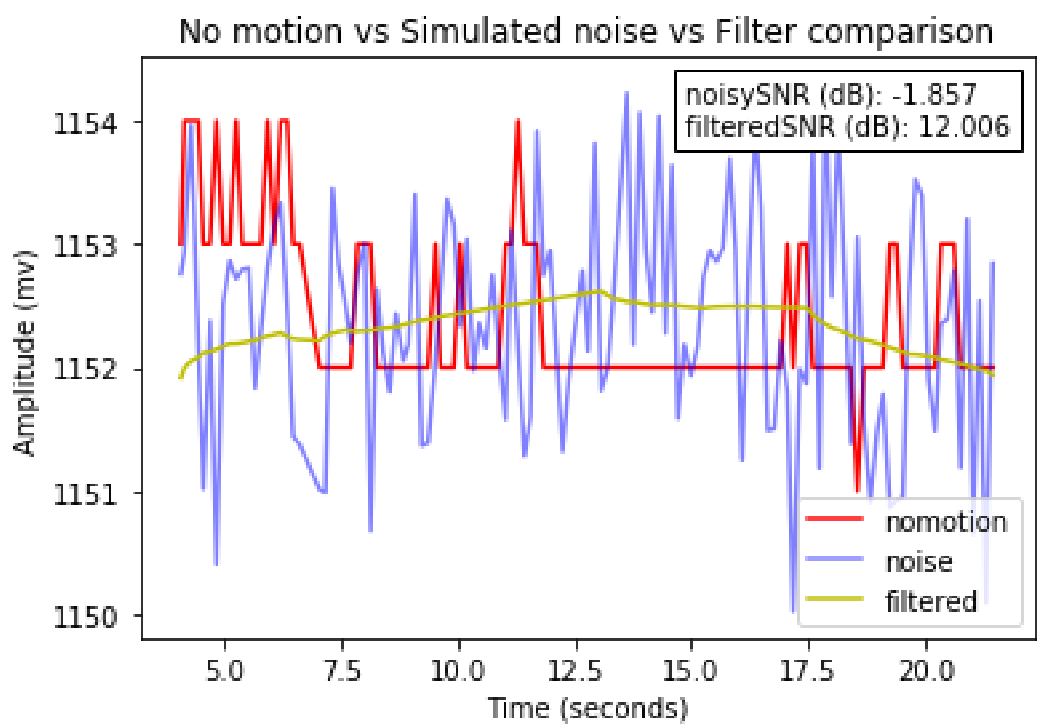

No motion vs Simulated noise vs Filter comparison
noisySNR (dB): -4.207
filteredSNR (dB): 2.052
1158
1157
1156
1155
1154
1153
Amplitude (mv)
nomotion
noise
filtered
5
10
15
20
25
30
35
40
Time (seconds)

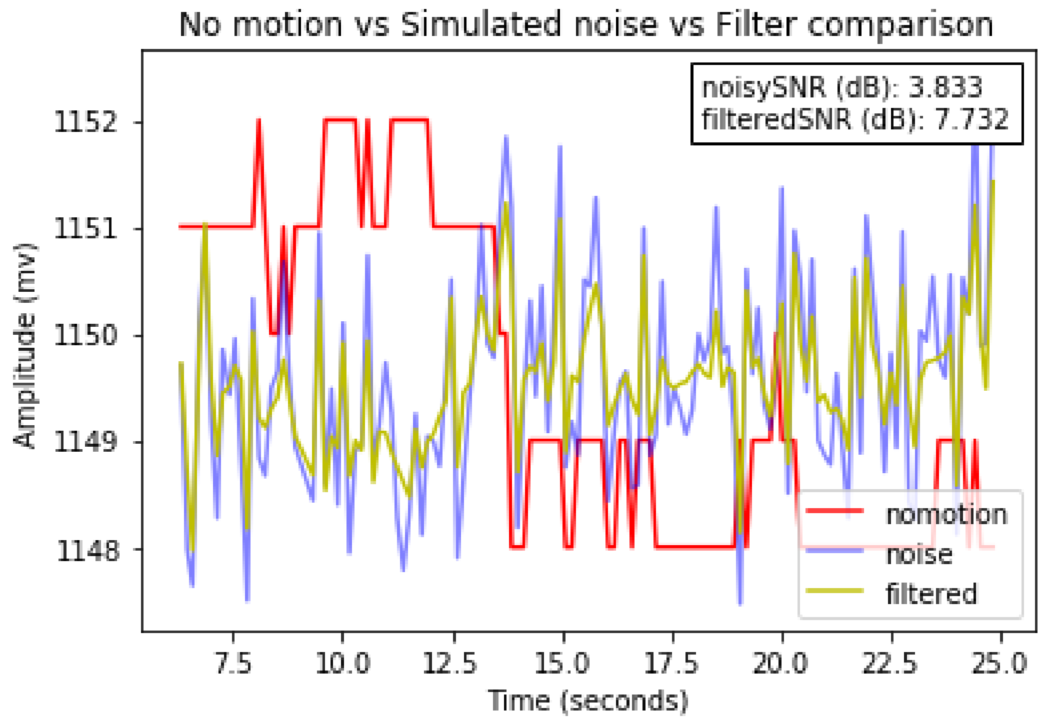

No motion vs Simulated noise vs Filter comparison
noisySNR (dB): -3.396
filteredSNR (dB): 19.167
1158
1157
1156
1155
1154
1153
Amplitude (mv)
nomotion
noise
filtered
5
10
15
20
25
30
35
40
Time (seconds)

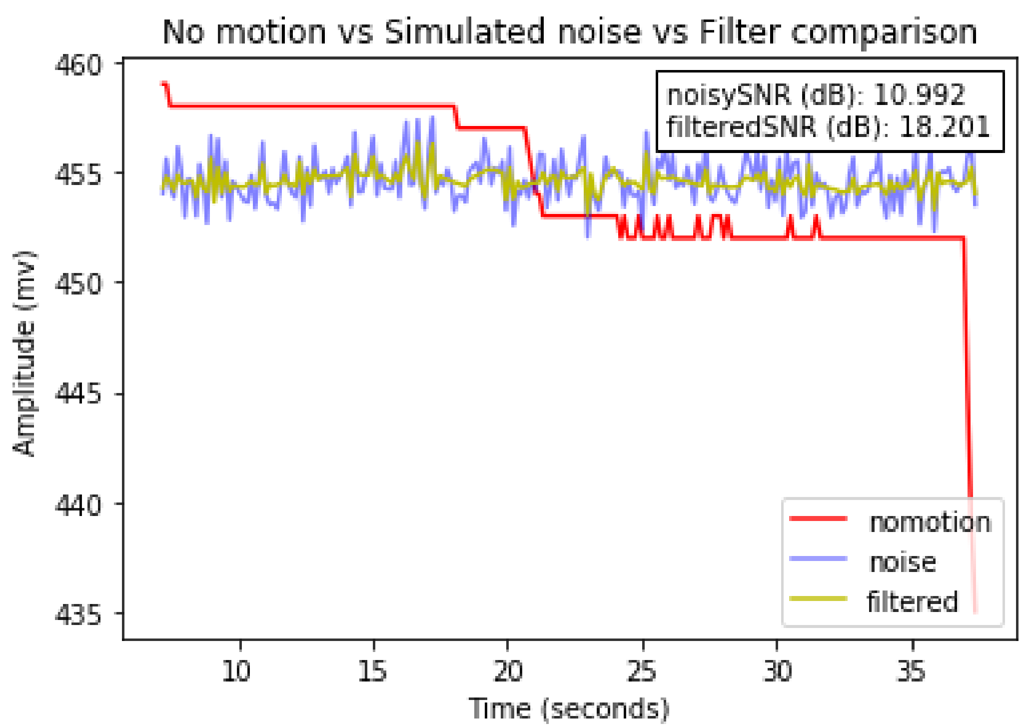

No motion vs Simulated noise vs Filter comparison
noisySNR (dB): -6.491
filteredSNR (dB): 0.610
nomotion
noise
filtered
Amplitude (mv)
Time (seconds)
1159
1158
1157
1156
1155
1154
1153
10
20
30
40
50


No motion vs Simulated noise vs Filter comparison
noisySNR (dB): -2.277
filteredSNR (dB): 10.361
nomotion
noise
filtered
Amplitude (mv)
Time (seconds)
1155
1154
1153
1152
1151
1150
5
10
15
20
25


No motion vs Simulated noise vs Filter comparison
noisySNR (dB): 3.144
filteredSNR (dB): 11.178
Amplitude (mv)
Time (seconds)
nomotion
noise
filtered


No motion vs Simulated noise vs Filter comparison
noisySNR (dB): -1.857
filteredSNR (dB): 12.006
Amplitude (mv)
Time (seconds)
nomotion
noise
filtered


No motion vs Simulated noise vs Filter comparison
noisySNR (dB): 3.833
filteredSNR (dB): 7.732
Amplitude (mv)
1152
1151
1150
1149
1148
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
Time (seconds)
nomotion
noise
filtered


No motion vs Simulated noise vs Filter comparison
noisySNR (dB): 10.992
filteredSNR (dB): 18.201
Amplitude (mv)
460
455
450
445
440
435
10
15
20
25
30
35
Time (seconds)
nomotion
noise
filtered

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-175-93ef04b7b2cf> in <module>()
     45         # Signal to noise ratio calculations
     46         # raw no motion data with simulated gaussian noise
---> 47         noisySNR = signal_to_noise(data[col], noisydata[col])
     48         # filtered signal
     49         filtSNR = signal_to_noise(data[col], filtdata)


<ipython-input-167-d3285d62036b> in signal_to_noise(signal, noise)
     46         SNRdb = float('nan')
     47     else:
---> 48         SNRdb = 10*math.log10((Asignal/Anoise)**2)
     49     return SNRdb


ValueError: math domain error
```

```
In [ ]:
```

```
In [ ]:
```