simulated_noise_filter

May 1, 2017

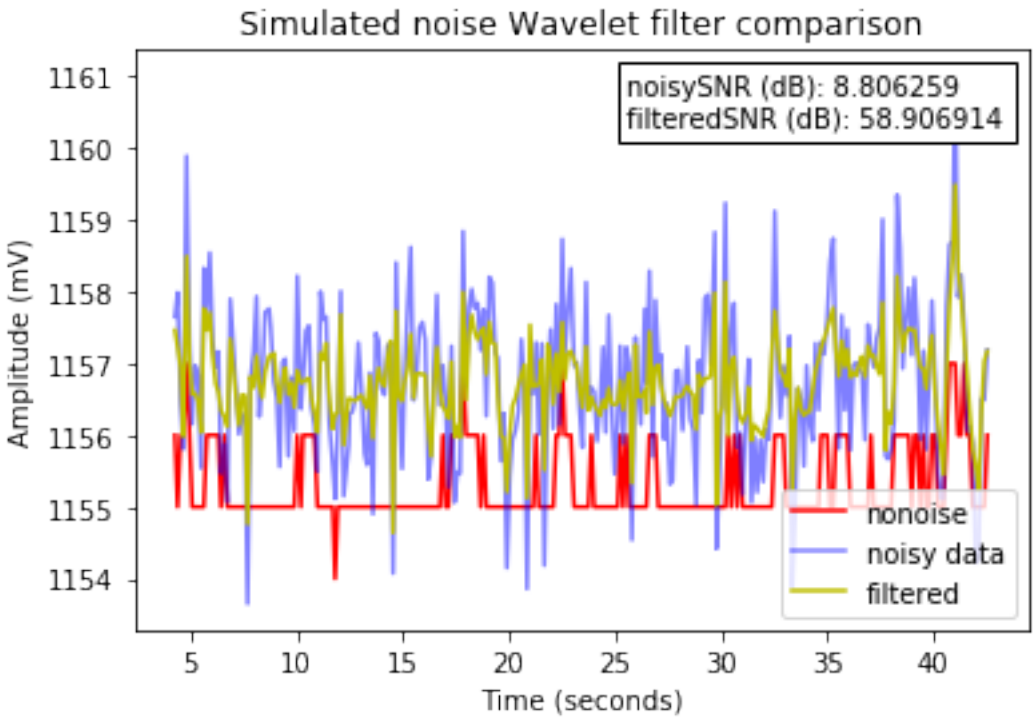
In [70]:
```
%matplotlib inline

import os
import sys
import pywt
import numpy as np
from numpy import mean, sqrt, square, log10, isnan, isscalar, asscalar
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 gaus
    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

    # check and fill any nan
    if np.isnan(data).any():
        data = data[~np.isnan(data).any(axis=1)]
```

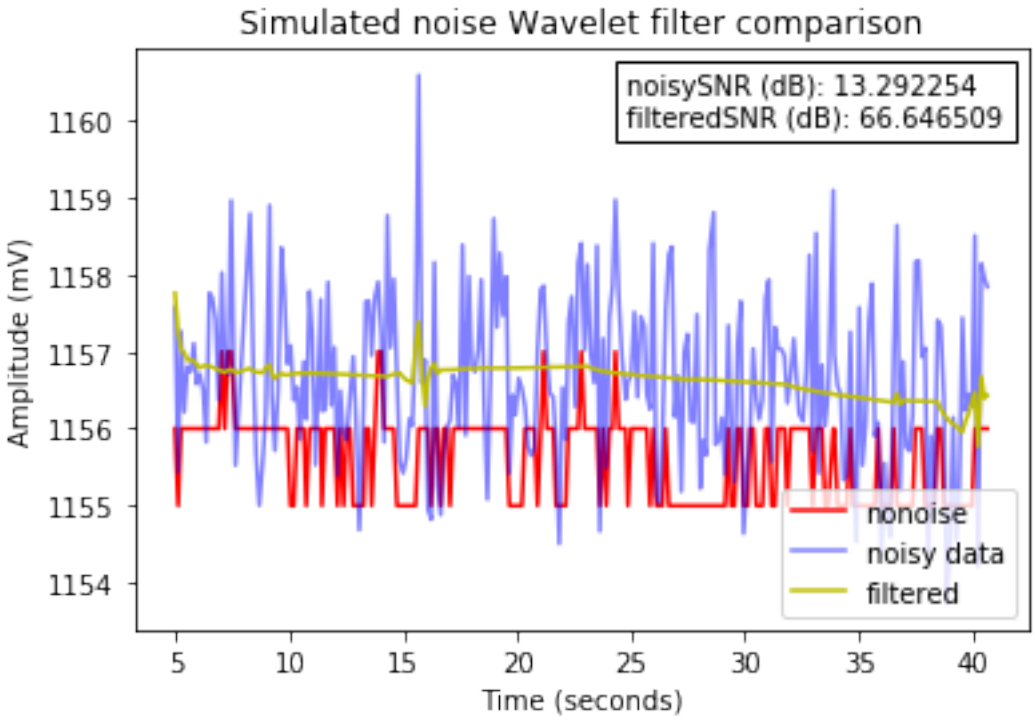
```
        B = np.min(data)
        rms = sqrt(mean(square(data - B)))
        #print(rms)

        # convert nonscalar rms to a scalar float
        if ~isscalar(rms):
            #print('rms is nonscalar')
            rms = asscalar(rms)

        return rms

    def signal_to_noise(signal, noise):
        # returns the signal to noise ratio
        # assumption: Equal impedance

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

        #if ~isscalar(Anoise):
        #    print('Anoise is nonscalar')
        #    break

        if Anoise == 0:
            SNRdb = float('nan')
        else:
            SNRdb = -10*log10((Asignal/Anoise)**2)

        return SNRdb
```

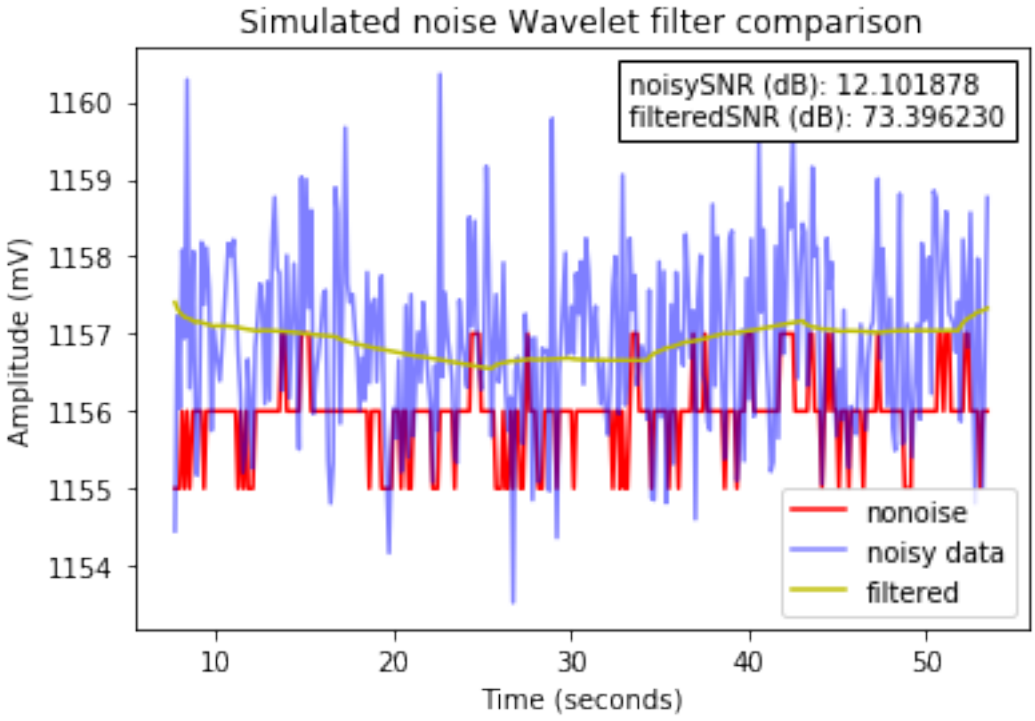
In [71]:
```
    # Simulated noise comparative verification of Wavelet Filter
    # no noise signal vs simulated noise added to signal vs filtered signal
    # Author: Anna Lu
    # Modified: May 1, 2017

    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]
```

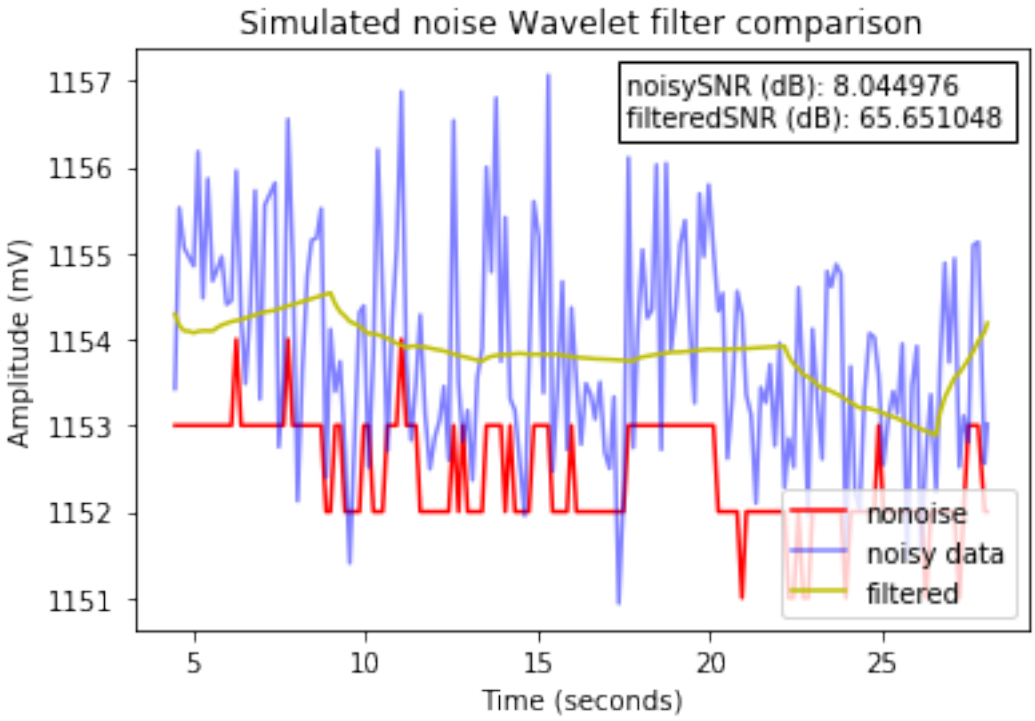
```
### simulate addition of gaussian noise

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

datapkpk = abs(max(data[col]) - min(data[col]))

pkpkmean = (datapkpk/2)

#print(pkpkmean)

noise = np.random.normal(pkpkmean, 1, np.size(data[1:]))

    # pkpkmean is the mean peak to peak where the normal distribution centered
    # 1 is the standard deviation of the normal distribution
    # sqauared size of data = number of elements

# reshape and add to data except time
# data[1:]
noise = np.reshape(noise, data[1:].shape)
noisydata = data[col] + noise

### 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 lev

# 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], noise) #noisydata[col])

# filtered noise for SNR comparison
filterednoise = filtdata - data

filtSNR = signal_to_noise(filtdata, filterednoise)
# filtSNR = signal_to_noise(filtdata[col], noise)
SNR_report = ('noisySNR (dB): %f\nfilteredSNR (dB): %f' %(noisySNR, filtSNR))

#print(type(filtSNR))
# visualize no noise vs simulated noise vs filter
comp = 'Simulated noise Wavelet filter comparison'
f, ax = plt.subplots(1,1)
plt.title(comp)
```

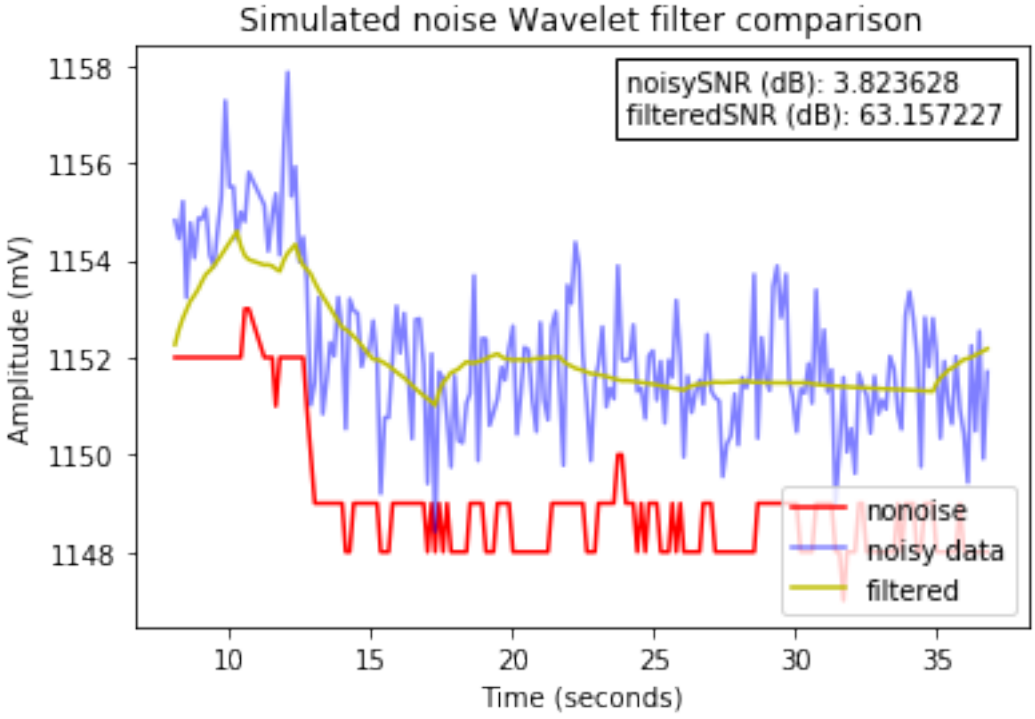
```
plt.ylabel('Amplitude (mV)')
plt.xlabel('Time (seconds)')
p1, = plt.plot(time, data[col], color='r', label='nonoise')
p2, = plt.plot(time, noisydata[col], color='b', alpha=0.5, label='noisy data')
p3, = plt.plot(time, filtdata, color='y', label='filtered')
#plt.annotate(SNR_report, xy=(1.05, 0.8))
txt = AnchoredText(SNR_report, loc=1)
ax.add_artist(txt)
plt.legend(handles=[p1, p2, p3], loc=4)
plt.show()
```

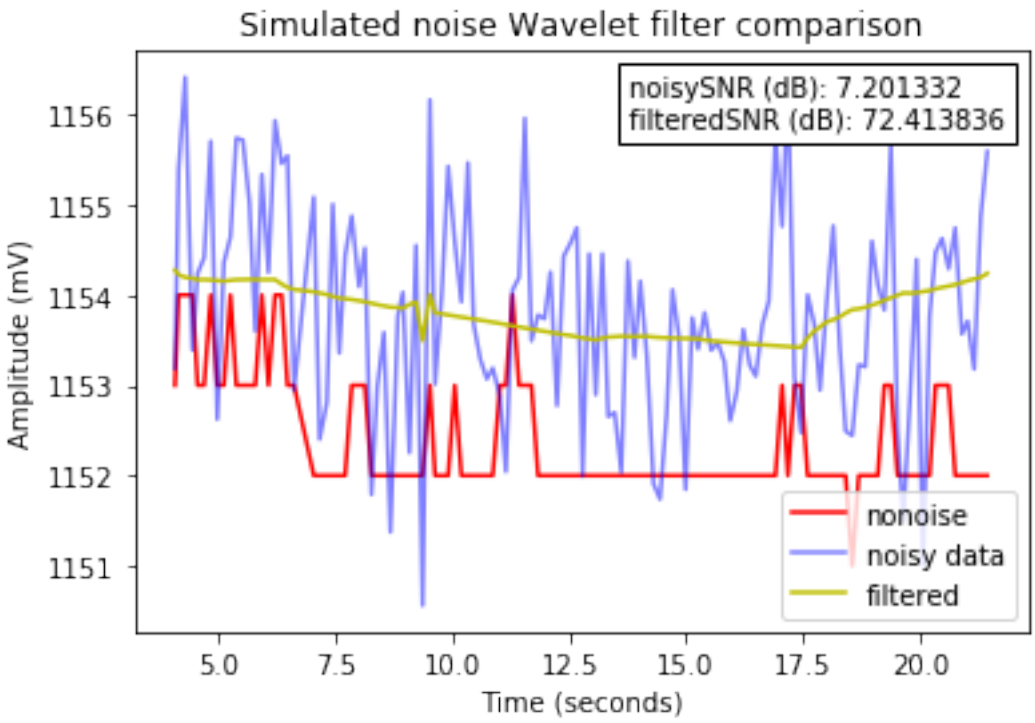


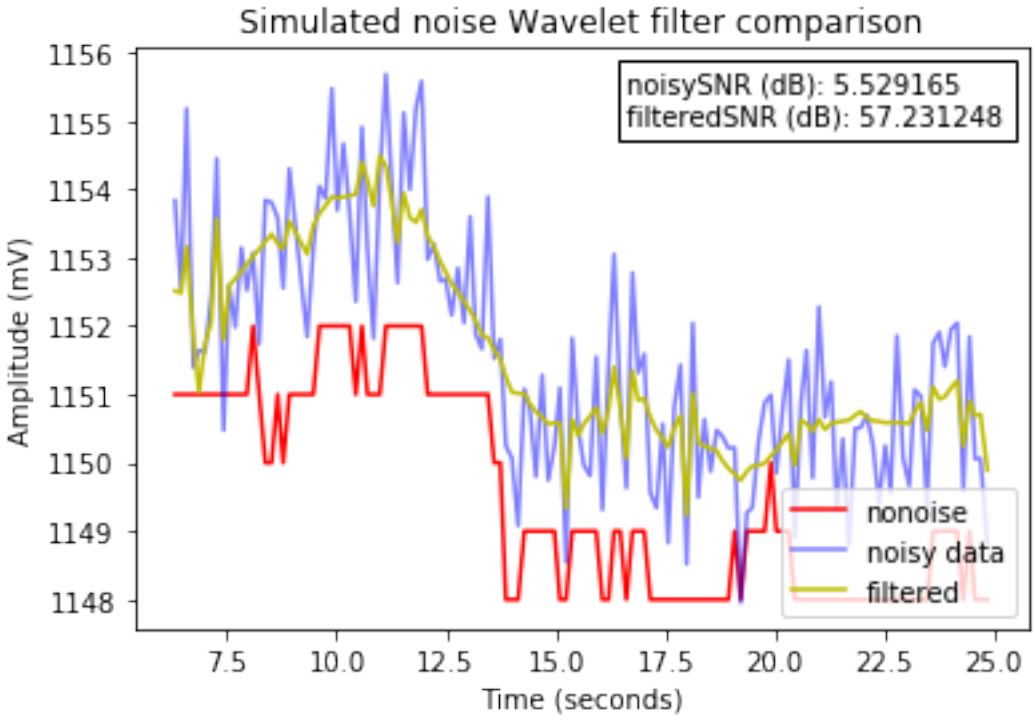

Simulated noise Wavelet filter comparison
noisySNR (dB): 13.292254
filteredSNR (dB): 66.646509
Amplitude (mV)
Time (seconds)
nonoise
noisy data
filtered

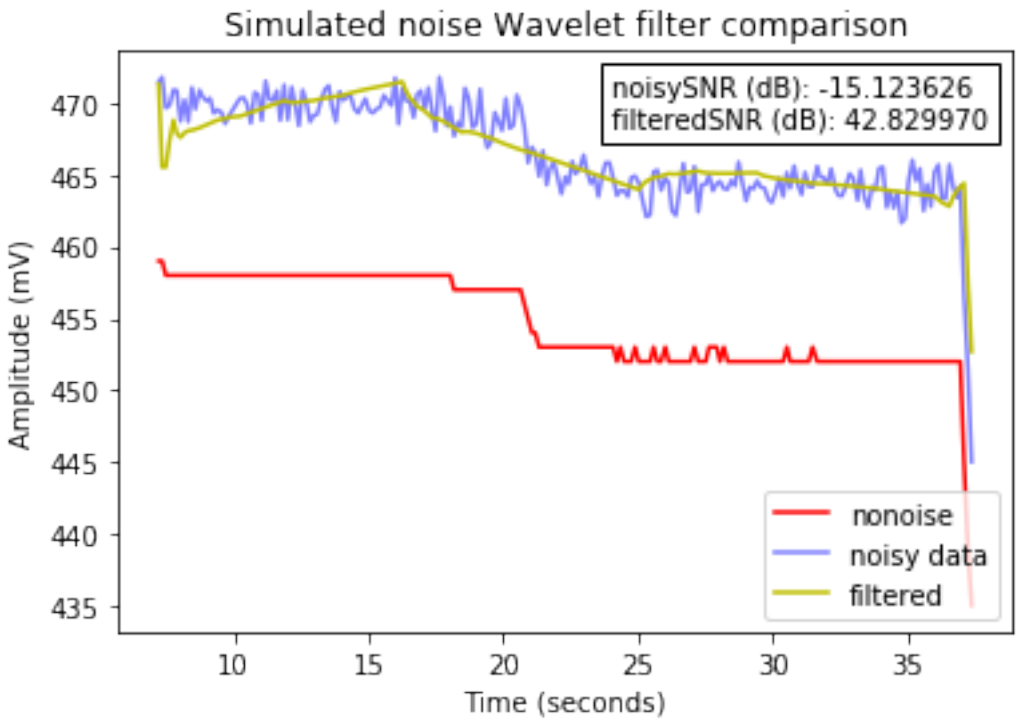

Simulated noise Wavelet filter comparison
noisySNR (dB): 12.101878
filteredSNR (dB): 73.396230
Amplitude (mV)
Time (seconds)
nonoise
noisy data
filtered


Simulated noise Wavelet filter comparison
noisySNR (dB): 8.044976
filteredSNR (dB): 65.651048
nonoise
noisy data
filtered
Amplitude (mV)
Time (seconds)
1157
1156
1155
1154
1153
1152
1151
5
10
15
20
25

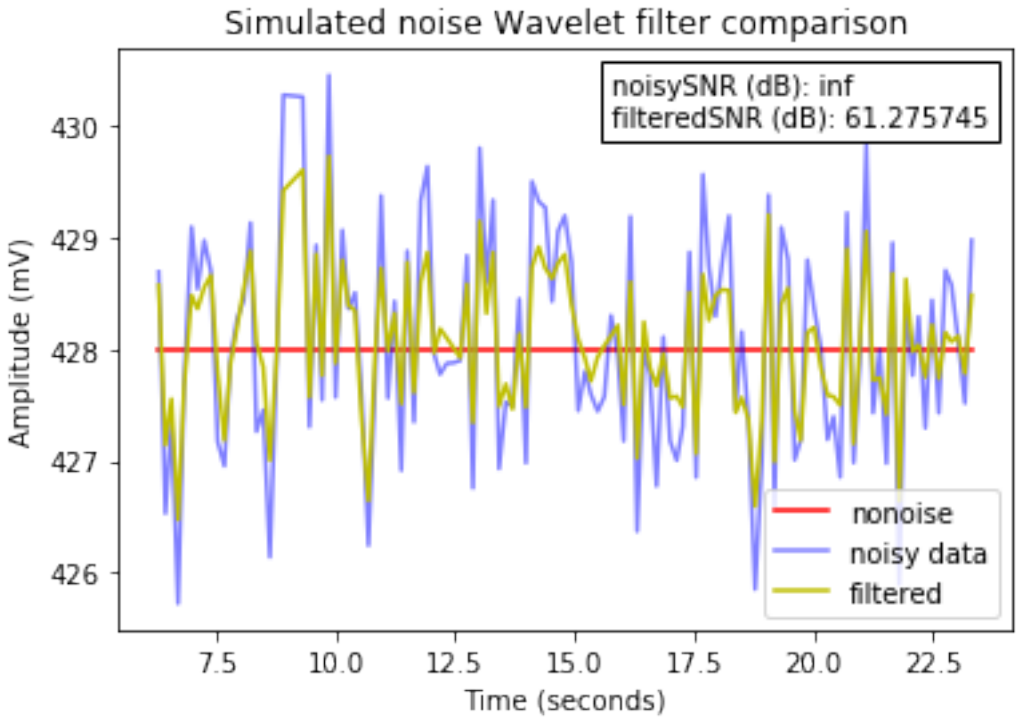

Simulated noise Wavelet filter comparison
noisySNR (dB): 3.823628
filteredSNR (dB): 63.157227
nonoise
noisy data
filtered
Amplitude (mV)
Time (seconds)
1158
1156
1154
1152
1150
1148
10
15
20
25
30
35

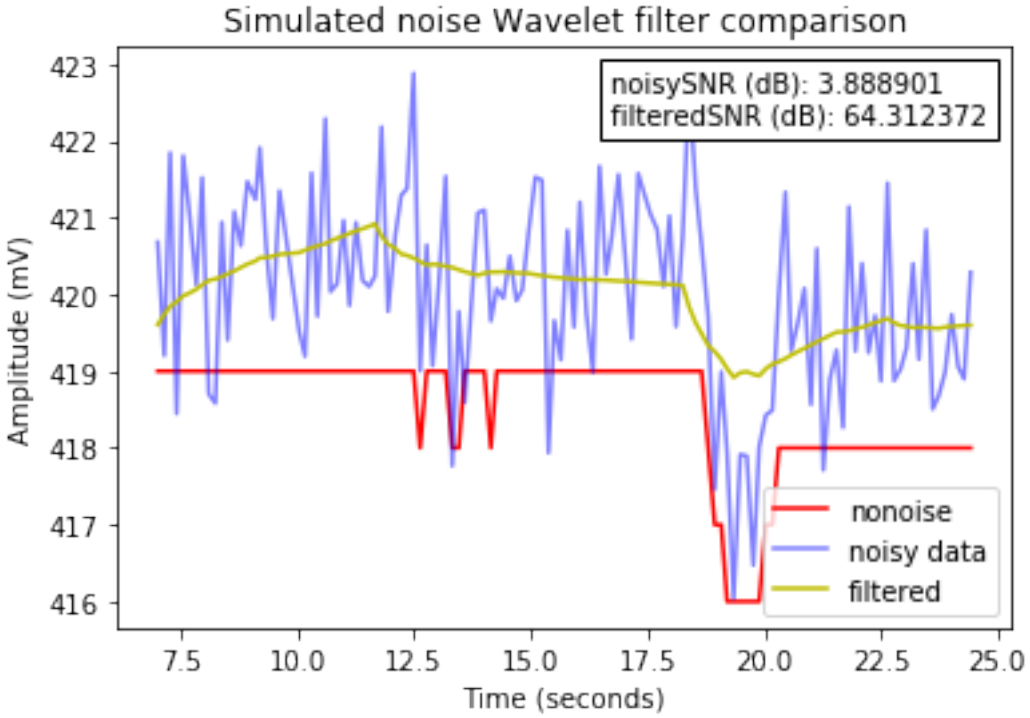

Simulated noise Wavelet filter comparison
noisySNR (dB): 7.201332
filteredSNR (dB): 72.413836
nonoise
noisy data
filtered
Amplitude (mV)
Time (seconds)
1156
1155
1154
1153
1152
1151
5.0
7.5
10.0
12.5
15.0
17.5
20.0

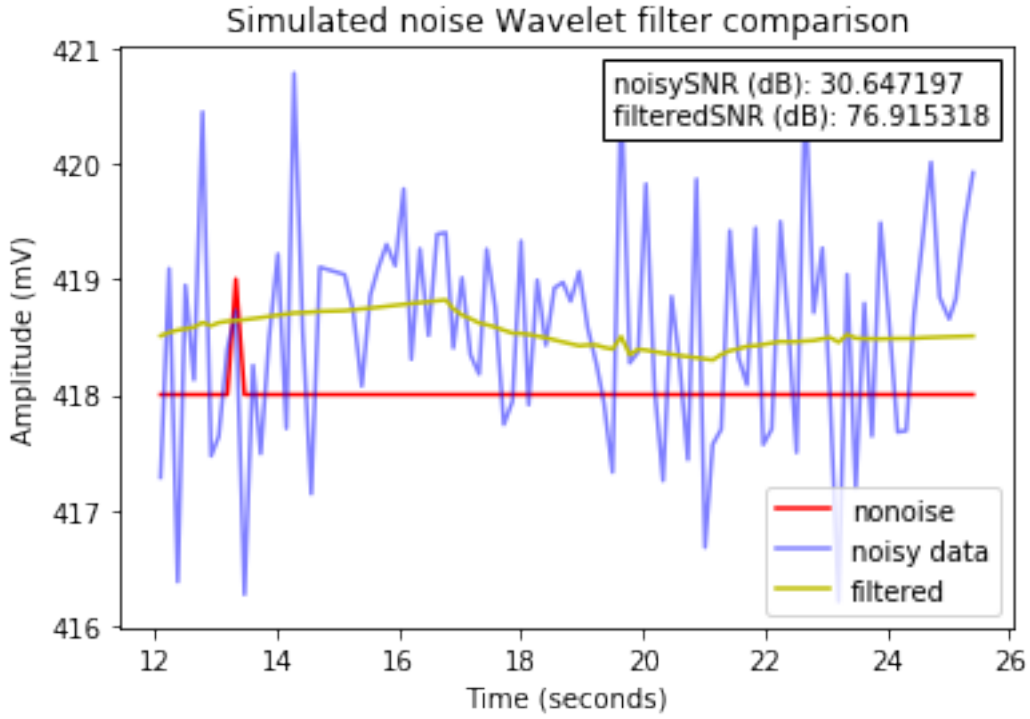

Simulated noise Wavelet filter comparison
noisySNR (dB): 5.529165
filteredSNR (dB): 57.231248
nonoise
noisy data
filtered
Amplitude (mV)
Time (seconds)
1156
1155
1154
1153
1152
1151
1150
1149
1148
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0

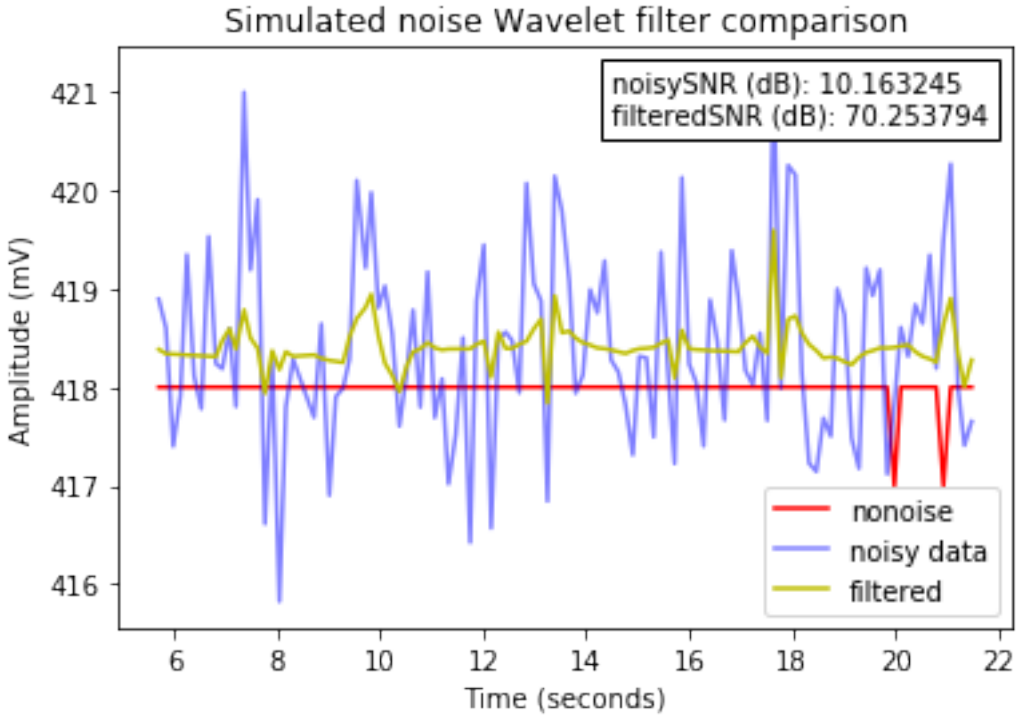


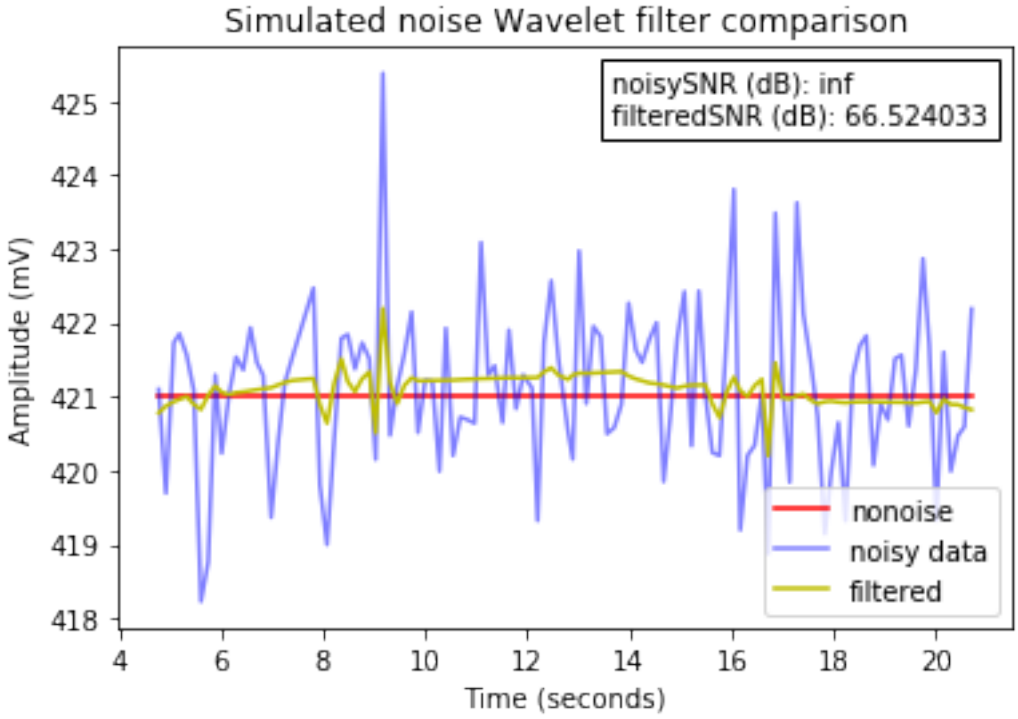

```
c:\users\anna\appdata\local\programs\python\python36-32\lib\site-packages\ipykernel\__main__.py:
```

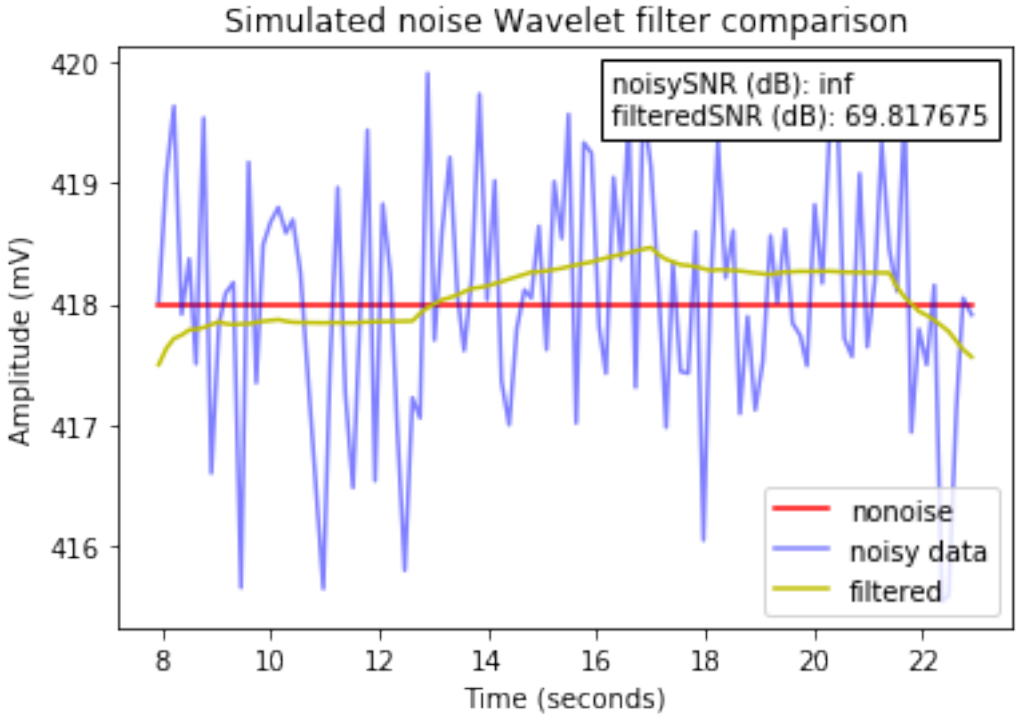


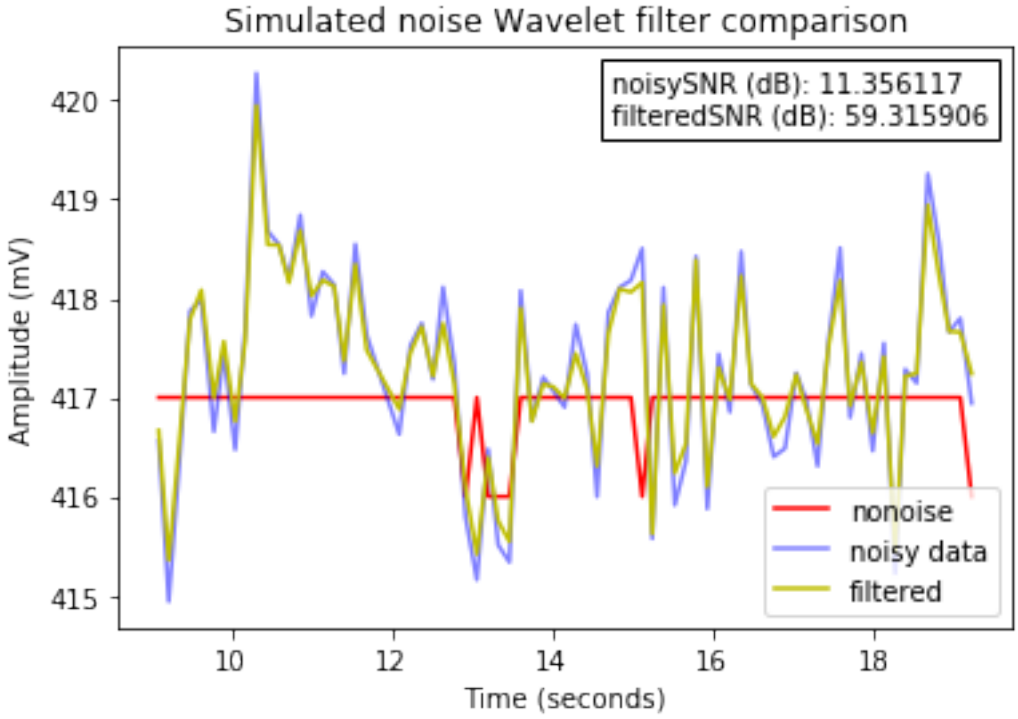

Simulated noise Wavelet filter comparison
noisySNR (dB): 3.888901
filteredSNR (dB): 64.312372
nonoise
noisy data
filtered
Amplitude (mV)
Time (seconds)
423
422
421
420
419
418
417
416
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0

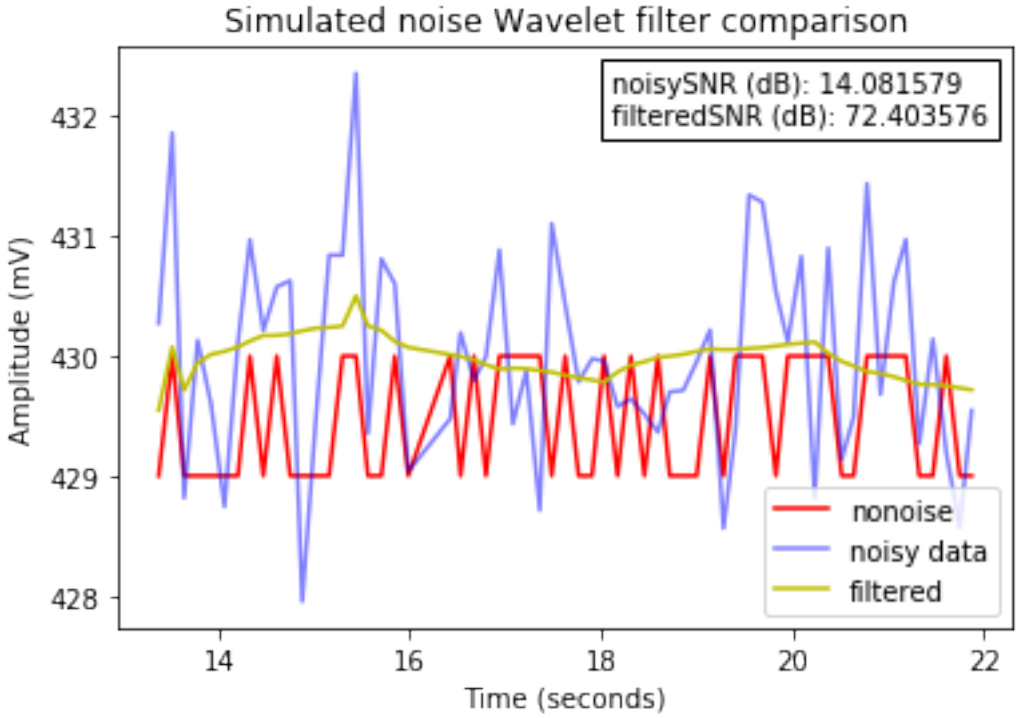

Simulated noise Wavelet filter comparison
noisySNR (dB): 30.647197
filteredSNR (dB): 76.915318
nonoise
noisy data
filtered
Amplitude (mV)
Time (seconds)
421
420
419
418
417
416
12
14
16
18
20
22
24
26

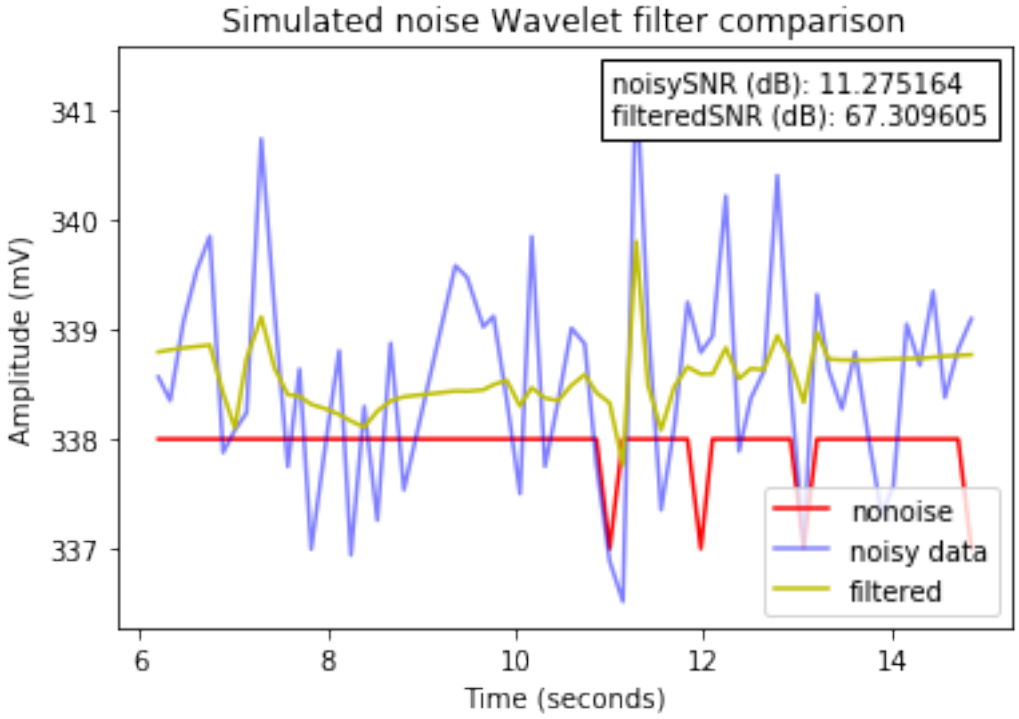

Simulated noise Wavelet filter comparison
noisySNR (dB): 10.163245
filteredSNR (dB): 70.253794
nonoise
noisy data
filtered
Amplitude (mV)
Time (seconds)
421
420
419
418
417
416
6
8
10
12
14
16
18
20
22

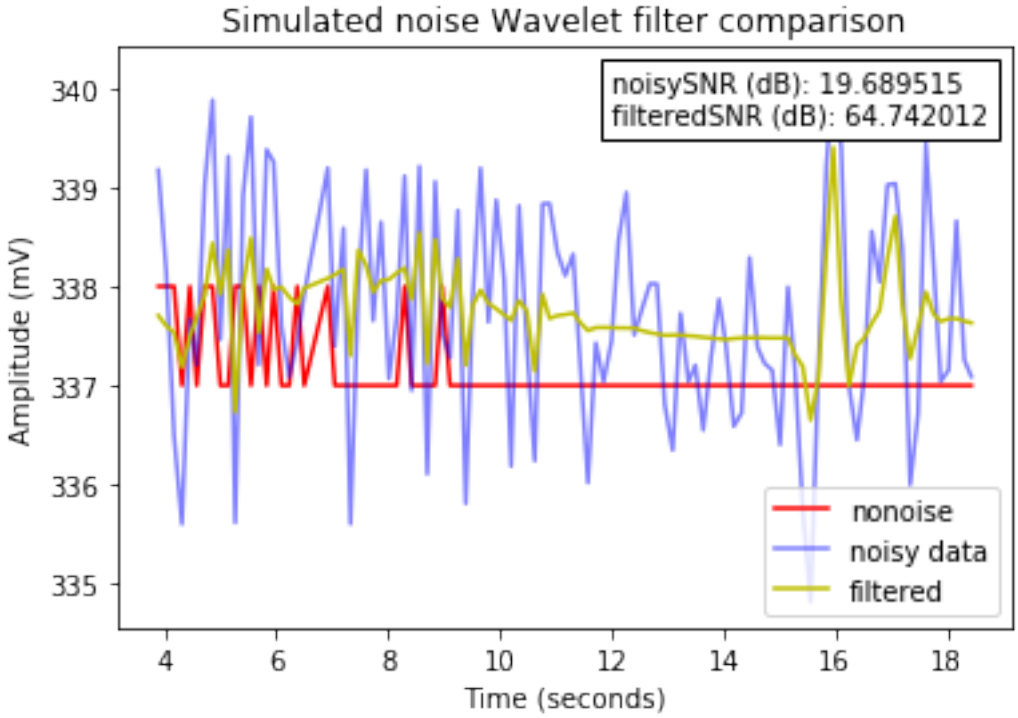

Simulated noise Wavelet filter comparison
noisySNR (dB): inf
filteredSNR (dB): 66.524033
nonoise
noisy data
filtered
Amplitude (mV)
Time (seconds)
425
424
423
422
421
420
419
418
4
6
8
10
12
14
16
18
20


Simulated noise Wavelet filter comparison
noisySNR (dB): inf
filteredSNR (dB): 69.817675
nonoise
noisy data
filtered
Amplitude (mV)
Time (seconds)
420
419
418
417
416
8
10
12
14
16
18
20
22


Simulated noise Wavelet filter comparison
noisySNR (dB): 11.356117
filteredSNR (dB): 59.315906
nonoise
noisy data
filtered
Amplitude (mV)
Time (seconds)
420
419
418
417
416
415
10
12
14
16
18


Simulated noise Wavelet filter comparison
noisySNR (dB): 14.081579
filteredSNR (dB): 72.403576
Amplitude (mV)
Time (seconds)
nonoise
noisy data
filtered


Simulated noise Wavelet filter comparison
noisySNR (dB): 11.275164
filteredSNR (dB): 67.309605
Amplitude (mV)
Time (seconds)
nonoise
noisy data
filtered


Simulated noise Wavelet filter comparison
noisySNR (dB): 19.689515
filteredSNR (dB): 64.742012
nonoise
noisy data
filtered
Amplitude (mV)
Time (seconds)
340
339
338
337
336
335
4
6
8
10
12
14
16
18

In [ ]: