Problem 1:

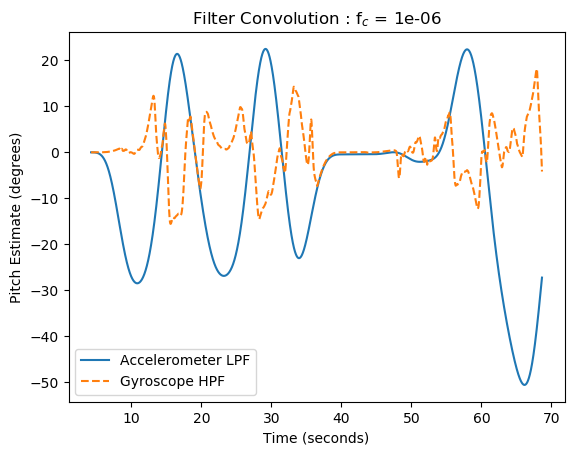
The accelerometer pitch angle data seems to be more accurate. The gyroscope is more sensitive to fast, aggressive movements, and the quadcopter was not subjected to very many sudden movements during the test. The gyroscope also experiences drift over time and must make an assumption of where the zero point is using the data collected at the beginning of the test. The accelerometer is able to self-orient and can determine position even without perfect initial conditions upon startup. On the downside, the accelerometer generates data with a lot of noise due to the sensitivity of the instrument.

Problem 2:

The finite impulse response (FIR) filter does not require filter output from a previous iteration to filter the data. It is always stable, but might be more computationally expensive, requiring a higher order filter to get an adequate response. If we do not know anything about the dataset and we are filtering the data after collection, it might be best to use a FIR filter to get a sense of what the data should look like.

An infinite impulse response (IIR) filter requires that the existing filter data output gets fed back into the filtering equation. This means that the filter is not able to attenuate the first few values until the previous filter outputs are calculated and these values find their way back into the filtering equation. These filter types are most commonly the named filters. These filters do not require as many computations to get the same performance as the FIR filters, so this would be a good option for when we need real time live filtering for use in a feedback loop.

Problem 3:

This filter output was generated using a sinc filter with a blackman window because it is easier to see the difference between the high pass and low pass filtering schemes using the FIR filter. A transition band of 0.005 was used because it yielded datasets that ran along the same scale and somewhat resembled the expected raw data output. Increasing the transition band resulted in a larger time lag between the start of the dataset and the filter output response. Using the same filter parameters for both pitch estimates produced a matching slope in the gyroscope during the steepest accelerometer outputs but tended to generate noise while the accelerometer was still tracking with the physical movement of the quadcopter.

|  |
| --- |
| # standard libraries |
|  | import math, os |
|  | # external libraries (for convolve, sinc, arrays, csv parsing, and plotting) |
|  | import numpy as np, matplotlib.pyplot as plt |
|  |  |
|  | def butter2(f\_c, f\_s, in\_list): |
|  | ''' |
|  | 2nd Order Butterworth Filter: |
|  | f\_c = cutoff frequency - adjust this to change amount of filtering |
|  | f\_s = sampling frequency - determined by the sampling rate of the dataset |
|  | in\_list = list of collected data points to be filtered |
|  | ''' |
|  | gam = math.tan((math.pi\*f\_c)/f\_s) |
|  | coef\_d = gam\*\*2 + (2\*\*0.5)\*gam + 1 |
|  | a\_1 = (2 \* (gam\*\*2 - 1)) / coef\_d |
|  | a\_2 = (gam\*\*2 - (2\*\*0.5)\*gam + 1)/coef\_d |
|  | b\_0 = (gam\*\*2) / coef\_d |
|  | b\_1 = (2\*b\_0) / coef\_d |
|  | b\_2 = b\_0 |
|  | x\_n1 = 0 |
|  | x\_n2 = 0 |
|  | y\_n1 = 0 |
|  | y\_n2 = 0 |
|  | out\_list = [] |
|  | for value in in\_list: |
|  | y\_n = b\_0\*value + b\_1\*x\_n1 + b\_2\*x\_n2 - a\_1\*y\_n1 - a\_2\*y\_n2 |
|  | out\_list.append(y\_n) |
|  | x\_n2 = x\_n1 |
|  | x\_n1 = value |
|  | y\_n2 = y\_n1 |
|  | y\_n1 = y\_n |
|  |  |
|  | return out\_list |
|  |  |
|  | def fir\_wind(f\_c, t\_b, in\_list, highpass=False): |
|  | ''' |
|  | Windowed sinc filter for low or high pass filtering using a blackman window |
|  | f\_c = cutoff frequency |
|  | t\_b = transition bandwidth |
|  | in\_list = list of collected data points to be filtered |
|  | ''' |
|  | # big n |
|  | b\_n = int(math.ceil(4/t\_b)) |
|  | if not b\_n % 2 : b\_n += 1 # make big n odd |
|  | l\_n = np.arange(b\_n) |
|  |  |
|  | # sinc filter calculation |
|  | h = np.sinc(2 \* f\_c \* (l\_n - (b\_n - 1) / 2.)) |
|  |  |
|  | # blackman window |
|  | w = 0.42 - 0.5 \* np.cos((2 \* np.pi \* l\_n)/(b\_n - 1)) + \ |
|  | 0.08 \* np.cos((4 \* np.pi \* l\_n)/(b\_n - 1)) |
|  |  |
|  | # combine sinc filter with blackman window |
|  | h = h \* w |
|  |  |
|  | # normalize the filter to get unity gain |
|  | h = h / sum(h) |
|  |  |
|  | if highpass: |
|  | # generate high pass filter through spectral inversion |
|  | h = -h |
|  | h[int((b\_n - 1) / 2.)] += 1 |
|  |  |
|  | # apply the filter to the signal |
|  | sig = np.convolve(in\_list, h) |
|  | return sig |
|  |  |
|  | for file\_name in os.listdir(): |
|  | if file\_name.startswith('final'): |
|  | d\_time, pitch\_acc, pitch\_gyro = np.loadtxt(open(file\_name), |
|  | delimiter = ',', |
|  | unpack = True, |
|  | usecols = (1,30, 31), |
|  | skiprows = 2) |
|  |  |
|  | d\_time /= 1000000 # convert time from microseconds to milliseconds |
|  | cutoff = .000001 # cutoff frequency |
|  | tran\_band = 0.005 # transition bandwdth |
|  | sample = 100 # Sampling frequency is 100Hz |
|  |  |
|  | # f\_acc = butter2(cutoff, sample, pitch\_acc) # Filtered accelerometer |
|  | # f\_gyro = butter2(cutoff, sample, pitch\_gyro) # Filtered gyrometer |
|  |  |
|  | lp\_acc = fir\_wind(cutoff, tran\_band, pitch\_acc) # Filtered accelerometer |
|  | hp\_gyr = fir\_wind(cutoff, tran\_band, pitch\_gyro, highpass=True) # Filtered gyrometer |
|  |  |
|  | # Problem 1 plot |
|  | plt.plot(d\_time, pitch\_acc, label='Accelerometer') |
|  | plt.plot(d\_time, -pitch\_gyro,'--', label='Gyroscope') |
|  | plt.title('Accelerometer vs. Gyroscope Pitch Estimates') |
|  | plt.xlabel('Time (seconds)') |
|  | plt.ylabel('Pitch Estimate (degrees)') |
|  | plt.legend() |
|  | # plt.savefig('ME457\_HW3\_P1.png', bbox\_inches='tight') |
|  | plt.show() |
|  |  |
|  | # Problem 2 plot |
|  | plt.plot(d\_time, lp\_acc[0:len(d\_time)], label='Accelerometer LPF') |
|  | plt.plot(d\_time, hp\_gyr[0:len(d\_time)], '--', label='Gyroscope HPF') |
|  | plt.title(r'Filter Convolution : f$\_c$ = {}'.format(cutoff)) |
|  | plt.xlabel('Time (seconds)') |
|  | plt.ylabel('Pitch Estimate (degrees)') |
|  | plt.legend() |
|  | plt.savefig('ME457\_HW3\_P3.png', bbox\_inches='tight') |
|  | plt.show() |