Gina Jozef
4/5/22

ATOC5860 – Application Lab #5
Filtering Timeseries
Spring 2022

Notebook #1 – ATOC5860_applicationlab5
ATOC5860_applicationlab5_check_python_convolution.ipynb

LEARNING GOAL
1) Understand what is happening "under the hood" in different python functions that are used to smooth data in the time domain.

Use this notebook to understand the different python functions that can be used to smooth data in the time domain. Compare with a "by hand" convolution function. Look at your data by printing its shape and also values. Understand what the python function is doing, especially how it is treating edge effects.

The "by hand" method results in the same first three and last three points as np.convolve either when no mode is specified, or when "full" mode is specified (because the default mode is "full" when none is specified). Using the "same" mode, the size of the filtered array is the same size as the original array, but using the "valid" mode results in a filtered array that has 2 fewer values than the original array, and using "full" results in a filtered array with 2 extra values than the original array. Below describes that each python function for filtering does:

- np.convolve (default is 'full')
  - 'full'
    - starts with the first value and add points at the end - matches calculation by hand at both start and end
    - This returns the convolution at each point of overlap, with an output shape of (N+M-1,).
    - At the end-points of the convolution, the signals do not overlap completely, and boundary effects may be seen.
  - 'same'
    - Mode 'same' returns output of length max(M, N). Boundary effects are still visible.
  - 'valid'
    - Mode 'valid' returns output of length max(M, N) - min(M, N) + 1.
    - The convolution product is only given for points where the signals overlap completely.
    - Values outside the signal boundary have no effect.
- sig
  - sig.lfilter
    - uses np.convolve so it also starts with the first value, but cuts off the last value (at the end)
  - sig.filtfilt do

- apply the filter forward - cut it off second to last
- apply the filter backward - cut if off second to last (in other words the first point)
- going each way -- you get to filter the edges from both directions at least once.
- filtering using filtfilt makes the most of the data at the edges

Notebook #2 – Filtering Synthetic Data
ATOC5860_applicationlab5_synthetic_data_with_filters.ipynb

LEARNING GOALS:
1) Apply both non-recursive and recursive filters to a synthetic dataset
2) Contrast the influence of applying different non-recursive filters including the 1-2-1 filter, 1-1-1 filter, the 1-1-1-1-1 filter, and the Lanczos filter.
3) Investigate the influence of changing the window and cutoff on Lanczos smoothing.

DATA and UNDERLYING SCIENCE:
In this notebook, you analyze a timeseries with known properties. You will apply filters of different types and assess their influence on the resulting filtered dataset.

Questions to guide your analysis of Notebook #2:

1) Create a red noise timeseries with oscillations. Plot your synthetic data – Look at your data!! Look at the underlying equation. What type of frequencies might you expect to be able to remove with filtering?

Below is a plot of our synthetic data, which is a red noise time series with oscillations. The underlying equation uses specifications of the number of time steps (200), the autocorrelation (0.5), and the frequency of oscillations (one at 52/256 and one at 100/256) to create a red noise time series. The frequencies we might expect to be able to remove with filtering are the frequencies that are specified (0.2 and 0.39 per time step).

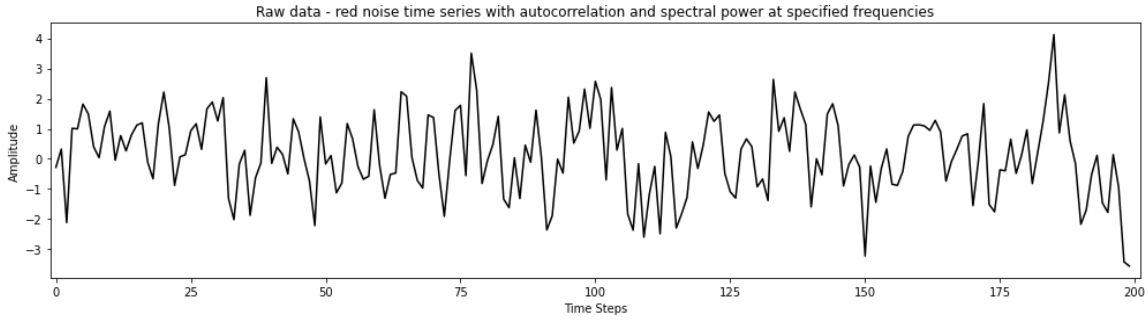

Figure 1: Raw data time series of red noise with autocorrelation 0.5 and spectral power at frequencies of 0.2 and 0.39 (per time step) specified.

2) Apply non-recursive filters in the time domain (i.e., apply a moving average to the original data) to reduce power at high frequencies. Compare the filtered time series with

the original data (top plot). Look at the moving window weights (bottom plot). You are using the function "filtfilt" from scipy.signal, which applies both a forward and a backward running average. Try different filter types – What is the influence of the length of the smoothing window or weighted average that is applied (e.g., 1-1-1 filter vs. 1-1-1-1-1 filter)? What is the influence of the amplitude of the smoothing window or the weighted average that is applied (e.g., 1-1-1 filter vs. 1-2-1 filter)? Tinker with different filters and see what the impact is on the filtering that you obtain.

Below, in Figures 2-4, we show the results of applying a non-recursive filter with a moving average to the original time series with different length and/or amplitude of the smoothing windows and weighted averages. Comparing Figures 2 and 3 shows us the influence of the amplitude of the smoothing window or the weighted average that is applied (1-2-1 in Fig. 2 vs. 1-1-1 in Fig. 3). It is very difficult to discern much of a difference between the smoothed time series from each of these methods. The only difference I notice is a slight decrease in the amplitude of wiggles in the 1-1-1 filter versus the 1-2-1 filter, which indicates that the 1-1-1 filter results in a greater extent of smoothing. This shows that using a greater amplitude of the central value of the smoothing window, which means this average is weighted heavier, results in a more smoothed time series.

Next, comparing Figures 3 and 4 shows us the influence of the length of the smoothing window or weighted average that is applied (1-1-1 in Fig. 3 vs. 1-1-1-1-1 in Fig. 4). It is clear that the filtered time series using the 1-1-1-1-1 filter is much more smoothed than that using the 1-1-1 filter. This shows that using a greater length of a moving average window, with less weight given to each average, results in a more smoothed time series.

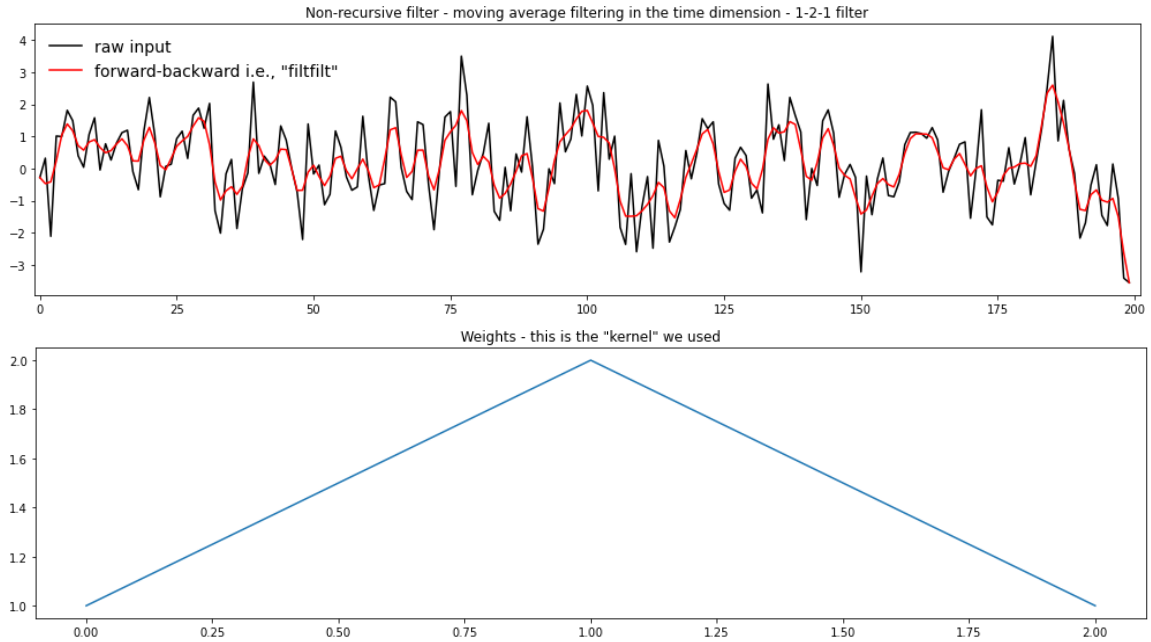

Figure 2: (top) Original time series (black) and filtered time series using a non-recursive filter with moving average 1-2-1 filter (red). (bottom) The associated moving window weights.

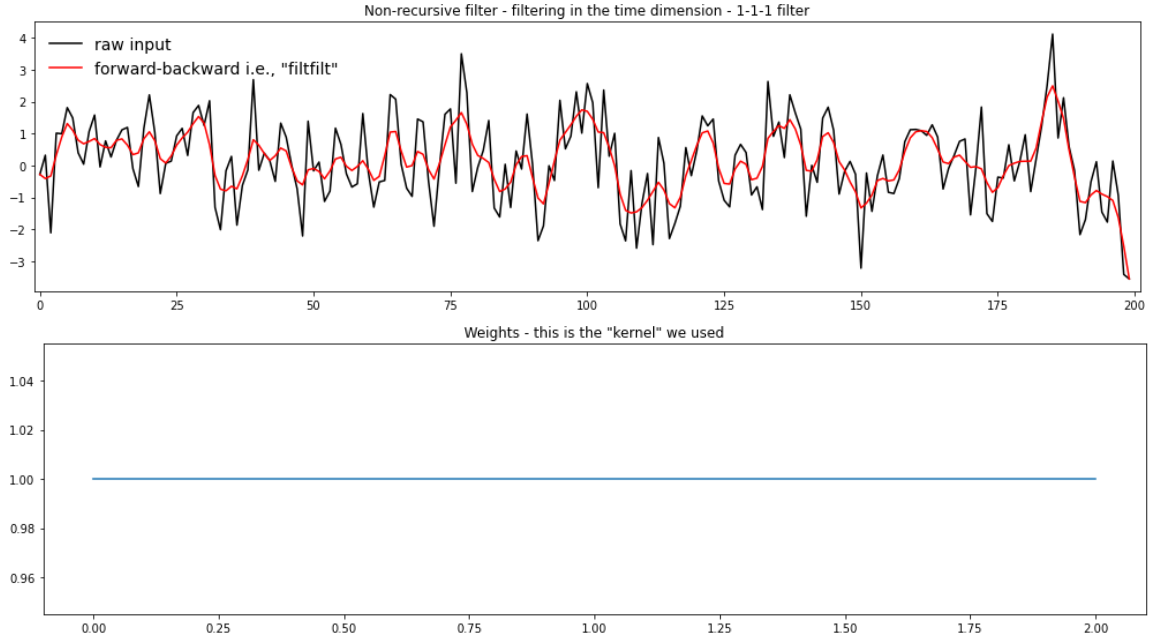

Figure 3: (top) Original time series (black) and filtered time series using a non-recursive filter with moving average 1-1-1 filter (red). (bottom) The associated moving window weights.

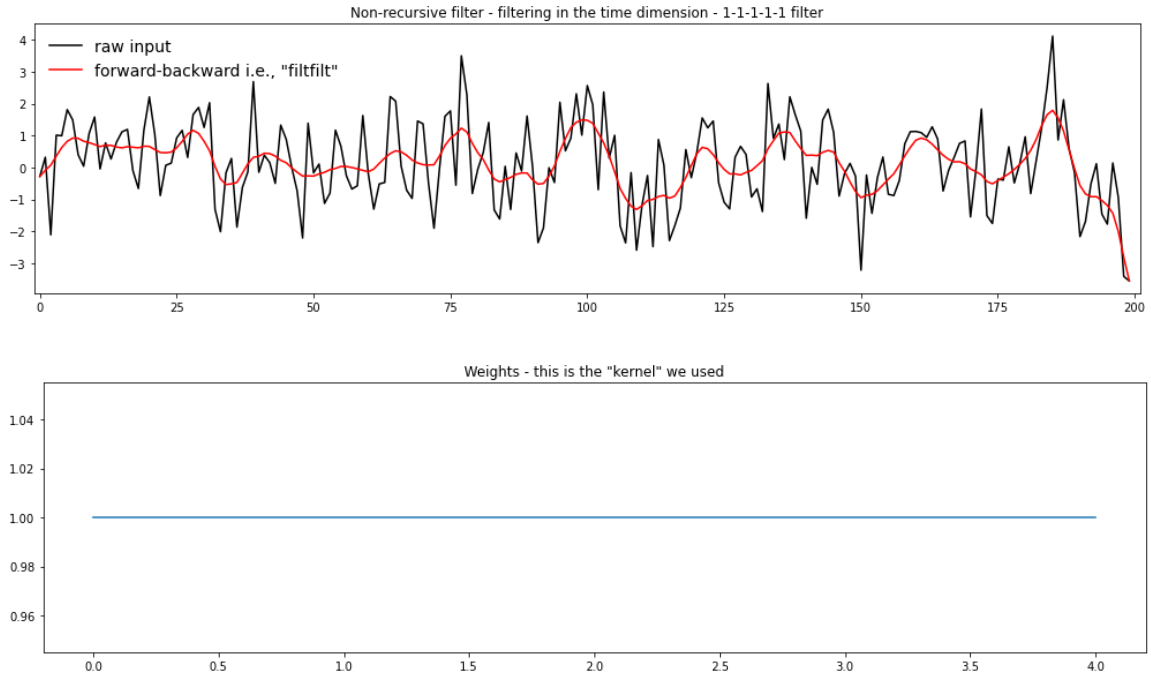

Figure 4: (top) Original time series (black) and filtered time series using a non-recursive filter with moving average 1-1-1-1-1 filter (red). (bottom) The associated moving window weights.

Gina Jozef
4/5/22

3) Apply a Lanczos filter to remove high frequency noise (i.e., to smooth the data). What is the influence of increasing/decreasing the window length on the smoothing and the response function (Moving Window Weights) in the Lanczos filter? What is the influence of increasing/decreasing the cutoff on the smoothing and the response function?

Figure 5 below shows the results of applying a Lanczos filter with intermediate window length (25) and cutoff (1/11). We can compare this to Figure 6 which shows the results of a Lanczos filter with the same cutoff, but smaller window length (5). The result of the shorter window length is that the time series is less smoothed. We can then compare this to Figure 7 which shows the results of a Lanczos filter with the same window length as Figure 6 (5) but decreased cutoff (11/100). This results in more weight concentrated at the edges of the window, which leads to more smoothing. Lastly, Figure 8 shows the results of increasing the window length (to 50) and decreasing the cutoff (to 25/100) which demonstrates the results of both actions. This filtered time series looks similar to the original time series, with some smoothing between fluctuations. Overall, this may not be an ideal combination of parameters for filtering a time series.

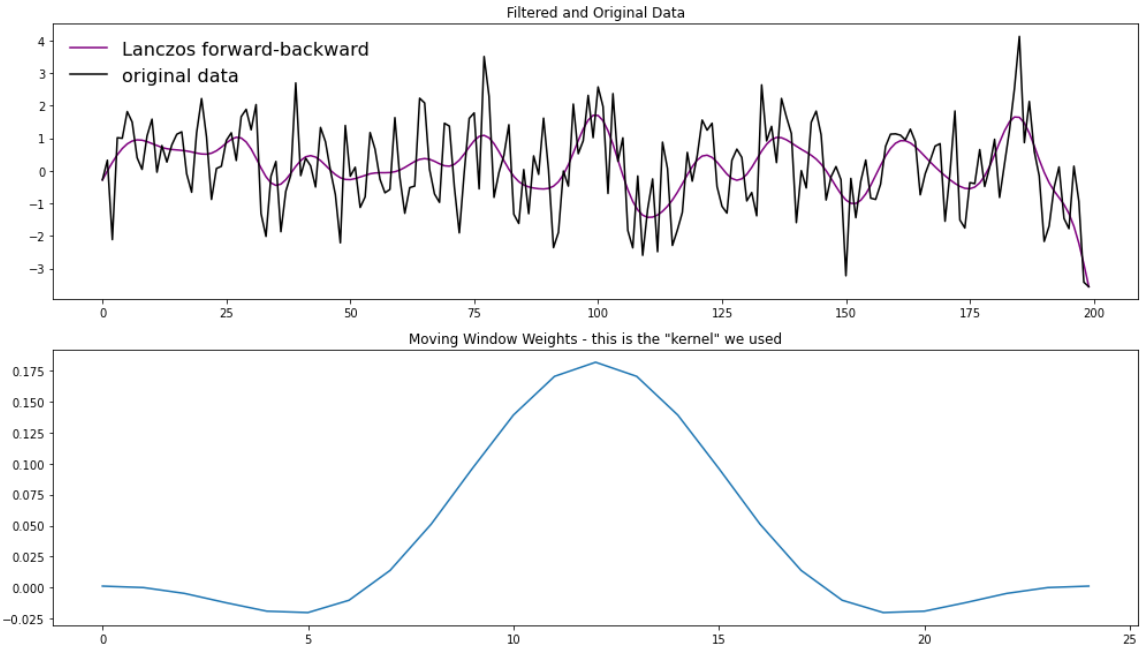

Figure 5: (top) Original time series (black) and filtered time series using a Lanczos filter with window length of 25 and cutoff of 1/11 (purple). (bottom) The associated moving window weights.

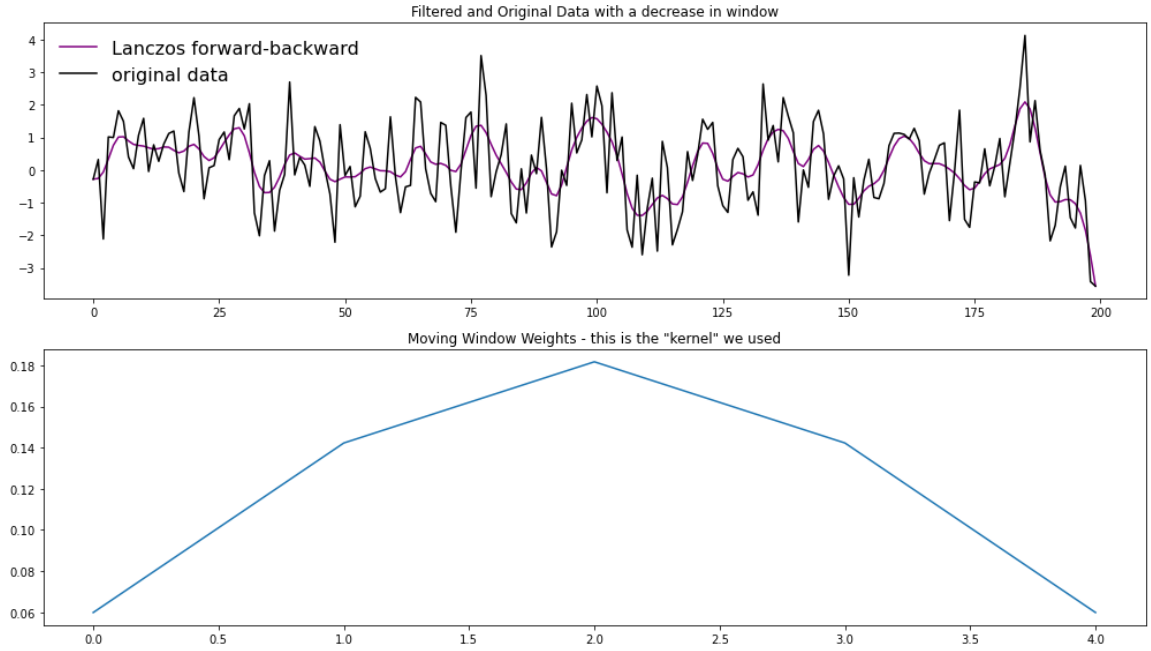

Figure 6: (top) Original time series (black) and filtered time series using a Lanczos filter with window length of 5 and cutoff of 1/11 (purple). (bottom) The associated moving window weights.

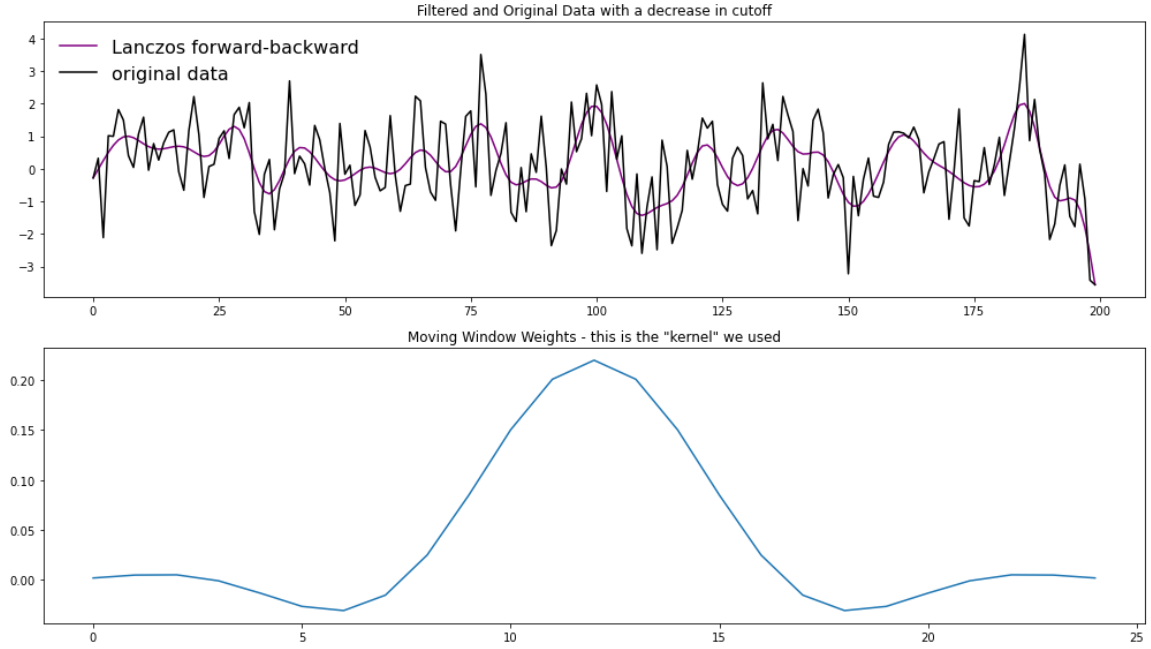

Figure 7: (top) Original time series (black) and filtered time series using a Lanczos filter with window length of 5 and cutoff of 11/100 (purple). (bottom) The associated moving window weights.

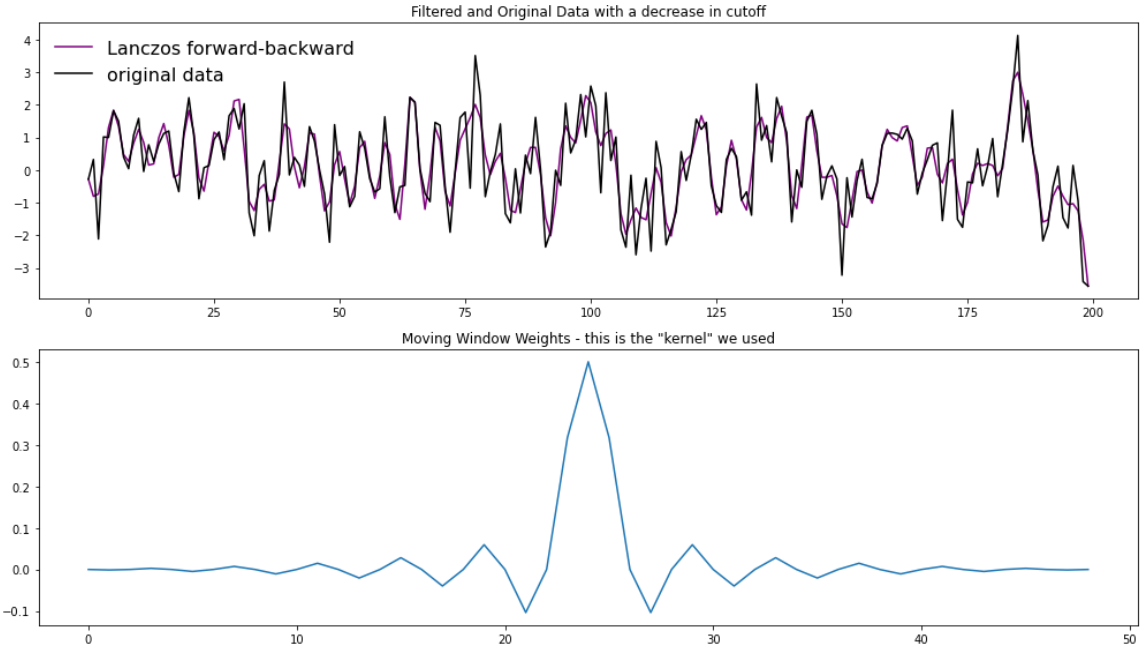

Figure 8: (top) Original time series (black) and filtered time series using a Lanczos filter with window length of 50 and cutoff of 25/100 (purple). (bottom) The associated moving window weights.

4) Apply a Butterworth filter, a recursive filter. Compare the response function (Moving Window Weights) with the non-recursive filters analyzed above.

Below, we apply a Butterworth filter to remove frequencies above 0.25, using a number of weights of 9. The Butterworth filter works by doing 2 passes (one forward and one backward). The results show that the high frequency fluctuations have been removed and the most weight is given to the center of the window. The time series results look most similar to those from the Lanczos filter using a window length of 5 and cutoff of 11/100. The response function (moving window weights) looks most similar to that from the Lanczos filter using window length of 5 and cutoff of 1/11. The main difference between the response function of this recursive filter, compared to those from the non-recursive filters above is that the response function for the Butterworth filter shows the most weight given to the center of the window, rather than the edges of the window, and is also spread out more evenly across frequencies.

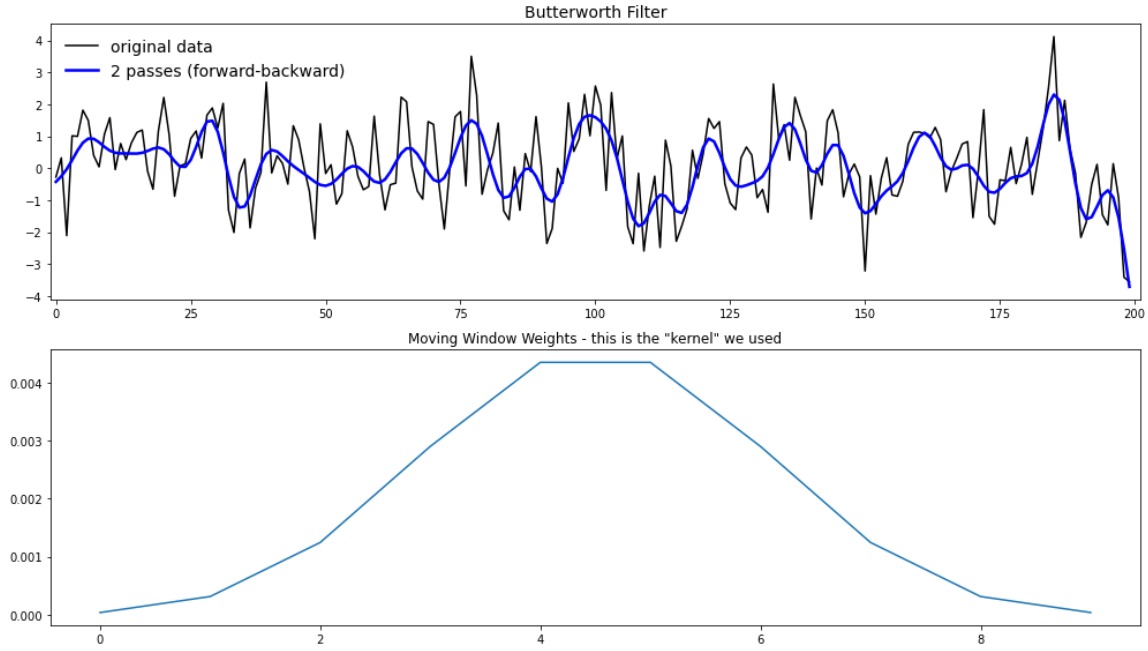

Figure 9: (top) Original time series (black) and filtered time series using a Butterworth with N=9, removing frequencies above 0.25 (blue). (bottom) The associated moving window weights.

Notebook #3 – Filtering ENSO data
ATOC5860_applicationlab5_mrbutterworth_example.ipynb

LEARNING GOALS:
1) Assess the influence of filtering on data in both the time domain (i.e., in time series plots) and the spectral domain (i.e., in plots of the power spectra).
2) Apply a Butterworth filter to remove power of specific frequencies from a time series.
3) Contrast the influence of differing window weights on the filtered dataset both in the time domain and the spectral domain.
4) Calculate the response function using the Convolution Theorem.
5) Assess why the python function filtfilt is filtering twice.

DATA and UNDERLYING SCIENCE:
In this notebook, you analyze monthly sea surface temperature anomalies in the Nino3.4 region from the Community Earth System (CESM) Large Ensemble project fully coupled 1850 control run (http://www.cesm.ucar.edu/projects/community-projects/LENS/). A reminder that an pre-industrial control run has perpetual 1850 conditions (i.e., they have constant 1850 climate). The file containing the data is in netcdf4 format: CESM1_LENS_Coupled_Control.cvdp_data.401-2200.nc

Does this all look and sound really familiar? It should!! This dataset is the same one you analyzed in Homework #4.

Questions to guide your analysis of Notebook #3:

Gina Jozef
4/5/22

1) Look at your data! Read in your data and Make a plot of your data. Make sure your data are anomalies (i.e., the mean has been removed). Look at your data. Do you see variance at frequencies that you might be able to remove?

In the below time series, we show the anomalies of SST in the Nino3.4 region from CESM.

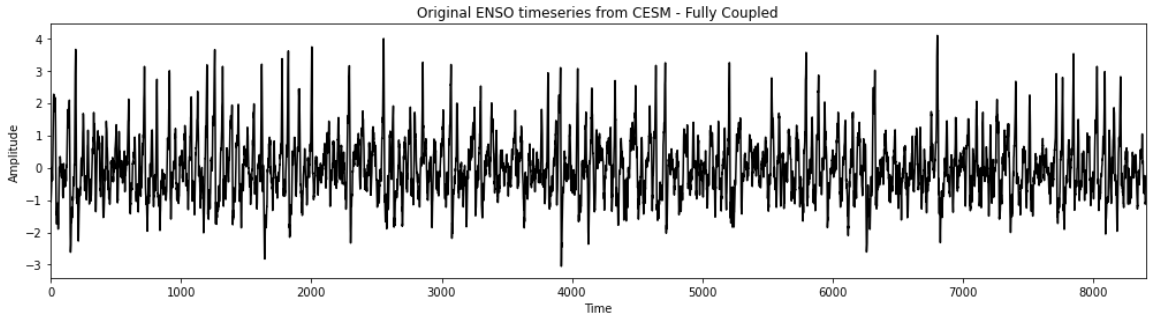

Figure 1: Time series of SST anomaly in the Nino3.4 region from CESM fully coupled 1850 control run.

2) Calculate the power spectrum of your original data. Calculate the power spectra of the Nino3.4 SST index (variable called "nino34") in the fully coupled model 1850 control run. Apply the analysis to the first 700 years of the run. Use Welch's method (WOSA!) with a Hanning window and a window length of 50 years. Make a plot of normalized spectral power vs. frequency. Where is their power that you might be able to remove with filtering?

In the below figure, we show the power spectrum of SST anomaly in the Nino3.4 region, using the first 700 years of data. There is some power at a frequency of ~0.018 and 0.068 that we should be able to remove with filtering.

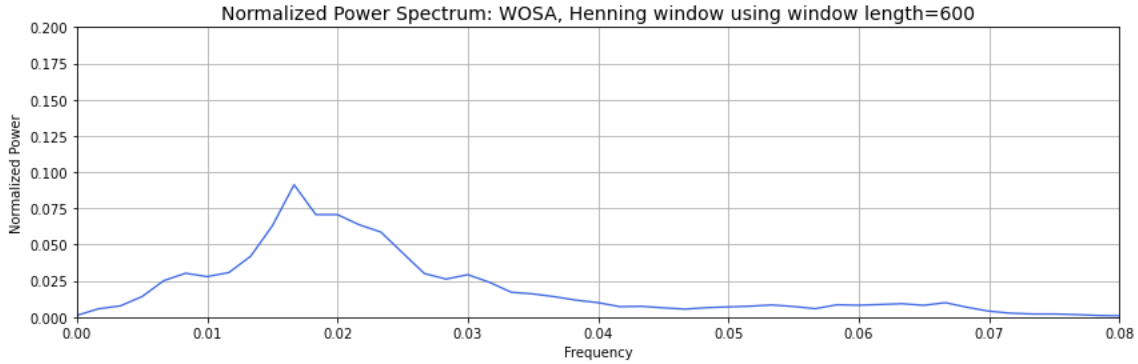

Figure 2: Power spectrum of of SST anomaly in the Nino3.4 region using WOSA with a Hanning window of length 600 months (50 years).

3) Apply a Butterworth Filter. Use a Butterworth filter to remove all spectral power at frequencies greater than 0.04 per month (i.e., less than 2 year). Use an order 1 Butterworth filter (N=1, 1 weight). Replot the original data and the filtered data. Calculate the power spectra of your filtered data. Assess the influence of your filtering in both in time domain (i.e., by comparing the original data time series and filtered time series data) and the

frequency domain (i.e., by comparing the power spectrum of the original data and the power spectrum of the filtered data). Look at the response function of the filter in spectral domain using the convolution theorem. Well that was pretty boring… we still have most of the power retained….

The below figures show that some of the power of low frequency fluctuations has been removed, but all power of high frequency (>0.04/month) fluctuations have been removed.

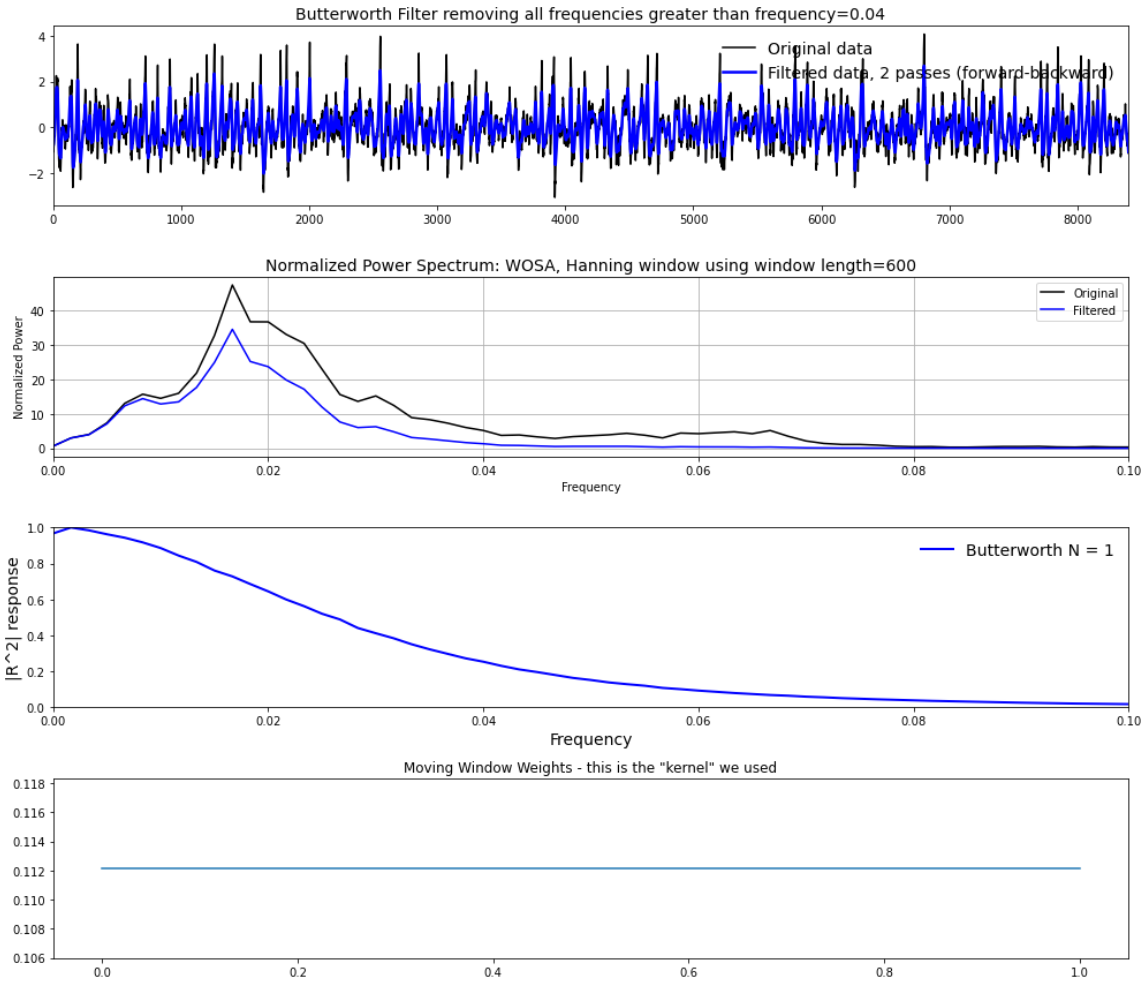

Figure 3: (First) time series of original data (black) and filtered data using Butterworth filter removing frequencies above 0.04 and N=1 (blue). (Second) power spectra of original data (black) and filtered data (blue). (Third) response function of the Butterworth filter. (Fourth) moving window weights for the Butterworth filter.

4) Let's apply another Butterworth Filter and this time really get rid of ENSO power!. Let's really have some fun with the Butterworth filter and have a big impact on our data... Let's remove ENSO variability from our original timeseries. Apply the Butterworth filter but this time change the frequency that you are cutting off to 0.01 per month (i.e., remove all power with timescales less than 8 years). Use an order 1 filter (N=1). Replot the original data and the filtered data. Calculate the power spectra of your filtered data. Assess the influence of your filtering in both in time domain (i.e., by comparing the original data time

series and filtered time series data) and the frequency domain (i.e., by comparing the power spectrum of the original data and the power spectrum of the filtered data). Look at the response function of the filter in spectral domain using the convolution theorem.

The below figures show that most of the power of all frequency fluctuations has been removed, since we set out to remove frequencies above 0.01/month. Since all of the significant power in the original power spectrum was for frequencies greater than this, then we don't see much power in the resulting time series at any frequency.

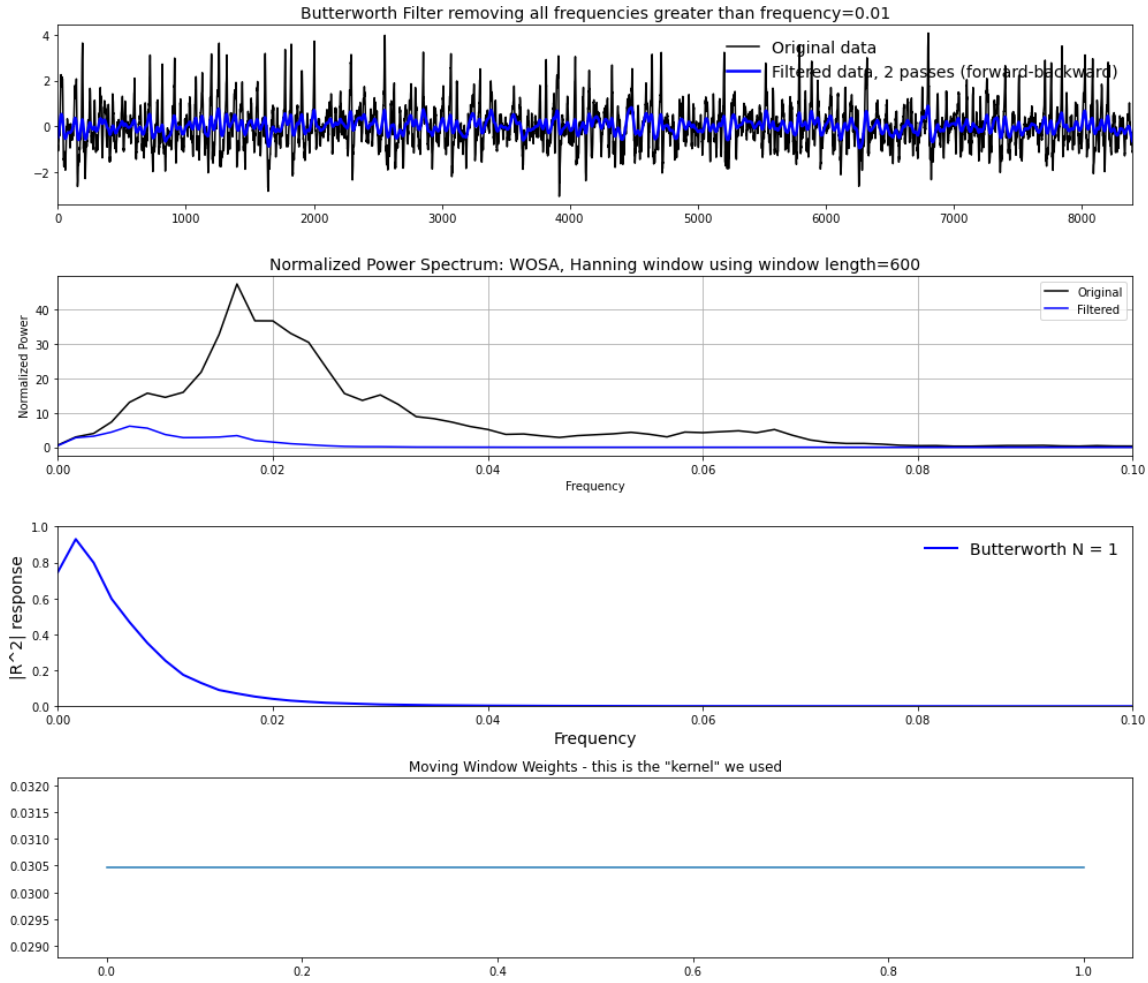

Figure 4: (First) time series of original data (black) and filtered data using Butterworth filter removing frequencies above 0.01 and N=1 (blue). (Second) power spectra of original data (black) and filtered data (blue). (Third) response function of the Butterworth filter. (Fourth) moving window weights for the Butterworth filter.

5) Let's apply yet another Butterworth Filter – and this time one with more weights. Repeat step 4) but this time change the order of the filter. In other words, increase the number of weights being used in the filter by increasing the parameter N in the jupyter notebook. What is the impact of increasing N on the filtered dataset, the power spectra, and the

moving window weights? You should see that as you increase N – a sharper cutoff in frequency space occurs in the power spectra. Why?

The below figures show that, again, most of the power of all frequency fluctuations has been removed, since we set out to remove frequencies above 0.01/month. However, there is a sharper cutoff in frequency space after 0.01/month than in the previous figure. This is because the response function decreases more drastically from its peak, but more weight is given within the center of the window, which results in a power spectrum which has more power before the cutoff frequency is reached, but a sharper decrease in power at the cutoff frequency.

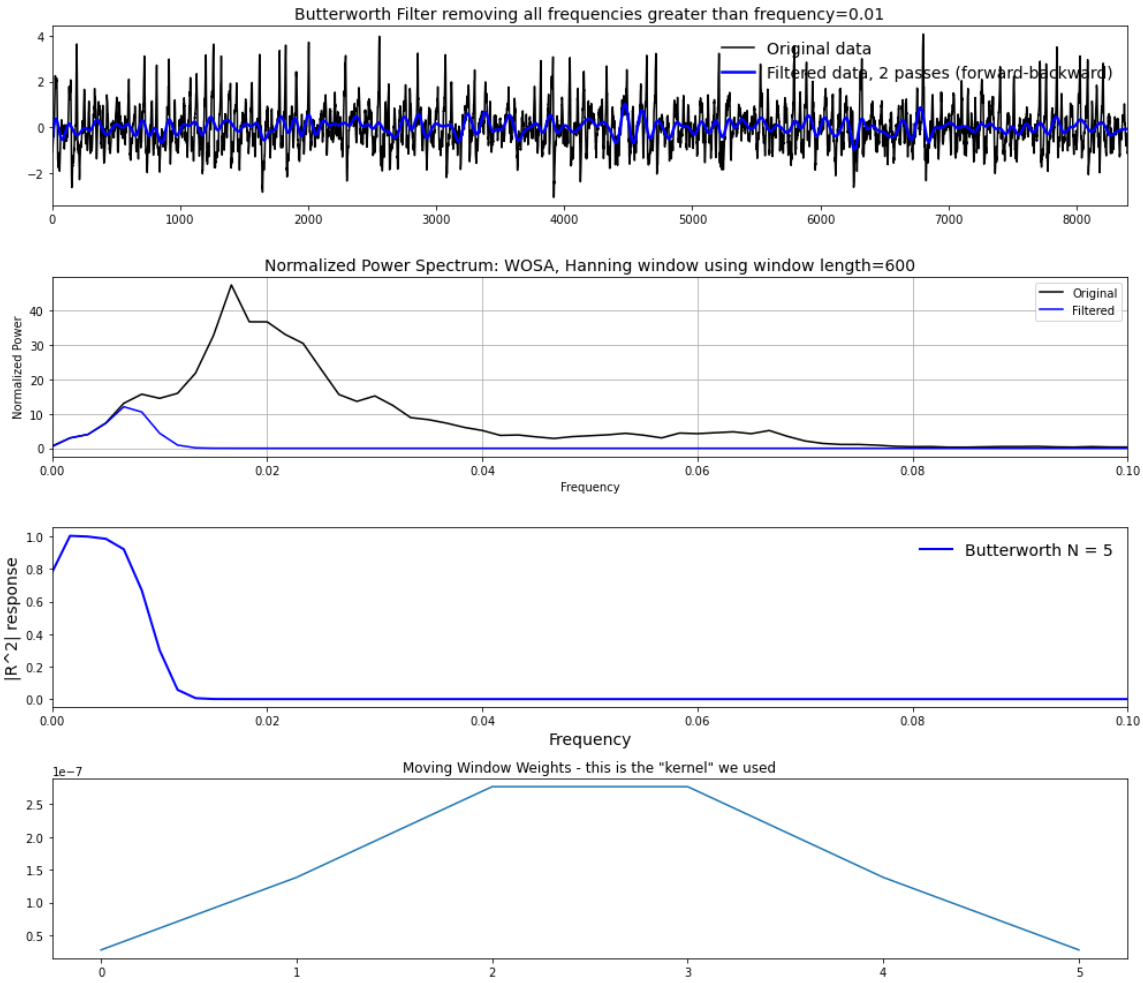

Figure 5: (First) time series of original data (black) and filtered data using Butterworth filter removing frequencies above 0.01 and N=5 (blue). (Second) power spectra of original data (black) and filtered data (blue). (Third) response function of the Butterworth filter. (Fourth) moving window weights for the Butterworth filter.

Lastly, below we show the results of if we apply a cutoff frequency of 0.08 and N=26. The results of the filtered data look very similar to the results of the original data. This is because most of the power in the original data is for frequencies below 0.08/month, and there is not much beyond that, so removing higher frequency data than 0.08/month does

not do much. Setting N=26 makes the response function look kind of silly, in that is hovers around 1 for most frequencies but then increases above 1 just before the cutoff frequency, and then decreases to zero. This amplifies the power at frequencies just below 0.08/month, which is not ideal since it introduces fake power.

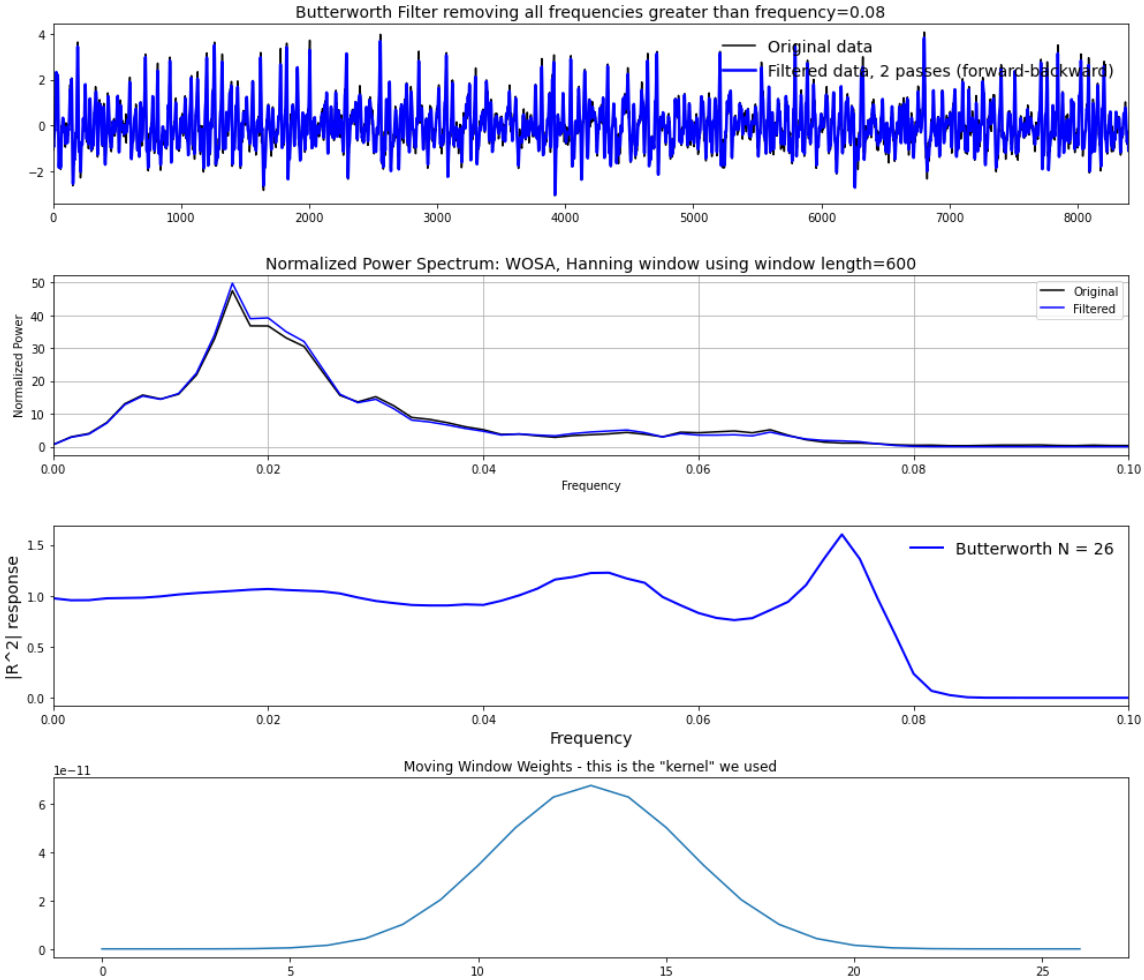

Figure 6: (First) time series of original data (black) and filtered data using Butterworth filter removing frequencies above 0.08 and N=26 (blue). (Second) power spectra of original data (black) and filtered data (blue). (Third) response function of the Butterworth filter. (Fourth) moving window weights for the Butterworth filter.

6) Assess what is "under the hood" of the python function. How are the edge effects treated? Why is the function filtfilt filtering twice?

filtfilt helps to remove the edge effects by filtering twice. This way, the beginning and end of the time series are not missed. In each individual run, either the beginning or end of the series would be removed based on how the filtering is conducted, but by filtering twice, once in each direction, then the information in both the beginning and end of the time series is retained.