Michelle Maclennan
11 November 2020

ATOC7500 – Application Lab #5
Filtering Timeseries
in class Wednesday November 11 and Monday November 16

Notebook #1 – ATOC7500_applicationlab5
ATOC7500_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.

Different smoothing functions handle the endpoints differently, and some (like the 'same' and 'full' modes in np.convolve) actually add smoothed points at the end to keep the length of the original array. The important this to know are if the mode changes the length of your timeseries, and if your timeseries is shifted or offset in the process.

Notebook #2 – Filtering Synthetic Data
ATOC7500_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:

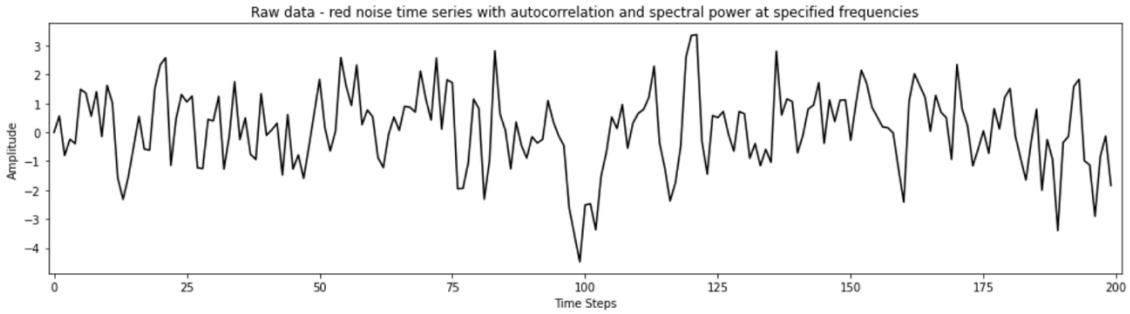
1) Create a red noise timeseries with oscillations. Plot your synthetic data – Look at your data!! Look at the underlying equation. What type of frequencies might you expect to be able to remove with filtering?

The synthetic data are generated using the sum of two cosine waves multiplied by two oscillation frequencies, 52/256 and 100/256. With filtering, you can remove data at high or low filtering, or both. In this case, the red noise dataset seems to exhibit a lot of variability in the high frequency range, so we could use filtering to remove these high oscillation frequencies.

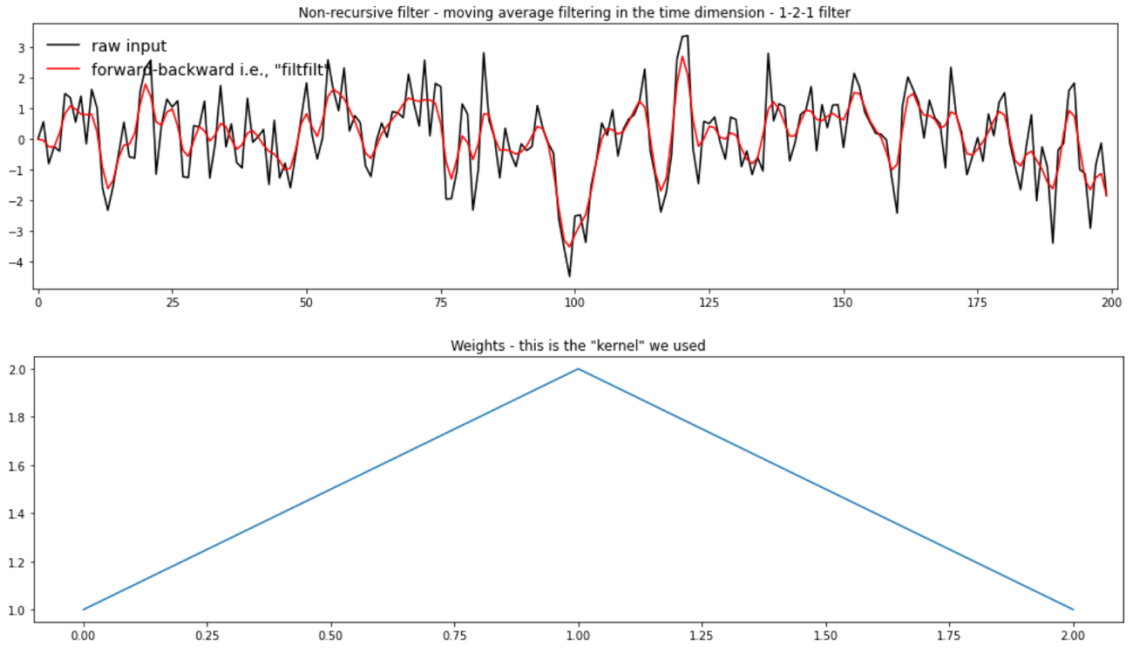
2) Apply non-recursive filters in the time domain (i.e., apply a moving average to the original data) to reduce power at high frequencies. Compare the filtered time series with the original data (top plot). Look at the moving window weights (bottom plot). You are using the function "filtfilt" from scipy.signal, 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.

The moving average filter strongly reduces power at high frequencies because it takes the mean of a certain number of points to the right and left of each data point to generate a smoothed curve.

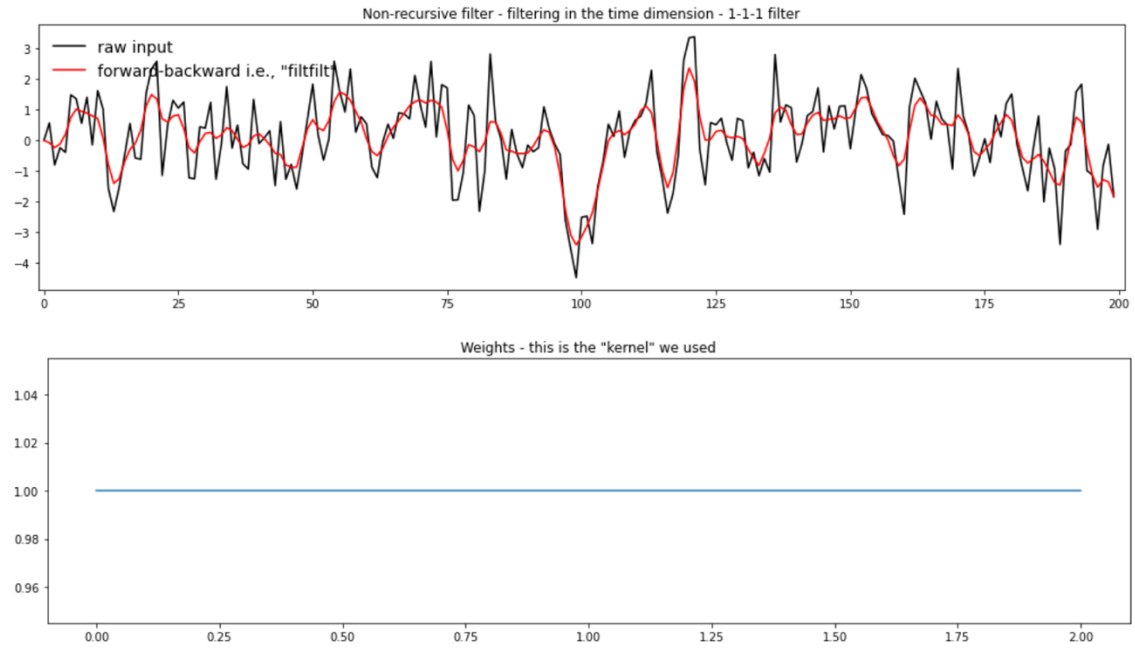
Here is the original plot:



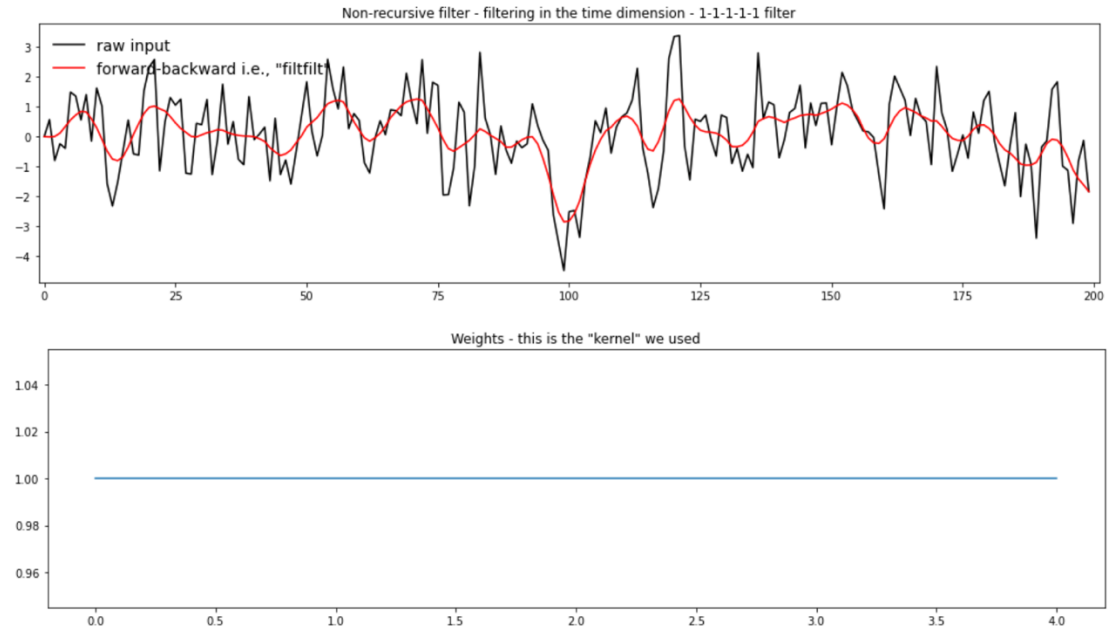
When we use a 1-2-1 filter, it smoothens out the curve slightly, but still applies the most weight to the middle data point:



A 1-1-1 filter does an even better job smoothing out the data because it applies equal weight to all three points:

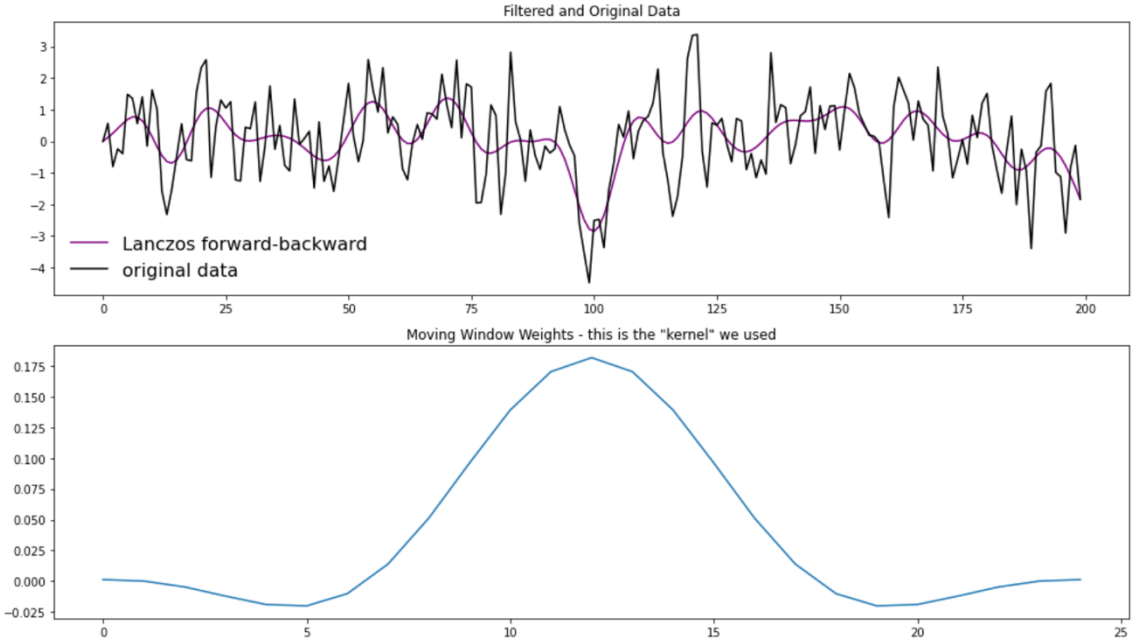


The more weights (forward and backward) applied in addition to the data point in question, the more the power at high frequencies is reduced:

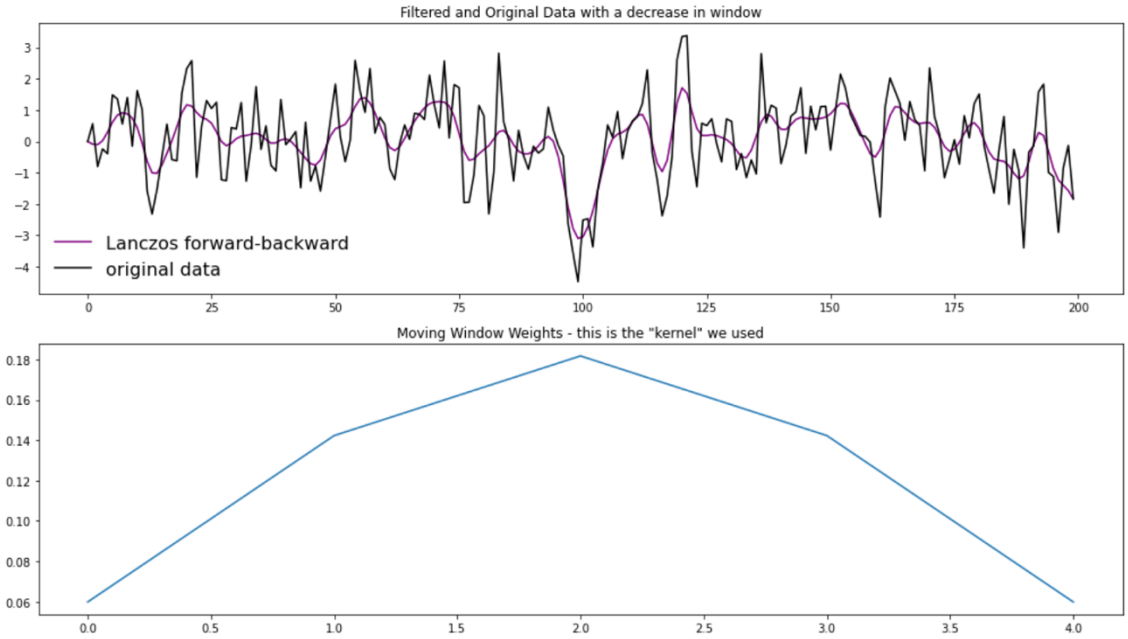


3) Apply a Lanczos filter to remove high frequency noise (i.e., to smooth the data). What is the influence of increasing/decreasing the window length on the smoothing and the response function (Moving Window Weights) in the Lanczos filter? What is the influence of increasing/decreasing the cutoff on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

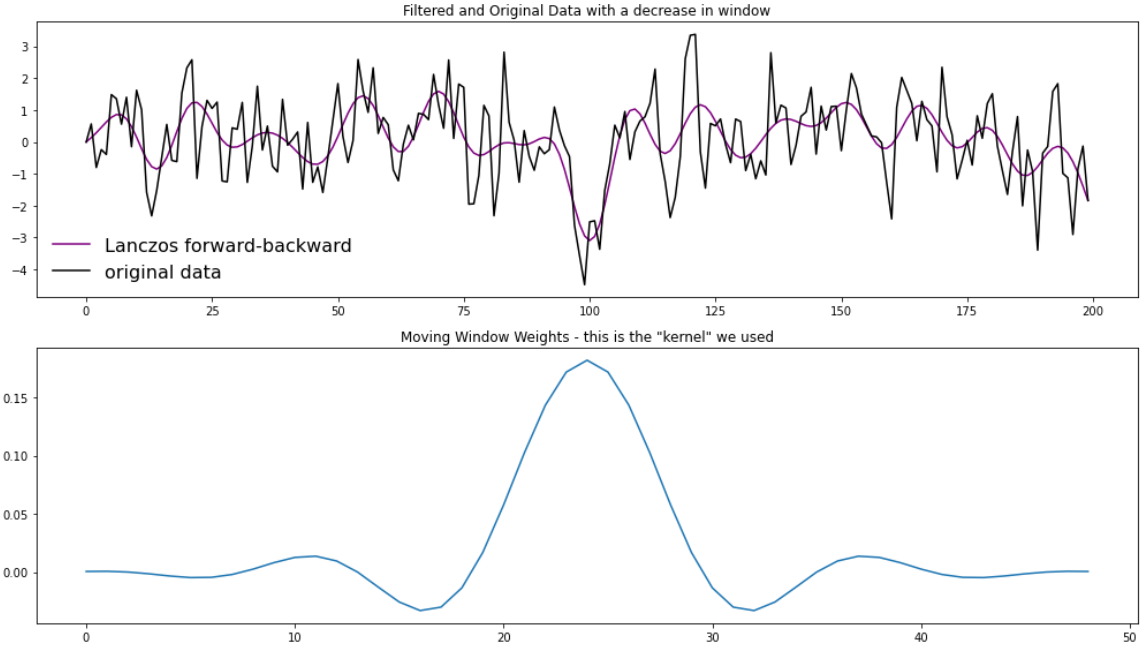
Here is the plot of the smoothed data using the Lanczos filter with a window length of 25 (the original length):



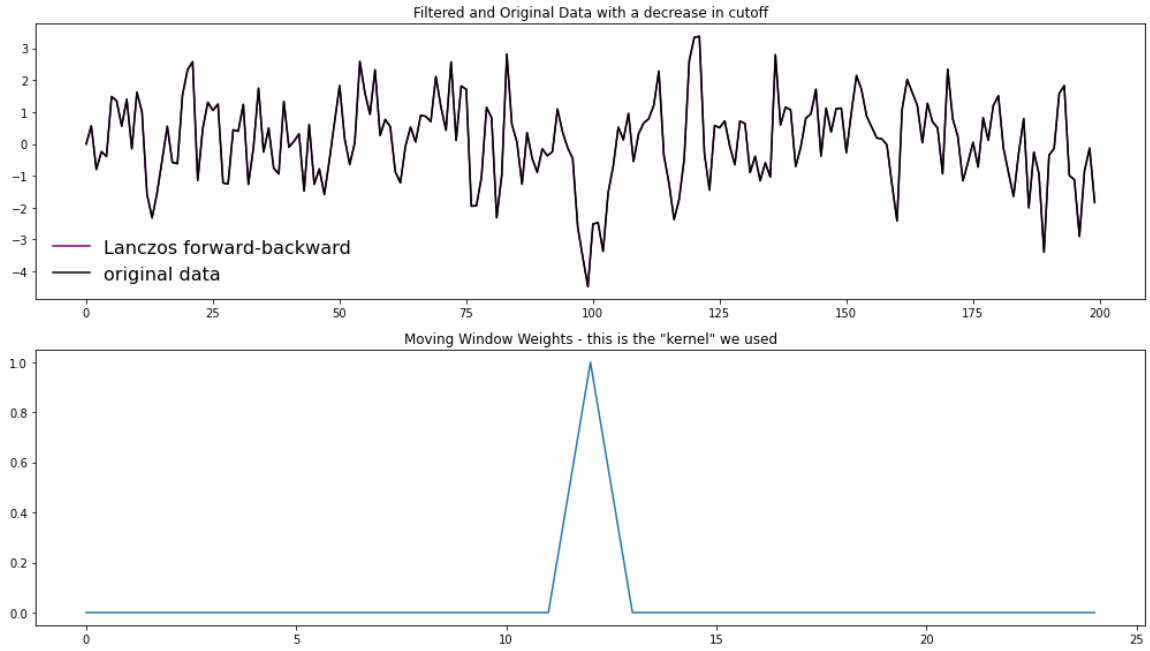
When you decrease the window length to 5, the “kernel” or window weights operate on a shorter spectrum (only 0-4 instead of 0-24 moving window weights), there’s no longer a nice Gaussian curve of window weights, and you get less smoothing:



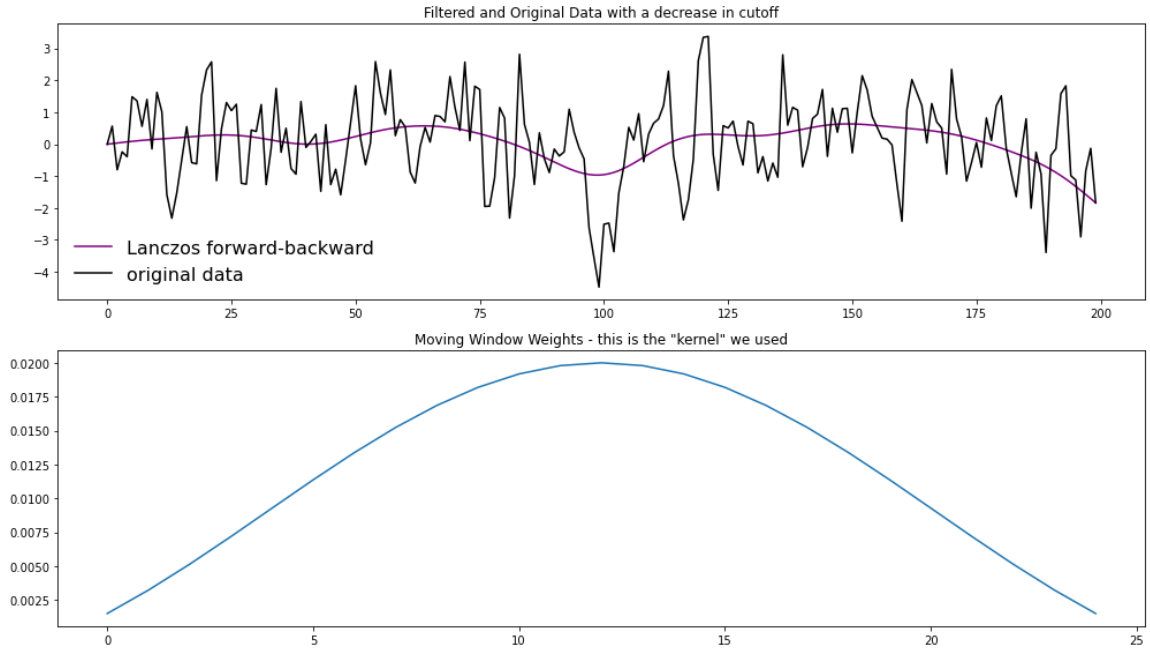
If you increase the window length to 50, you get a smoother, more Gaussian curve for the moving window weights and a correspondingly smoother plot. However, the resulting smoothed data don’t look all that different from the ones with a window length of 25, so it seems like there’s maybe a threshold where window weights have the most impact on smoothing the dataset.



If you increase the cutoff to ½ (originally the cutoff was 1/11, and I'm using the standard window length 25 here), the weights are concentrated at the center of the window. This means the final time series looks a lot like the initial one since very few high frequencies are cut out:



If you decrease the cutoff to 1/100, then the weights are much more spread out on the ends and you end up with a very smooth time series. This cuts off all of the high frequency variability in the dataset. The lower the cutoff frequency, the higher the influence of the window length on smoothing.



Note that the sharper you want the cutoff to be, the more weights you need.

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

Unlike the Lanczos filter above and the other weighted average filters above, the Butterworth filter can be applied to a dataset in real time since it is recursive. However, it actually yields really similar results to the smoothing of the time series, and the response function looks really similar to some of the Gaussian-like curves earlier on. The high the number of weights (N), the smoother the curve of the response function, and a much greater weight is associated with the curve. There's almost an exponential relationship between N and the weight associated with the middle value of the curve. Additionally, the higher N is, the longer it takes for the signal of a datapoint to die out, meaning there is more smoothing taking place at all frequencies. The frequency cutoff also impacts smoothing, because the lower the cutoff, the more frequencies are removed from the time series.

Notebook #3 – Filtering ENSO data
ATOC7500_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.

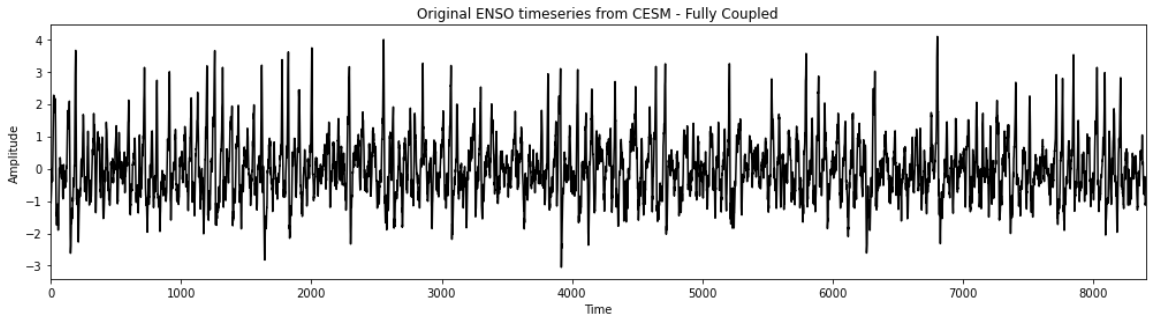
DATA and UNDERLYING SCIENCE:
In this notebook, you analyze monthly sea surface temperature anomalies in the Nino3.4 region from the Community Earth System (CESM) Large Ensemble project fully coupled 1850 control run (http://www.cesm.ucar.edu/projects/community-projects/LENS/). A reminder that an pre-industrial 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.

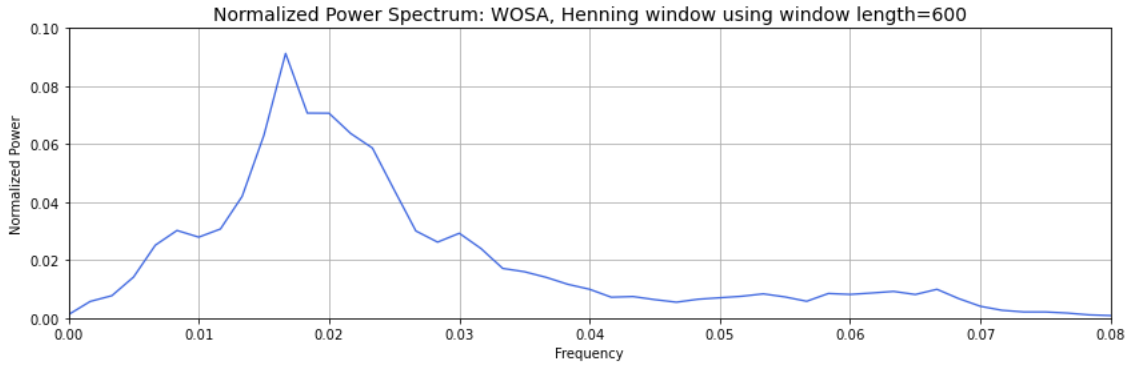
Questions to guide your analysis of Notebook #3:

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?

The data are centered around zero and represent the monthly Nino 3.4 SST anomaly over a period of 700 years in the 1850s climate. There is a lot of variance at high frequencies (months-years) and it looks like there's also some variance with a longer time period (time scale of 60-80 years).



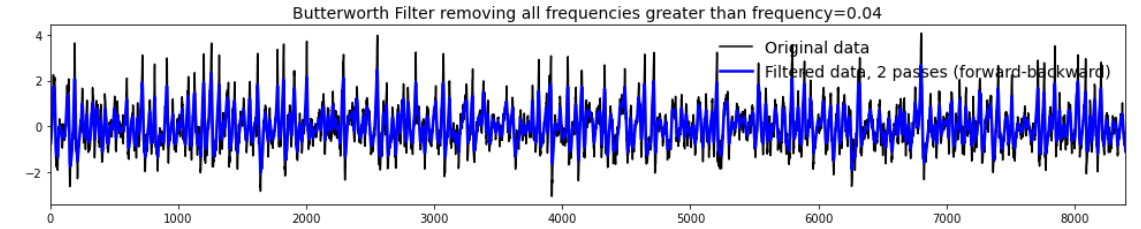
2) Calculate the power spectrum of your original data. Calculate the power spectra of the Nino3.4 SST index (variable called "nino34") in the fully coupled model 1850 control run. Apply the analysis to the first 700 years of the run. Use Welch's method (WOSA!) with a Hanning window and a window length 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?

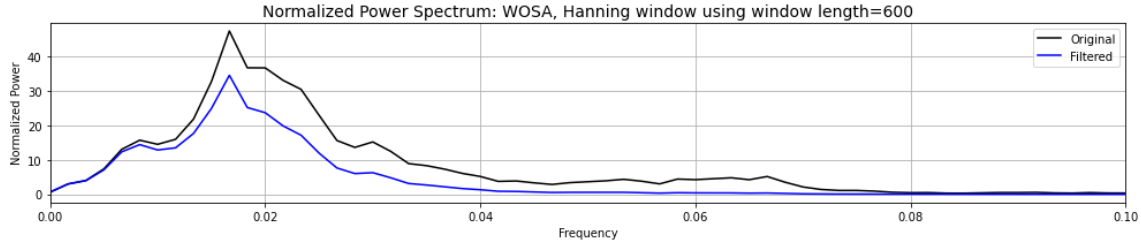


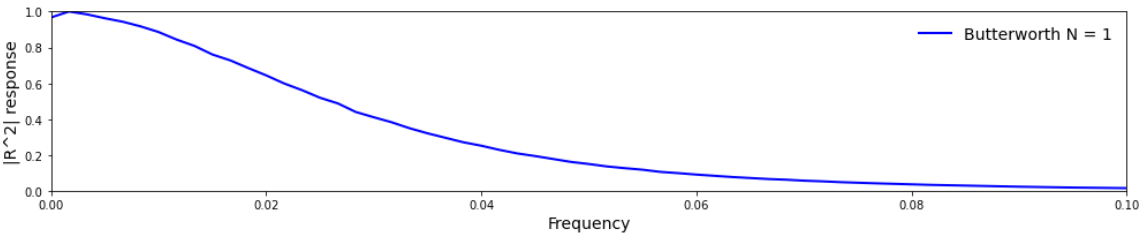
The primary peak in power occurs around a frequency of 0.017, or a period of 60 months (5 years). This is something we would expect, given that we know ENSO occurs on a 5-7 year timescale. However, there is also a peak at 0.008 (10 years) and multiple peaks in the high frequency range. Depending on what information we want to look at (short-term variability, long-term variability), we can decide what frequencies to cut out. Using the Butterworth filter, you can set a maximum frequency, and all frequencies above that one will be cut off.

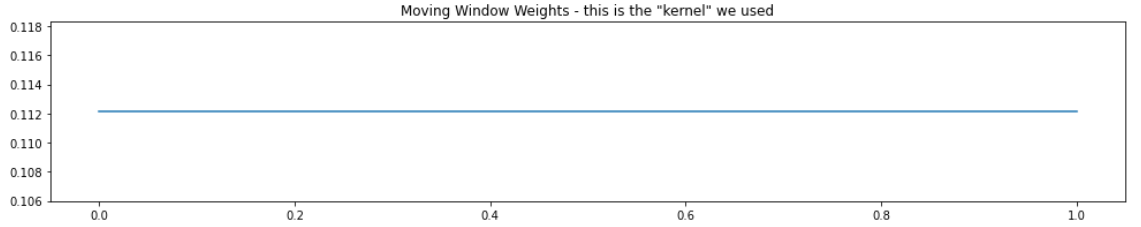
3) Apply a Butterworth Filter. Use a Butterworth filter to remove all spectral power at frequencies greater than 0.04 per month (i.e., less than 2 year). Use an order 1 Butterworth filter (N=1, 1 weight). Replot the original data and the filtered data. Calculate the power spectra of your filtered data.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

As was already mentioned in the question, removing frequencies greater than 0.04 per month (less than two years) didn't really change the time series or the power spectrum very much. That's because it only got rid of very high frequencies or variability on timescales less than 2 years – which is considerably shorter than the usual period of variability we see around ENSO. The response function has a very smooth and shallow curve, which is another indicator that the impact of the frequency cutoff was weak on the smoothed time series.



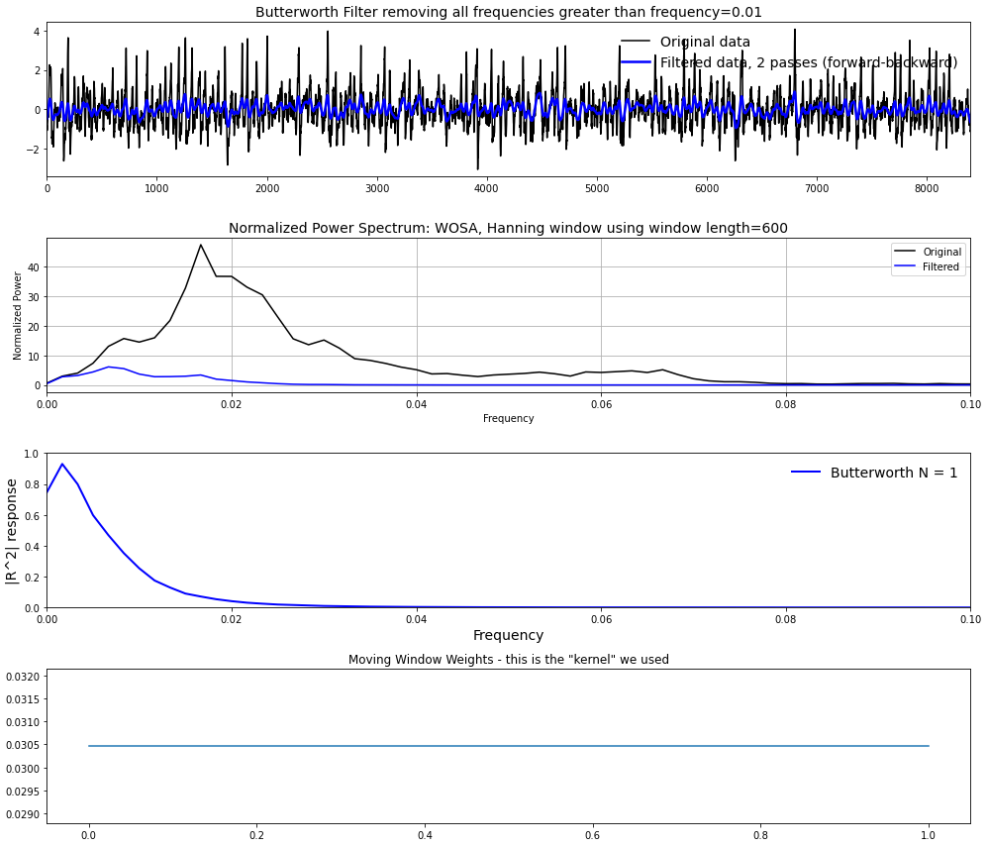






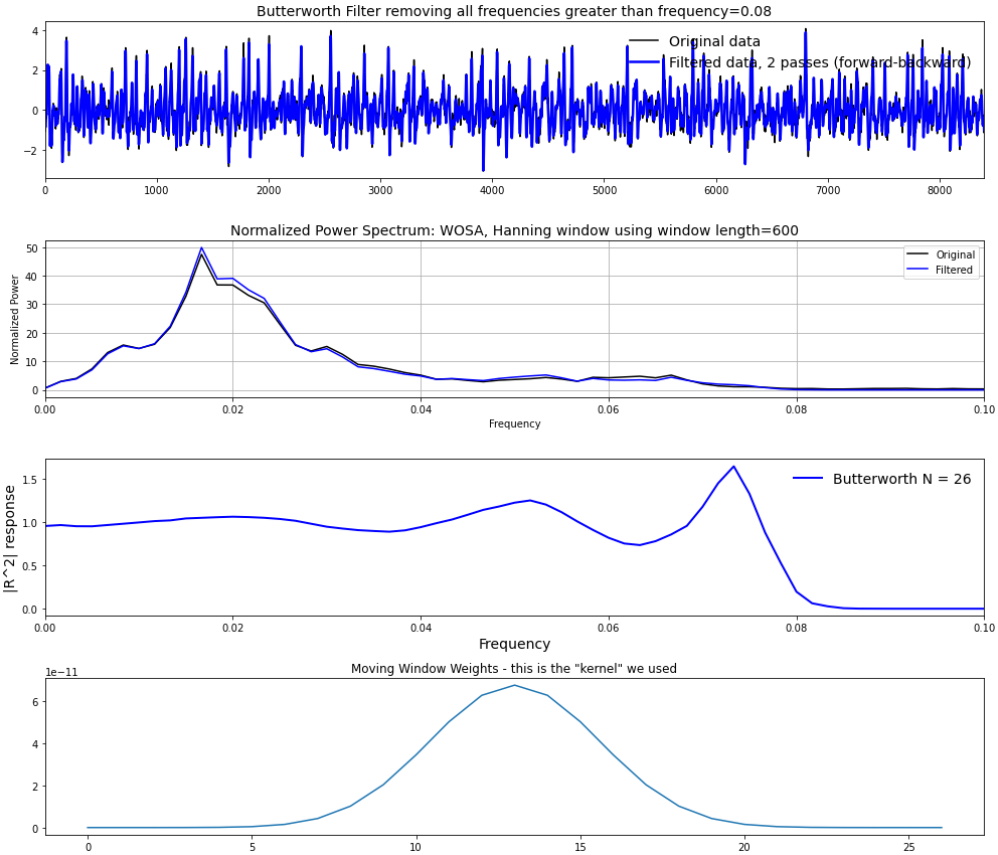
4) Let's apply another Butterworth Filter and this time really get rid of ENSO power!. Let's really have some fun with the Butterworth filter and have a big impact on our data... Let's remove ENSO variability from our original timeseries. Apply the Butterworth filter but this time change the frequency 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.

Cutting off frequencies greater than 0.01 per month (time scales of 8 years or shorter) essentially removes ENSO variability from the time series. Not only does this cut out the main power we saw earlier in the power spectrum, but it also cuts out all over high frequency variability, leaving only decadal variability and longer. It is clear from the smoothed time series that these high frequencies have been filtered out and the dataset is smoothed considerably. Using an order 1 (N = 1) filter means that there moving window weights are constant across values and the response function decreases much more quickly with frequency than for the last question. In the power spectrum, the ENSO peak has been removed.



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?

Adding more weights (N = 26) to the filter causes the moving window weights to change shape – so that they now resemble more of a Gaussian curve where the most weight is given to center values. The response function no longer resembles a smooth curve with low frequencies weighted the most but instead oscillates and has a peak around the frequency of 0.073 or 13 months. Essentially, as N increases from 1 to 26, the frequency cutoff slowly shifts toward higher frequencies again in the response function. Around N = 24, the response function begins to oscillate and becomes unstable by N = 26. This is a known issue with the Butterworth filter, because it relies on only previous data and requires fewer weights than a recursive filter. The result of this high N combined with a frequency cutoff of 0.08 is that high-frequency variability is not smoothed out, and the filtered time series resembles the initial time series very closely in both the smoothed time series and the power spectrum.



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

The scipy signal Butterworth function takes in the order and the frequency cutoff and returns the filter coefficients associated with these values (b is the numerator, also plotted as the Moving Window Weights, and a is the denominator). The filtfilt function then takes these coefficients and filters the time series forwards and backwards, which removes the phase shift associated with filtering in a single direction. Filtering twice also increases the filter order to twice that of the original. This function also handles the edges of the signal using padding or Gustafsson’s method to avoid the Gibbs phenomenon. This function then returns the filtered time series.