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.

Convolution, the mode you pick pads things differently and then you loose points or keep them.

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 biggest takeaway is to Know if your timeseries is changing in length, or if you are offsetting it.

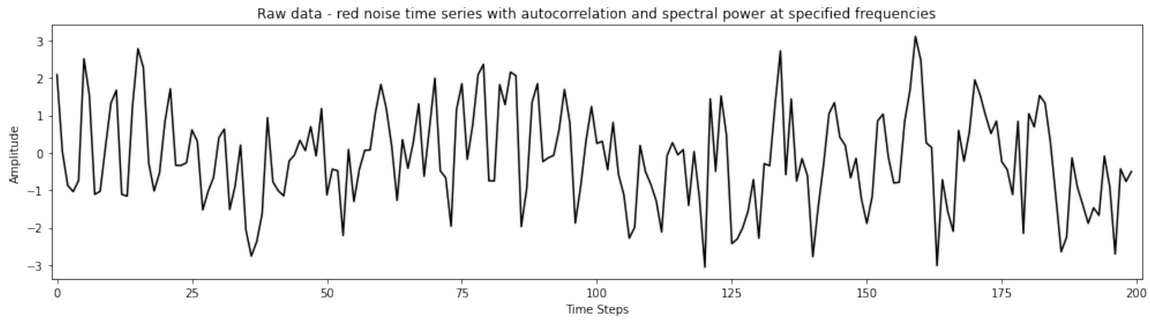
Notebook #2 – Filtering Synthetic Data
ATOC7500_applicationlab5_synthetic_data_with_filters.ipynb

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.

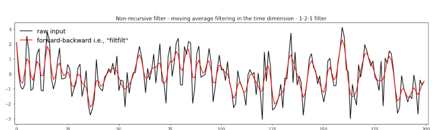
Questions to guide your analysis of Notebook #2:

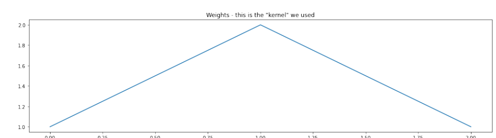
1) Create a red noise timeseries with oscillations. Plot your synthetic data – Look at your data!! Look at the underlying equation. What type of frequencies might you expect to be able to remove with filtering?

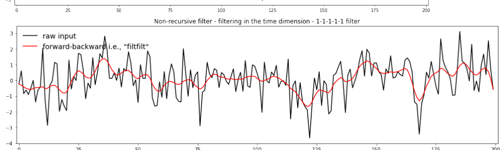


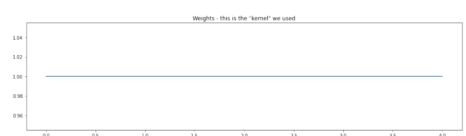
I would expect to remove the spikey bits in order to be able to see the oscillation. We can either remove the low or high frequency.

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.









Moving window filters are mostly high frequency filters
The 1-1-1 and longer 1-1-1-1-1 smooth out the high frequencies

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?
If the window is maxed out, (66) then the kernel is wavey and the signal is very smoothed, the shorter the window, the less smooth the filtered data. Same with the cutoff – increasing the cutoff (2/11) compresses the weights and matches the raw data more.

decreasing the cutoff (2/11) compresses the weights and matches the raw data more. The higher the cut-off the higher the drop off. The sharper the drop, the smaller the window, so when you have a large cutoff the filter time series looks the same as the OG.

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

This filter has a window of 9 and outside the window it's weighted zero, unlike the Lanczos which weights a wider window but a narrowed amount within it.

<u>Notebook #3 – Filtering ENSO data</u>
ATOC7500_applicationlab5_mrbutterworth_example.ipynb

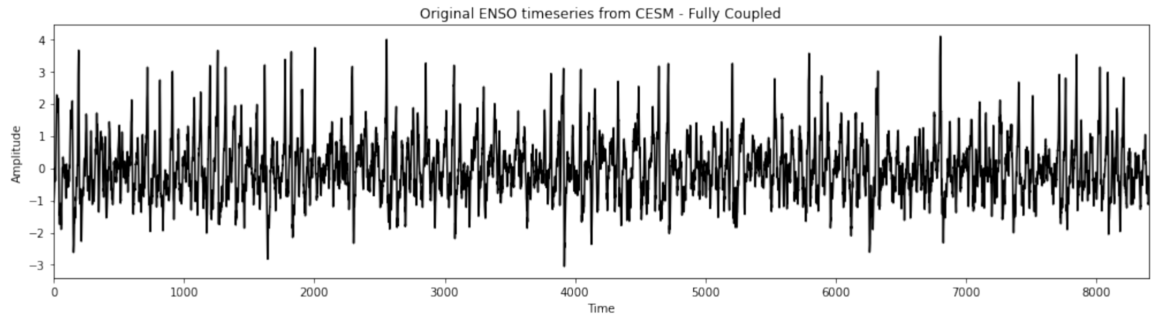
LEARNING GOALS:
1) Assess the influence of filtering on data in both the time domain (i.e., in time series plots) and the spectral domain (i.e., in plots of the power spectra).
2) Apply a Butterworth filter to remove power of specific frequencies from a time series.
3) Contrast the influence of differing 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

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

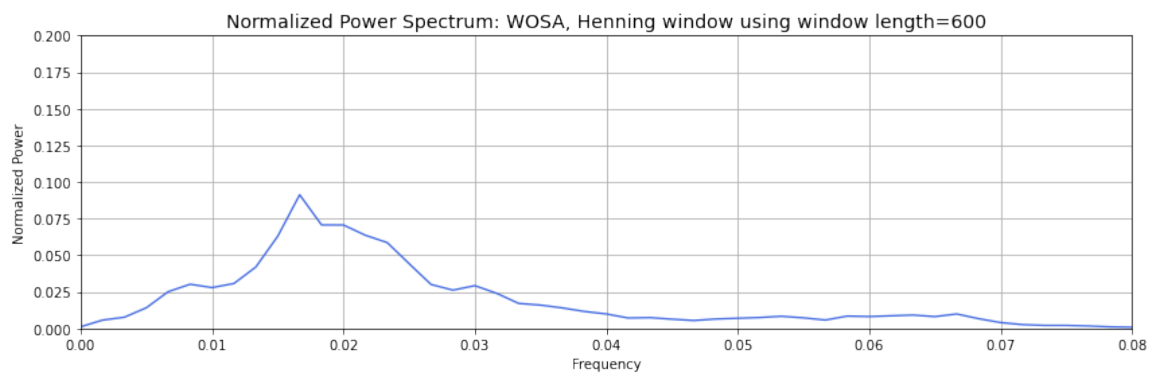
Questions to guide your analysis of Notebook #3:

1) <u>Look at your data!</u> 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?



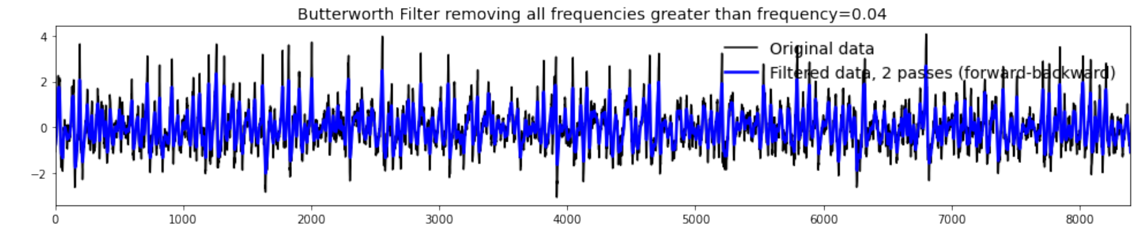
We can remove high frequencies which would allow us to visualize a less noisy time series.

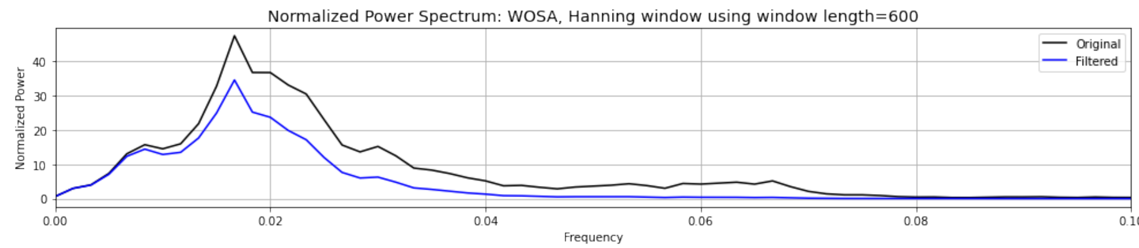
2) <u>Calculate the power spectrum of your original data.</u> Calculate the power spectra of the Nino3.4 SST index (variable called "nino34") in the fully coupled model 1850 control run. Apply the analysis to the first 700 years of the run. Use Welch's method (WOSA!) with a Hanning window and a window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

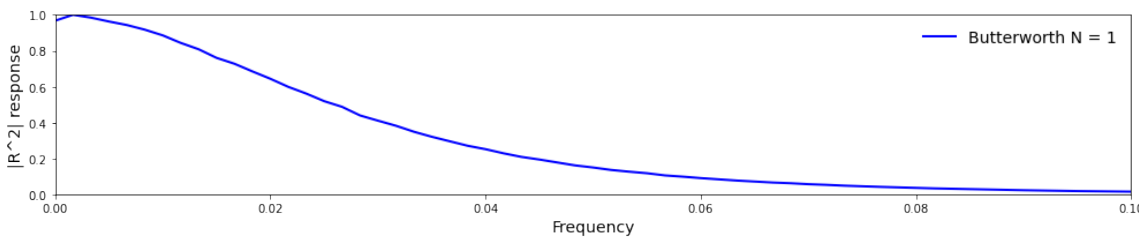


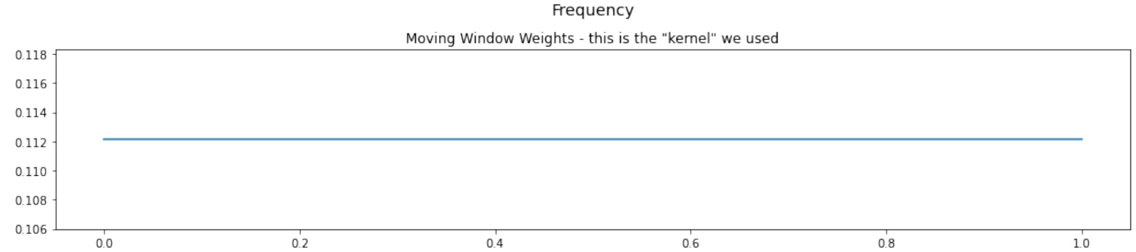
If we removed the frequencies above 0.04 we would probably be able to more clearly see the low frequency peaks.

3) <u>Apply a Butterworth Filter.</u> 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….

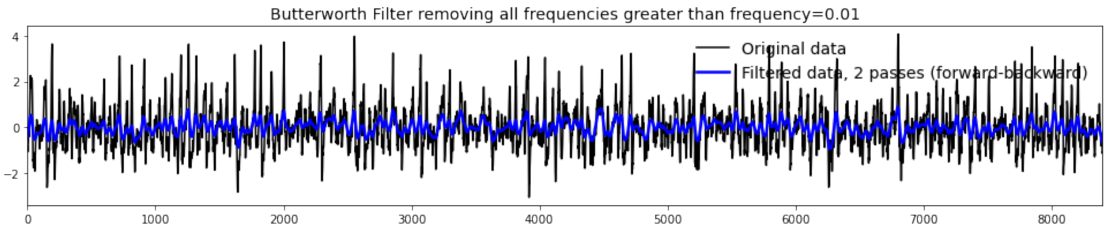


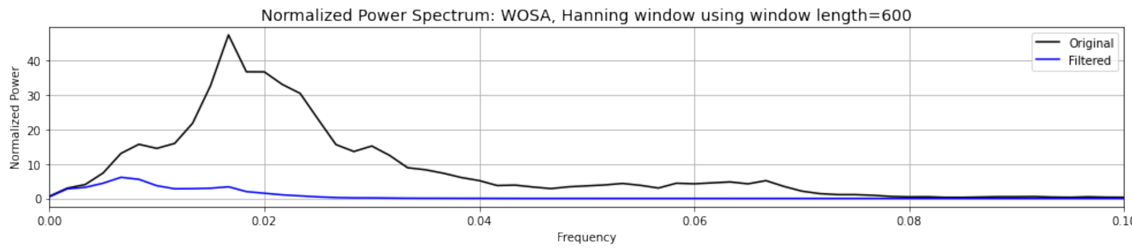


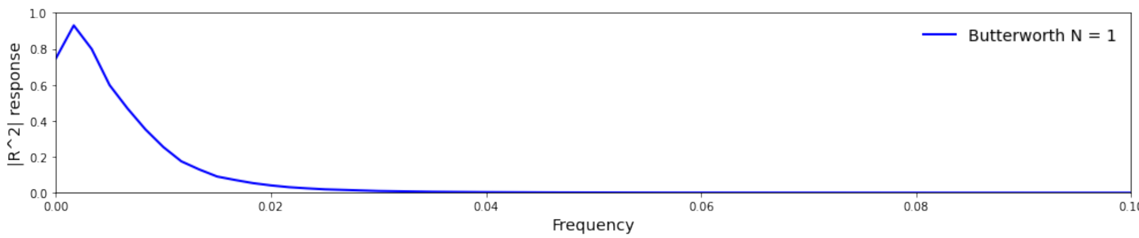


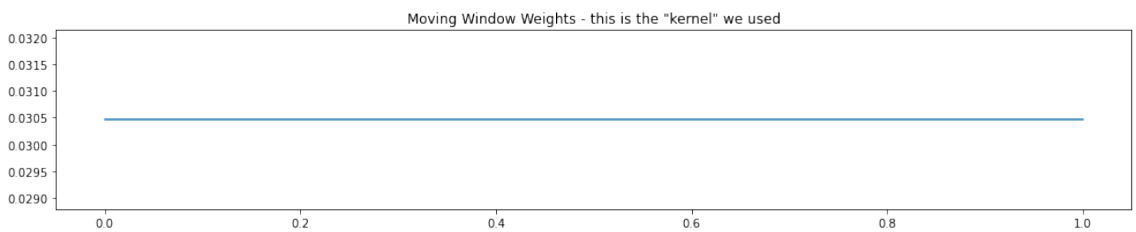


4) <u>Let's apply another Butterworth Filter and this time really get rid of ENSO power!.</u> Let's really have some fun with the Butterworth filter and have a big impact on our data... Let's remove ENSO variability from our original timeseries. Apply the Butterworth filter but this time change the frequency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



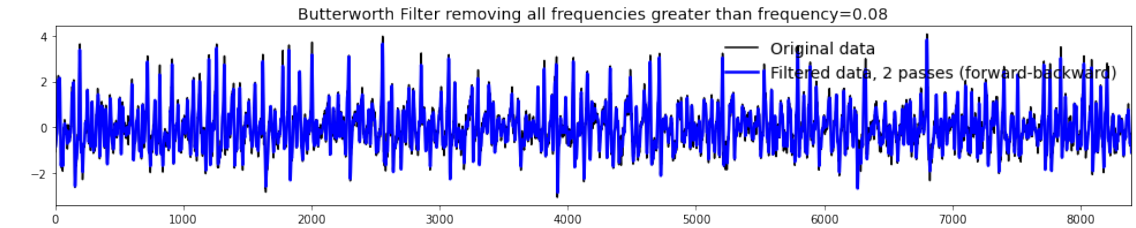


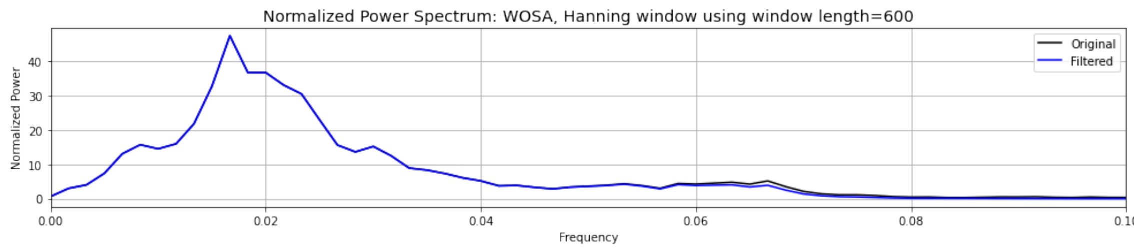


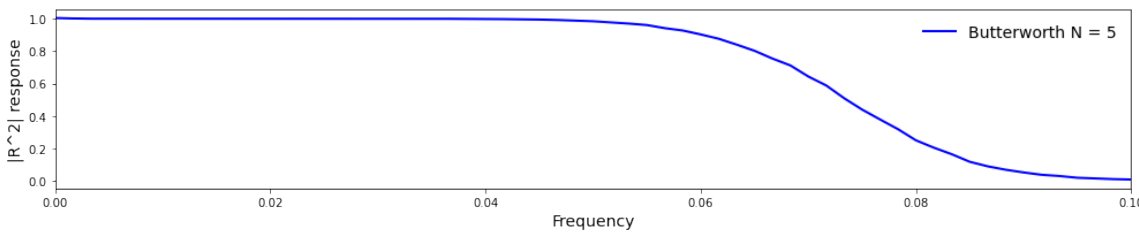


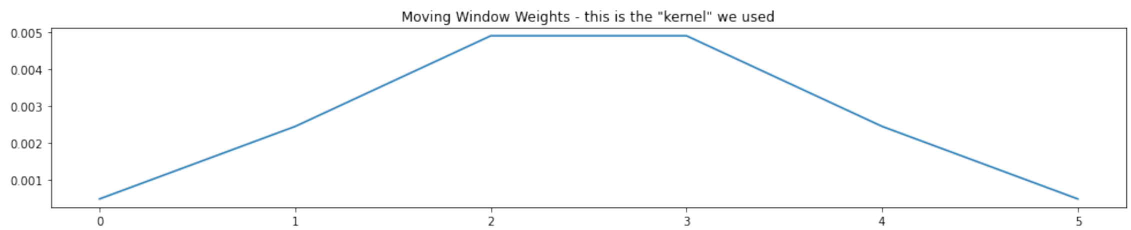
This time plot shows normalized power is really reduced but the response frequency is more pointed. Response frequencies = percent of variance retained.

5) <u>Let’s apply yet another Butterworth Filter – and this time one with more weights.</u>

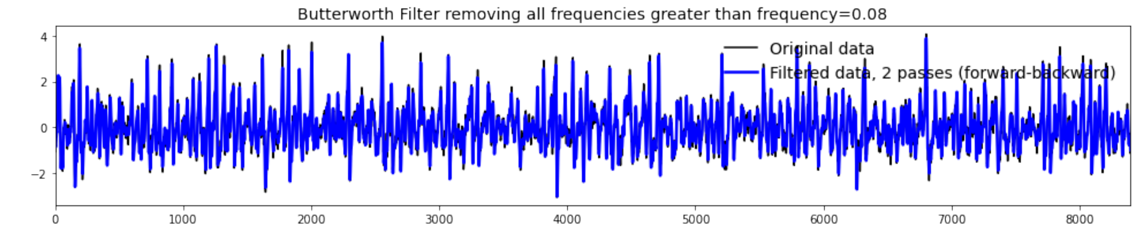


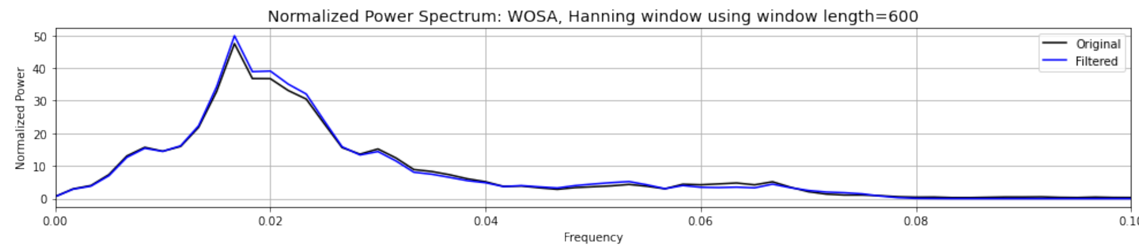


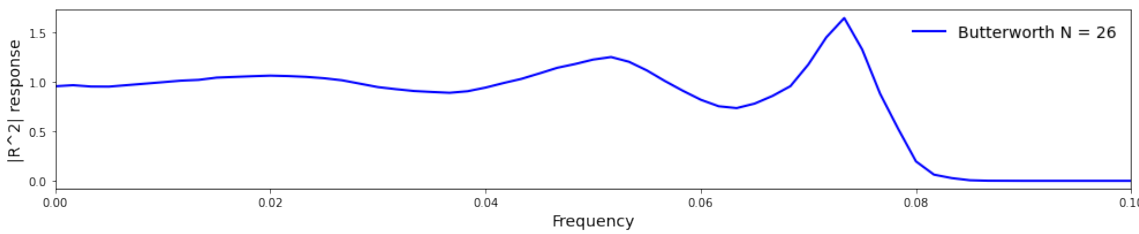


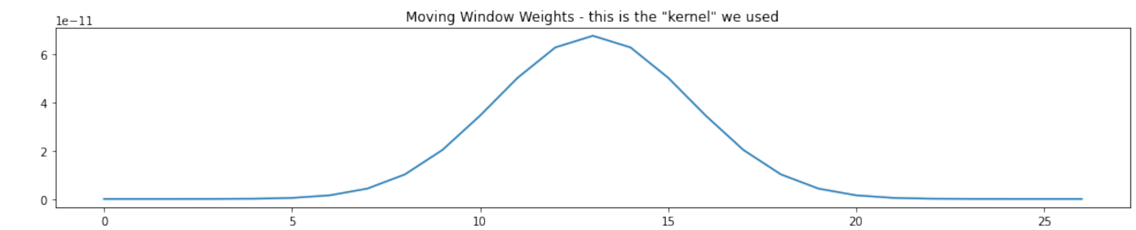


Repeat step 4) but this time change the order of the filter. In other words, increase the number of weights being used in the filter by increasing the parameter N in the jupyter notebook.









Increasing frequencies everywhere but you leave all the areas where it's zero, still zero so you are amplifying all the rest. This example shows how you could "over do-it".

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?

As you increase N – a sharper cutoff in frequency space occurs in the power spectra BECAUSE increasing the number of weights being used in the filter by increasing the parameter N doesn't filter out high frequencies.

More weights smooth it more and more removing more low frequencies

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 filters twice because it applies a linear digital filter twice, once forward and once backwards. The combined filter has zero phase and a filter order twice that of the original.

The type of extension to use for the padded signal to which the filter is applied is 'odd'.

Meaning: that it cuts off points in order to make sure that each point is filtered the same amount of times

The function also provides options for handling the edges of the signal if you need it.