**Elina Valkonen**

**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.

All the different functions treat sides of the data a bit differently, and some do backward and forward in one function, which makes most use of the data.

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.

**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?

There are two cosine waves embedded to the time series. One with frequency ~ 0.02 and another with slightly higher frequency of ~0.04.

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 weights tell which parts of the original data are “focused’ on. What I mean is that 1-2-1 filter takes more into account the big dips and hills than 1-1-1 filter, which smooths everything equally. The legtht of the weighting window tells over how many point the averaging is done. So usually causes more smoothing.

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 cutoff (which frequencies are removed) is kept the same but window length is increased, we get more smoothing. Larger window length produces less smoothing. The weighting function changes similarly. The larger window means that the weights are distributed over larger “area” resulting in more smoothing.

If the window is kept the same, but cutoff changed: higher cutoff means more power to the centralpoint(s) of the window, which will result to less smoothing. Lower cut of less power to the central point all points are treated more equally, which leads to more smoothing.

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

Smaller wn (critical frequency) less smoothing, the response function has some weight in the central points but the power is distributed more equally than if wn is increased. The biggest difference to Lanczos filter is that the shape of the response function can be changed more by changing n than what one can do with the Lanczos filter.

**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.

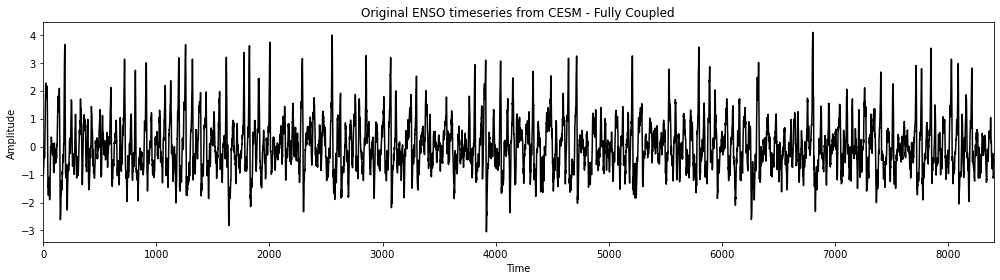
**DATA and UNDERLYING SCIENCE:**

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.*

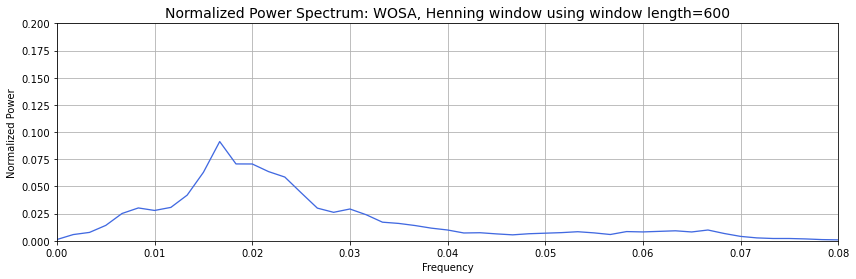
**Questions to guide your analysis of Notebook #3:**

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?

****

To me it seems that there is some high frequency variations embedded with lower frequency wave. By picking the right filters I assume both frequencies could be removed.

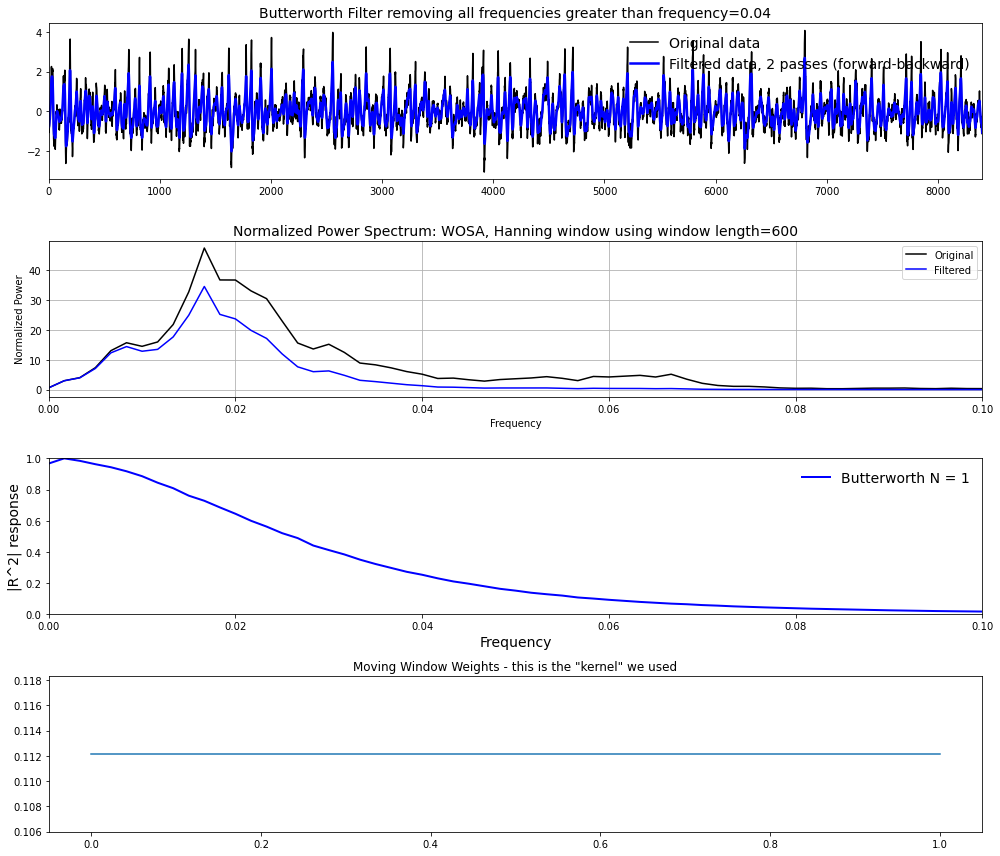
1. Calculate the power spectrum of your original data. Calculate the power spectra of the Nino3.4 SST index (variable called “nino34”) in the fully coupled model 1850 control run. Apply the analysis to the first 700 years of the run. Use Welch’s method (WOSA!) with a Hanning window and a window length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



There’s power at multiple frequencies: 0.016, 0.023 and 0.03

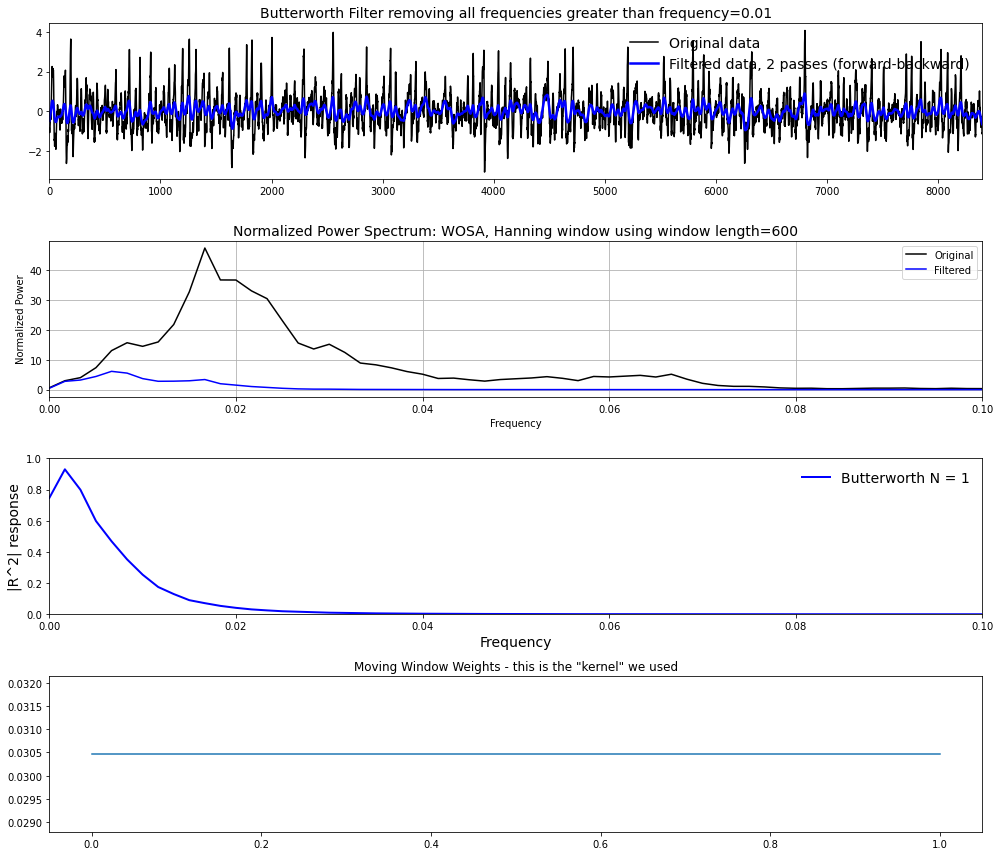
1. Apply a Butterworth Filter. Use a Butterworth filter to remove all spectral power at frequencies greater than 0.04 per month (i.e., less than 2 year). Use an order 1 Butterworth filter (N=1, 1 weight). Replot the original data and the filtered data. Calculate the power spectra of your filtered data.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

On timedomain there is not a lot of change. Highest peaks have been cut of but still quite a lot of the high frequency variation remains. This can also be seen in the frequency domain. Some power has been removed but not much.



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.

This was much better. Both at the time and frequency domain most of the low frequency variations are 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?

The response function of butterworth filter dictates that as n approaches infinity the reponse function will become a rectangle function, which only let’s by frequencies that are lower than the critical frequency (cutoff). This results in more tangential response function as N (the number of weights) is increased.

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

Due to how the ends of the timeseries are treated (data is padded at the ends of the timeseries) only filtering forward (or backward) (for example with the python function lfilter ) would cause a phase shift in the filtered data. That is why it is preferable to use the filtfilt function, which does forward/backward filtering and results in zero-phase response as demonstrated in this stackexchange discussion: https://dsp.stackexchange.com/questions/19084/applying-filter-in-scipy-signal-use-lfilter-or-filtfilt