**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.

Chart, line chart

Description automatically generated

**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?

Chart

Description automatically generated

I think filtering will remove the higher frequency oscillations from this dataset and leave the low frequency oscillations.

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.

When using a window with a longer length (1-1-1-1-1), the timeseries gets smoothed more than when using a window with a shorter length (1-1-1). The (1-1-1) filter has a smaller amplitude which muffles the oscillations of the filtered dataset.

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?

Increasing the window length removes more high frequency data. However, increasing the window length also creates lobes in the weighting function. Increasing the cutoff for the smoothing and response function makes the weighting plot less smooth and concentrates the weights in the middle of the window. This also has the effect of increasing the amplitude of the weighting.

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

This filter spreads the weights out over the entire window. This also has the effect of reducing the amplitude of the weighting. Another difference is that the weighting remains constant in the middle of the window for a certain number of data points.

**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:**

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?

**A picture containing bar chart

Description automatically generated**

Yes, this is incredibly noisy data. I would imagine the autocorrelation is quite low. By removing the high frequency variability, we will be able to see the broad changes more easily, such as the 3-7 year ENSO period.

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?

A picture containing shoji, tiled

Description automatically generated

First, we see a peak in power at a frequency of 0.017 month-1. This corresponds to a cycle of roughly 5 years. The next spike occurs at roughly 0.03 month-1 which corresponds to a cycle of roughly 3 years. This is the ENSO cycle as we know it. So, we want to remove the power at frequencies greater than 0.03 month-1 since this is noise.

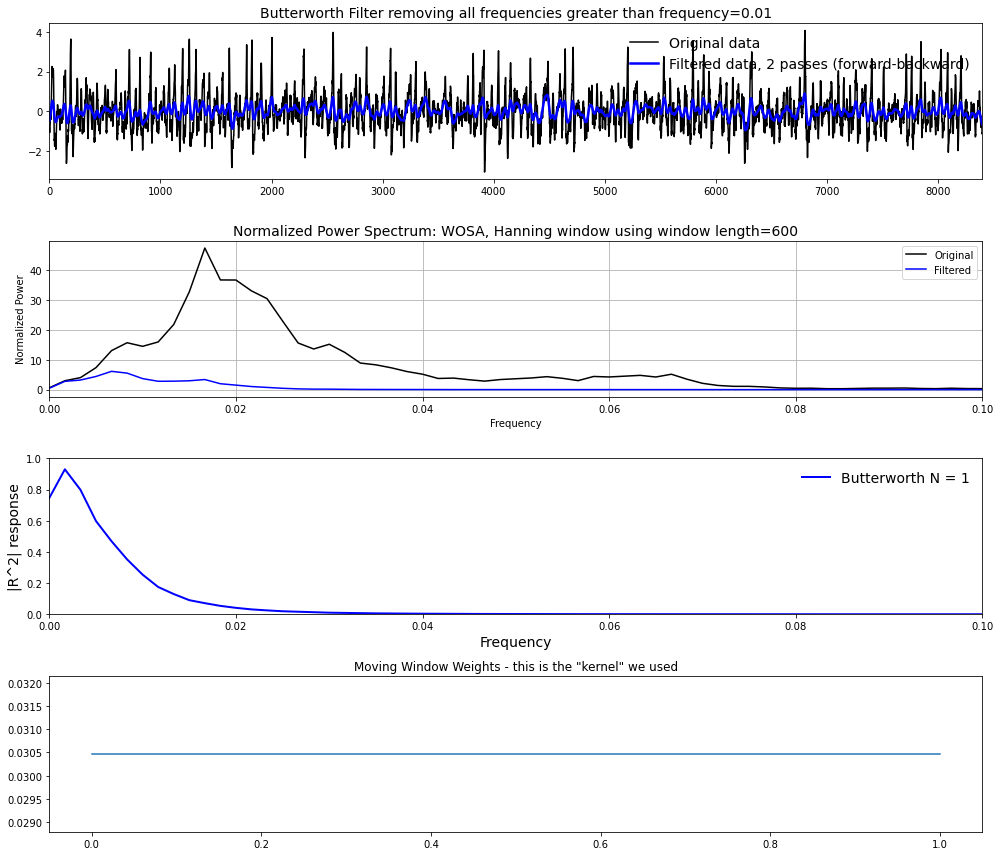
3) Apply a Butterworth Filter. Use a Butterworth filter to remove all spectral power at frequencies greater than 0.04 per month (i.e., less than 2 year). Use an order 1 Butterworth filter (N=1, 1 weight). Replot the original data and the filtered data. Calculate the power spectra of your filtered data.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

A picture containing graphical user interface

Description automatically generated

Applying the Butterworth filter has caused the amplitude of variance in the temporal domain to reduce. This has also caused the power at all frequencies to reduce except the very low frequencies. This happens because filtering removes high frequency oscillations. It is good to see that this filtering has not shifted the frequency that the maximum power occurs at. The response function shows that we do not have a sharp cutoff at our desired frequency. Instead, the response function is tapered.

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.



When we apply a frequency cutoff of 0.01 month-1, oscillations faster than 0.01 month-1 are filtered out. This has caused the amplitude of the variance to reduce significantly. This has also caused the power to remain nearly constant at frequencies larger than our cutoff. Thus, we have removed the ENSO cycle. The only peak that remains is a low frequency oscillation that wasn’t filtered.

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?

Graphical user interface

Description automatically generated

When using more weights in the Butterworth filter, the timeseries looks nearly indistinguishable. However, about double the power is found near the cutoff frequency. In frequency space, more low frequencies have a high response value because the window is larger. However, the slope is steeper. This occurs because increasing the number of weights moves the window closer to the desired cutoff frequency. Thus, the weighting must decrease quickly in order to reach 0 at the cutoff. Finally, the weighting window has increased although this is obvious because we have increased the number of weights.

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

Using the numpy convolve function with the mode 'full' uses the first data point and adds points at the end moving forward through the timeseries. This method is the same as using convolution by hand using both starting and ending. This method returns the convolution at each point of overlap in the window, with an output shape of (N+M-1,). However, the end-points of the convolution do not overlap completely, which creates boundary effects.

Using the numpy convolve function with the mode “same” returns output of length M or N, whichever is larger. This method also moves forward through the timeseries. Boundary effects will still be visible using this method.

Using the numpy convolve function with the mode “valid” returns output with a length max(M, N) - min(M, N) + 1. The convolution product is only given for points where the signals overlap entirely. Due to this, values outside the signal boundary have no effect. Again, this method only moves forward through the timeseries.

Using the filtfilt function moves both forwards and backwards through the timeseries. This method does not include the last point when moving forward and omits the first data point when moving backwards. This removes the boundary effects and allows the first and last data points to be used once (depending on forward or backward) so more of the data is used for filtering. For this reason, filtfilt is likely the best method to use for applications.