**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.

*Various Python functions and sub-function filtering options allow the user to include or ignore boundary values which can otherwise corrupt the weighted output. The filtfilt method allows the user to retain the boundary values at the cost of double filtering the internal values. Regardless of how the user filters the data, the original dataset always contains boundary data.*

**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?

*The data series equation can be broken down into four components: a red noise component based on the previous value, a white noise component, and two cosine waves whose frequencies are determined by freq and freq2, respectively. Specifically, the cosine wave frequencies are modified at each point in the series by weighted random values pulled from a normal distribution. Therefore, the frequencies that the user may expect to remove are those given as freq and freq2, though freq2 is phase shifted by pi/4 radians and is only given 0.75x the amplitude of freq. Filtering will work even though the signals are shrouded in noise since the noise is taken from a normal distribution and integrated as a multiplicative factor against freq and freq2.*

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 influence of the length of the smoothing window is that longer windows act to provide additional smoothing to each data point as the number of averaged data points used to provide the smoothing increases. A weighted window, where the heaviest weight is the central point, will provide less smoothing than a flat weighted window since the original central value has the heaviest weight. That is, a 1-2-1 filter provides less smoothing than a 1-1-1 filter. By “more smoothing” I mean a flatter curve.*

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?

*Similar to the previous non-recursive filters, increasing the window length of the Lanczos filter increases smoothing while decreasing the window length decreases smoothing. Additionally, increasing the window length effectively includes more side-lobes in the weighting function which leads to the inclusion of Gibbs effects. Increasing the cutoff concentrates weights in the middle of the window, while decreasing the cutoff concentrates weights at the edges of the window.*

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

*The response function of the Butterworth filter appears similar to the Lanczos filter when the window length was limited to a value contained entirely by the main lobe. It generally resembles a discrete representation of a Gaussian distribution.* **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?

*Possibly. There appears to be some level of regular oscillation about a mean, though the data looks far too noisy to point at anything in particular. I suppose a better method than just looking at it would be an objective analysis.*

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?

*There is power that I might be able to remove, as identified by the highest peak in the power spectrum, between about 0.0125 and 0.0275 ­-1, which is between 6.67 and 3 years. Higher frequencies can also be removed but are generally insignificant.*

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 effect of the filter in the time domain is minimal as the data smoothing is minimal, while the effect of the filter in the frequency domain is also minimal as the significant peaks are still prominent. The response function of the filter is simple as well, where the signal gain gradually diminishes after the first few points.*

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.

*Yep. By removing spectral power at frequencies higher than once per eight years we effectively remove the ENSO signal from our data. In the time domain, the oscillations of the ENSO index are minimized, with no single signal amplitude exceeding 0.5. The filter is similarly influential in frequency space, where the prominent peaks in the normalized power spectrum are almost entirely flattened. The response function matches the diminished normalized power spectrum peaks, as the only frequencies which retain their power are below 0.01.*

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?

*Increasing N on the filtered dataset leads to a sharper cutoff in frequency space due to the increased tangency of the Butterworth 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?

*Filtfilt filters twice to remove the edge effects otherwise caused by insufficient sampling points at the edges of the datasets. This does incur a penalty of double-filtering the internal data. It filters based on the weights passed to it, which can be as simple as a 1-1-1 filter or as relatively complicated as weights derived from the Butterworth method or beyond.*