**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.

**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?

High frequency signals can be removed with smoothing filters and also the frequency input into the underlying equation will also be able to be removed since there will be significant power there.

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 wider windows smooth the data to a greater degree, so that individual peaks become mostly smoothed out. It is clear that the lower frequencies have greatly reduced power

A picture containing text, antenna

Description automatically generated

A picture containing chart

Description automatically generated

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?

Changing the Lanczos window changes the number of data points used in the smoothing. Generally, more data points lead to smoother data across all filters. It causes the response function to have use more data points to calculate the filtered value. Interestingly, the response function seems to be more or less the same shape but shifter to the middle of the window.

Graphical user interface, chart

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

Changing Lanczos cutoff value controls the minimum frequency of variance that is allowed. Signals with frequencies above the cutoff value are removed with the filter. So by lowering the cutoff value you remove more frequencies of variability, and by raising the cutoff value you include more high frequencies

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

It looks very similar. Theoretically it should perform better around the edges of the data since it goes both ways and uses all the data.

Chart

Description automatically generated

**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?

Yes, anywhere there is power, you can filter out the frequency. This data appears to have high frequency variance and some lower frequency variance as well.

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?

In this data there seems to be a peak of power in low frequency and then a long tail into the higher frequencies. It looks like filtering out this tail could be useful.

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….

Since most of the power is retain outside of high frequencies where there was minimal power anyway, the data is minimally changed in time. The opposite of tangency is observed in the response function which helps show why the filtering was minimally different.

A picture containing graphical user interface

Description automatically generated

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.

Graphical user interface

Description automatically generated

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 is useful because it increases the tangency significantly. This is very apparent when comparing the cutoff of 0.01 with N=1 (above) to N=9 (below). The higher N does a better job at both maintaining power for low frequencies and cutting power off for higher frequencies! This is ideal.

Graphical user interface

Description automatically generated

However, it is important to note that increasing N has its limits. It can both completely break the filter so that the results are nonsensically amplified or damped, or it can somewhat subtly alter the responses of the powers as shown below. For these reasons always checking the power spectrum and response functions for the filtering functions is good practice.

A picture containing graphical user interface

Description automatically generated

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 edges are padded with the specified symmetry (default=odd). By padding the edges the filter is able to act on the beginning and end data points since the values on either side of the data point are needed to calculate the filtered value.