**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.

Check. See documentation for padding effects. Look into Gustafsson’s method (1994) for further 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?

Graphical user interface, chart

Description automatically generated with medium confidence

We expect to be able to remove the data at our two oscillation frequencies using filtering. This corresponds to 52/256 and 100/256 frequencies.

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.

As we increase the length of our kernel, we get a greater degree of smoothing in our filtered data. This is because of the influence of further points is felt by our current point. In frequency space, this is corresponding to a low-pass filter with a higher effective cut-off frequency.

We see that the 121 filter kernel does not suppress the amplitude of our signal at peaks as much as the 111 filter. While we are normalizing each of the coefficients so that the amplitude of the original data is preserved, the 121 filter has less attenuation at higher frequencies. The effective cutoff frequency is probably higher.

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?

As we increase the window length, we get more smoothing. This make sense, because it is effectively increasing the length of the kernel, and we get influence from points farther away.

As we increase the cutoff, we get less smoothing. This cutoff makes the filter kernel taller and narrower. The overall shape of the kernel is just a triangle.

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

As we use the butterworth filter, we see the response function is closer to a gaussian shape, but has a flat top for odd values of the filter order, and a peak for even values. A benefit of this filter is that it can be used in real time.

**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?

Visually, we see strong periodicity in the signal that we expect to see in the frequency domain. You have to zoom in so that the signal doesn’t look like noise.

**Text

Description automatically generated**

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

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….A picture containing graphical user interface

   Description automatically generated

As expected, the response of our filter attenuates frequencies which are not very present in the signal anyway. The filtered signal is very close to the original signal, and the spectrum of the filtered signal still has a peak at the ~.015 1/ month frequency.

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

We see the desired effect. As we move the cutoff frequency down, we filter out the ENSO variations.

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

We see the sharper cutoff in the response function due to the higher filter order. Because we are filtering offline, we do not get adverse effects of the phase delay that comes with this high tangency due to the forward and backward filtering.

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 phase delay in our signal. This is only possible when we have all of our data a-priory (i.e. not suitable for online filtering). Edge effects are assumed that the data is from an odd function, so points are mirrored about the endpoint and the y axis. Better endpoint behavior may be attained with Gustafson’s Method.