**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.

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.

**The weighting function averages the given point depending on the proportions of the weighting function. E.g. 1,1,1 is weighting yesterday, today, and tomorrow equally.**

**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?

x[j] = x[j-1]\*alpha + factor\*np.random.randn()+1.0\*np.cos(2.0\*np.pi\*(1.-0.01\*np.random.randn())\*freq\*j) + 0.75\*np.cos(2.\*np.pi\*(1.-.001\*np.random.randn())\*freq2\*j-np.pi/4.).

**Where:**

freq = 52./256. ## oscillation frequency

freq2 = 100./256 ## oscillation frequency 2

**Periods of ~2.5-5 timesteps (1/52/256 and 1/100/256) appear to be the main frequencies.**  **A picture containing object, antenna

Description automatically generated**

1. 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 longer and flatter the smoothing window the smoother the outputted data.**

Chart, line chart

Description automatically generated

Chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart

Description automatically generatedChart

Description automatically generated

1. 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?

**Window length 25 (below) compared with window length 5 (below that). The larger window picks up on the larger oscillation shapes better. Smaller window changes gradient more quickly to be more noisy.**

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

**Using a window length of 25 again but now with a cutoff of 3/11 rather than 1/11 we find the weights are more concentrated in the middle of the window.**

Chart, line chart

Description automatically generated

**Now we increase the window length to 50 and increase the cutoff to 8/11 to get a very noisy result as we have negative influence from the surrounding data.**Chart

Description automatically generated

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

**The Butterworth filter never goes below zero in its weights so it reduces the lobe effects.**

Chart, line chart

Description automatically generated

**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?

**Yes, there appears to be high frequency data as there are many spikes.**

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 is a main peak at ~0.01-0.025 months-1 (100-40 months), with the general area of 0.001-0.08 (1000-12.5 months) being where most of the power is concentrated.**

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….

**Yes, this didn’t do much as most of the important frequencies are <0.04 and so they are retained. There isn’t much of a peak in the response function. So there is a broad range of frequencies in the smoothed data. I.e. the smoothed power spectrum is the original x response function.**

**A picture containing graphical user interface

Description automatically generated**

1. 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 does now make a difference as we can see the filtered power spectrum has a peak at ~0.0075 where there wasn’t much of a peak in the original data.**

Graphical user interface, application

Description automatically generated

1. Let’s apply yet another Butterworth Filter – and this time one with more weights (still 0.01 cutoff). 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?

**All we have done is increase the order of the Butterworth frequency. The sharper cutoff is due to adding more weights so the contribution from frequencies ~0.01+ goes from a small contribution to zero.**

Graphical user interface

Description automatically generated**Graphical user interface

Description automatically generated**

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

**The function works by filtering forward and then filtering backwards. A linear digital filter, order is twice the original and zero phase. filtfilt allows you to select the type of padding you can add at the edges to try and avoid spectral leakage. The default padding is ‘odd’. This also has the effect of being able to increase the weight to 26, much higher than before. N.B this will result in the response function being above 1 at times!**

**It ensures each part of the spectrum is filtered the same number of times.**

**Chart, histogram

Description automatically generated**