**ATOC5860 – Application Lab #5**

**Filtering Timeseries**

**Notebook #1 – ATOC5860\_applicationlab5**

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.

See jupyter notebook. The notebook shows how the different options of modes for np.convolve treat the data differently. To briefly summarize, mode = ’full’ is the default and appears to treat the data the same as doing it by hand. The mode = ‘same’ gives the same output but with boundary effects still visible. For the mode = ‘valid’ , the convolution product is only given for points where the signals overlap completely and values outside the signal boundary have no effect. One can also use ‘sig.lfilter’ from the scipy.signal package which uses np.convovle so it also starts with the first value but cuts off the last value at the end. Finally, one can use ‘sig.filtfilt’ which can be applied forward or backward and going each way, you get to filter the edges from both directions at least once. This method makes the most of the data at the edges and makes the most of all of the data so it is a good choice. The notebook also discusses padding

**Notebook #2 – Filtering Synthetic Data**

**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?

A picture containing chart

Description automatically generated

From the underlying equation, it looks like we input frequencies of 52/256 and 100/256 so I expect that we will be able to remove one or both of these frequencies. I think we can remove 52/256 with a low-pass filter.

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.

Type #1: 1-2-1 filter (filtering in the time dimension, non-recursive moving average filter)

Graphical user interface

Description automatically generated with medium confidence

Type #2: 1-1-1 filter, another example of a non-recursive moving average filter in the time dimension

Graphical user interface, application

Description automatically generated

Type #3: 1-1-1-1-1 filter, another example of a non-recursive moving average filter in the time dimension

Graphical user interface, application

Description automatically generated

When comparing the different filters, we see that the 1-1-1-1-1 filter smooths the data more than the 1-1-1 filter. I believe this indicates an increase in the length of the smoothing window, meaning the 1-1-1-1-1 filter has a larger smoothing window than the 1-1-1 filter.

When comparing the 1-2-1 filter with the 1-1-1, the results are very similar but the 1-2-1 filter results in slightly larger amplitudes. This makes sense because the 1-2-1 filter is doubling the power in one section of the data compared to the 1-1-1 filter.

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?

Window length of 25:

Graphical user interface

Description automatically generated

Window length of 5:

Graphical user interface, chart, line chart

Description automatically generated

Window length of 50:

Graphical user interface, chart, line chart

Description automatically generated

In the Lanczos filter, if you decrease the window length, you get less smoothing whereas if you increase the window, you get more smoothing. If you increase the cutoff, the weights get concentrated in the middle of the window whereas if you decrease the cutoff, more weight gets concentrated at the end 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.

Graphical user interface, chart, line chart

Description automatically generated

The Butterworth filter response function is different than the ones above because it has a more gradual slope and is flat at the top. The sides gradually fade out to 0 and there are no oscillations or side lobes.

**Notebook #3 – Filtering ENSO data**

**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

Since the data has time steps of months, I would suspect that we can remove the annual and seasonal frequencies in the data. There also seems to be some lower frequency oscillations in the data that we may be able to remove.

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?

Chart, line chart

Description automatically generated

From the plot above, it looks like there is power at lower frequencies, around 0.08 – 0.03 per month, that we may be able to remove with filtering. Specifically, there are peaks at frequencies of about 0.008, 0.017, and 0.03 per month that we may be able to remove. Also, there are some peaks at higher frequencies, such as 0.067 per month, that we may be able to remove.

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. Graphical user interface, application

Description automatically generatedWell that was pretty boring… we still have most of the power retained….

From the plot above, we can see that the filter maintained the original shape of the time series and the original shape of the normalized power spectrum, for the most part. The filtered time series looks very similar to the original but with lower values for the higher peaks. In other words, the higher frequency peaks in the data have much lower amplitudes in the filtered data than in the original data. The filtered power spectrum looks the same as the original at lower frequencies but with lower power. At higher frequencies, starting around 0.04 per month, the filtered power spectrum decreases to 0 and stays at 0. This shows that we did filter out frequencies higher than 0.04 per month.

The third panel in the plot shows the response function. We can see that at lower frequencies it maintains most of the power and gradually slopes off to 0 at higher frequencies.

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

In the time domain, the filtered timeseries keeps the lower frequency oscillations but does not keep any of the higher frequency oscillations. We can see this again in the second panel where the filtered power spectrum has very low power at the beginning (before 0.01 per month) and zero power at the end, starting to decline around 0.01. The response function reiterates this because it spikes at the lower frequencies and decreases quickly to 0.

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?

N = 5

Graphical user interface

Description automatically generated with medium confidence

N = 8

Graphical user interface, chart

Description automatically generated

N = 10

Graphical user interface

Description automatically generated

N = 26 for cutoff frequency of 0.08

Chart

Description automatically generated

As N increases, we see a sharper cutoff in frequency space, as expected. I think this is because we have more weights and thus can decrease the power in the filter more quickly.

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 the edges from each direction at least once, thus applying the filter symmetrically. This allows it to make the most of the data at the edges and the most of all the data. From the scipy page for the filter, we see that the combined filter (forward – backward) has zero phase and a filter order twice that of the original.