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.

Ok, done. Got it!

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?

With filtering I would expect to be able to remove the frequencies at which we provide power, i.e. 52/256 and 100/256. In addition, since the time series is relatively red, I expect to be able to remove power at low frequencies corresponding to autocorrelation.

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.

Indeed, all filters smooth the data set considerably, so that is great news! Increasing the length of the smoothing window reduces power at higher frequencies. Likewise decreasing the amplitude of the smoothing window i.e. 1-1-1 instead of 1-2-1 reduces power at high frequencies, leading to more smoothing. So to summarize, the different filter types exhibit different behavior.

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?

Increasing the window length (widening the response functions) leads to more smoothing, whereas decreasing the window length leads to the opposite behavior. Increasing the cutoff reduces smoothing by concentrating weights in the center 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 Butterworth filter allows us to explicitly state what frequencies we want to remove. Whereas the other filters simply smooth by using a moving average.

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?

Indeed, the data is an anomaly. Off to a great start! I see variance at cycles corresponding to roughly 5 years (ENSO) so we should be able to remove that!

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 a lot of power and frequencies between 0.01 and 0.03 months^{-1}. So, we should be able to remove power there.

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 original data and filtered data look quite similar, as do the power spectra. This is because there is not much power a frequencies greater than 0.04 months^{-1}. The response function emphasizes frequencies less than 0.04 months^{-1} as expected. Take away here that the filter does do much because the data doesn't have power at these high frequencies. So that is a good gut check!

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.

Now we have really filtered the data set! The data set is much smoother, because power at high frequencies has been considerably reduced. Interestingly the amplitude of the variations is reduced significantly when filtering. So that means that much of the data set's power was at high frequencies. As was the case in part 3, the response function now emphasizes frequencies less than the filtering threshold. So, it seems to be doing what we want, event with a lower order filter (N=1) ☺

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?

As N increases the filter response function approaches a boxcar function, which means that the cutoff frequency in the power spectra should become steeper. However because our filter is starting to become a boxcar, high frequency wobbles are needed to resolve the sharp edges which allows spurious relatively high frequency information to be retained.

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 function filtfilt filters twice so that the phase of the filtered data is the same as the original data. Edge effects are by default treated using padding at the ends of the signal.