Homework 5. Butterworth Filters

The Matlab and Python signal processing modules have an overwhelming array of options for designing and implementing filters, but for many geo-scientific applications we can use very simple filters. In this exercise we are going to explore the properties and use of a Butterworth IIR digital filter – perhaps the most commonly used filter.

Read the `butter()` documentation on Matlab or Python. It is important to remember that

- The order of the filter `n` is the order of the polynomials defined by `a` and `b` (i.e., the number of poles or zeros). Usually `n` is chosen to be even. The higher the order the sharper the frequency cutoff.
- The frequency cutoff limit(s) of the filter `wn` are specified in units that go from 0 to <u>1 at the Nyquist frequency</u>
- `b` and `a` are as defined in class. That is, the response of the filter is

$$G(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z + b_2 z^2 + \ldots + b_M z^M}{a_0 + a_1 z + a_2 z^2 + \ldots + a_N z^N}$$

Once you've obtained the coefficients, to apply a filter to the sequence `x`, execute the command `lfilter()` in python or `filter()` in Matlab.

This command calculates

$$y_j = \sum_{i=1}^{N} b_i x_{j-i} + \sum_{i=1}^{M} a_i y_{j-i}$$

Note that the indexing is different in Python and Matlab conventions. Now for any given output y_j, the filter requires the N previous input values of x and the M previous output values of y and so will not be getting all the inputs until j = max(N,M) + 1. Furthermore the feedback component will typically require several times M prior values to fully settle down (if you think about y_{M+1}, it will be getting prior values of y but they will in turn have been calculated without the necessary prior values) – Unless the time series starts off with zeros, always give the filter a little extra data and delete the first part of the output.

1. For this question start with a 128-sample time vector with unit sample interval and a start time of 0 and an input time series function comprising a delta function at 0.

(a) Create a 2nd, 4th, 8th and 16th order low-pass Butterworth filters with a cutoff at 0.5 of the Nyquist frequency. Apply them individually to the delta function time series and plot the results.

(b) For each filter in (a) plot its amplitude and phase.

(c) Comment on the differences of the time- and frequency-domain responses of the filters.

(d) Repeat (a) except for a high pass filter with `Wn = [0.8]`. Comment on the results.

(e) Repeat (a) except for a notch (stop) filter with `Wn = [0.4 0.6]`. Comment on the results.

(f) The `filtfilt` command in both Python and Matlab implements a zero-phase filter by running the filter through the filtered data in both directions. Use a 128-sample time vector with unit sample interval and a start time of -63 and an input time series function comprising a delta function at t = 0. Apply the filters from part (a) to the delta function and plot the results. Comment on the results.

2. Consider the Mauna Loa CO2 concentration time series from homework 2.

(a) Plot the data and its amplitude spectra

(b) Construct a 4^{th} order Butterworth high-pass filter with an appropriate cutoff frequency so that you remove the multi-decadal trend and leave only the annual variability signal. Plot the filtered data.

(c) Construct a 4^{th} order Butterworth low-pass filter with an appropriate cutoff frequency so that you remove the annual trend and leave only the long-term signal. Plot the filtered data.