Library of Python Functions

These are the basic functions that I use to go back and forth between the s and z domain, compute IIR coefficients and plot phase and frequency responses. As mentioned in the article I am not a programmer by trade and there are probably many improvements possible here. It is easy with Python to get a little confused around arrays, lists, and other ways of packing and unpacking data. Usually what seems obvious works but there are times when an explicit declaration is necessary. Also there is not thorough checking for bad inputs so they will simply fail or give a bad output. As noted also I use positive frequency for both the s and z domain in my computations, the built-in *roots()* function returns negative values as it should and they are inverted as necessary in my code.

dB(x) - Returns 20*log10(abs(x)) for any length argument.

f_warp(f, fs) – Returns the "warped" frequency of interest (f) for IIR filter coefficients. Fs is the sampling frequency.

f_from_z(z, fs) – Returns a frequency from the point z inside the unit circle.

z_from_f(f, fs) – Returns the point z inside the unit circle corresponding to frequency f.

Fs_at_f(Poles, Zeros, f, norm = 0) – Returns the complex value/s for s domain frequency response at list of frequencies f. The transfer function is described by its poles and zeros. If norm is true (1) the output is normalized to 1 at the maximum value.

Fz_at_f(Poles, Zeros , f, fs, norm = 0) – Returns the same for the z domain poles and zeros (here fs is needed also). The pole and zero frequencies are those that would be returned from *f_from_z(z, fs)*.

z_coeff(Poles, Zeros, fs, g, fg, fo = 'none') – Returns the a and b coefficients of the IIR filter defined by the poles and zeros. G and fg are the gain in db and frequency at which that gain is set. Fo is the frequency of interest, that is, for a peaking filter fo would be the peak while for a low pass or high pass filter it would be the break point. If 'none' or not specified there is no frequency warping and all mapping is simply by the bilinear transform.

biquad_filter(xin, z_coeff) – Returns xin filtered by the biquad filter defined by the z domain coefficients, [[a_0, a_1, a_2] ,[b_0,b_1,b_2]]. By convention the b coefficients are the numerator and the a are the denominator with $a_0 = 1$.

write_txt(filename, y) – Writes a simple text file of the values of the array y.

read_txt(filename) – Returns an array from the values in a text file.

read_dat(filename) – Returns a mono or stereo file in the SoX .dat format.

write_dat(filename, sr, left, right='none') – Writes a file in the SoX .dat format. Sr = the sample rate. Left and right are equal sized arrays of time samples.

plot_fft_log(fs, data, diff = 'none', start = 0,stop = 0, delay = 0) – Plots the FFT of a time domain waveform. Fs is the sampling frequency and data are the samples. If diff is specified the difference between data and diff is plotted. Start and stop are the start and stop frequencies for the plot, 20Hz and 20kHz are the defaults. A delay in samples is specified to compensate for phase wrapping, the default is 0. This function compensates for actual bin frequencies so, for example, a 65536 point FFT of a waveform sampled at 48kHz gives the right frequencies.

plot_fft_lin(fs, data,diff = 'none', start = 0, stop = 0, delay = 0) – Same function for a linear with frequency plot.

plot_dat(sound) – Plots a sound file read by *read_dat()* or in the .dat format.

plot_gain_phase(Poles, Zeros, zPoles = [], zZeros = [], fs = 48000.,fo = 1000. ,start = 20, stop = 20000,style = 0) – Plots the gain and phase vs frequency of a frequency response defined by a set of Poles and Zeros. If zPoles and zZeros are specified it becomes a comparison between the s and z domain with fs being the sampling frequency. Fo is the frequency where both responses are equalized. Start and stop are again the limits on the plot while style = 1 is a difference plot and style = 0 is a plot of both responses on the same axes.

min_phase_FIR(Poles, Zeros, fs, outfile, plot = 0) – Generates a minimum phase FIR filter in mono .dat format from the s domain poles and zeros at a sampling frequency fs. The output is written to ‘outfile’ and it expects a HOME directory to exist. You can change the code to output to whatever you want. Plot = 1 plots the result so you can examine it.