Physics 567
Homework 3
Due Monday, February 23, 2026

In this exercise you will perform wavelet filtering to suppress noise, similar to what was demonstrated in class with the LIGO data. You are given a sample of digital music, `Lacumparsita-noisy.wav`, sampled at 44.1 kHz.[1] Listen to it, and you will hear the noise problem. Fortunately, we have obtained an example of noise from the recording apparatus, `noise-sample.wav`, using identical conditions but without the musicians. Of course, the noise is random, so there is no straightforward way to subtract it.

Procedure. Obtain Torrence & Compo's wavelet software. Verify that you can read, wavelet transform, and accurately reconstruct the data, and save it as a WAVE audio file, `reconstructed.wav`. Listen to the original and reconstructed WAVE files carefully, to reassure yourself that there is no difference. If they don't match, check whether you allowed a wide enough range of periods to cover the range of human hearing, with dense enough sampling in `log(period)` to allow accurate reconstruction.

Use the noise sample to calculate an array of power thresholds for significance at each wavelet scale. You could do this by using Torrence & Compo's routine `wave_signif`. It can also be done by finding the average power of the noise for each period, and then setting a threshold at some multiple of this value.

Next, obtain the wavelet transform (I'll call it `data_w`) and the wavelet power spectrum (`data_wps`) of the data. Use the thresholds you established in the last step to filter the wavelet transform. This entails something like `mask = (data_wps < threshold)` and then `data_w .*= mask`. Then reconstruct the music from the filtered wavelet transform and save as `filtered.wav`. Play with the significance threshold a bit to obtain the best sounding result you can. This is not really about statistical significance; it is art, not science. Also save the difference between the filtered data and the original data in `residuals.wav`.

Turn in your code(s) and WAVE files electronically to the grader:

- `reconstructed.wav`
- `filtered.wav`
- `residuals.wav`
- `filtered.wav`

If you try the optional part (below), please let me know how it goes. I'd like to listen to your results!

Optional. The wavelet filtered music will exhibit a sort of fluttering sound. This is called "chatter". It is the result of sudden transitions in state between sound off and sound on in any of the various wavelet periods. If you would like an extra challenge, it is possible to eliminate this problem. My approach was to filter `mask` along the time axis only, as follows:

1. Median filter to remove short positive and negative glitches. This can be time consuming.
2. Dilate by boxcar smoothing and then setting nonzero values to 1.
3. Make the filter attack and let-off gradual by again smoothing with a boxcar.

Median smoothing is far slower than all the other operations. My finished code takes about 15 minutes to run.

[1] Gerardo Matos Rodríguez (Public domain).