Gravitational-Wave Data Analysis
Exercises – Day 2

Peter S. Shawhan
CGWA Summer School, May 29, 2012

Today you will get to search for a signal hidden in some data, using a few kinds of filtering. This is supposed to be like LIGO data, so the sampling rate is 16384 Hz.

1. Preparations

a. First, get copies of the files to work with today: the Day 2 "data file" (called `day2data.txt`) and "inspiral template" (`day2template.txt`).

b. Read the `day2template.txt` file into Matlab using the `load` function (or the `dlmread` function). How long is this time series, in seconds? Plot it with time units of seconds on the *x* axis. This is the signal you'll be looking for, which is roughly what you would expect from the last part of an inspiral of two 10-solar-mass black holes (with the amplitude artificially ramped up from zero at the beginning). Approximately what frequency does this signal start at? Approximately what frequency does it end up at?

c. Read the `day2data.txt` file into Matlab, and plot it with time units of seconds on the *x* axis. Can you see any hint of a signal?

d. Plot just the first 10000 data points so you can see what time structure it has.

e. Use the `pwelch` function to calculate the PSD of the data, specifying the sampling rate so that the horizontal axis is true frequency. Note that the spectrum isn't white.

Before we search for the inspiral signal using matched filtering, let's start with some basic digital filtering.

2. A very simple digital filter

a. Apply a simple first-difference filter, i.e. $y_k = x_k - x_{k-1}$, to your time series. You can do that this way, for example: `filtdata=data-[0 data(1:length(data)-1)];`

b. Plot just the first 10000 data points of the filtered time series so you can see how the time structure has changed compared to what you saw in part 1d. What happened?

c. Use `pwelch` to estimate the power spectrum for the filtered time series, and compare it to what you got in part 1e. Do you understand why it changed the way it did? The ratio of filtered to unfiltered is the magnitude of the transfer function of this particular filter.

3. A better digital filter

a. Butterworth filters are a pretty standard class of all-purpose IIR filters. Figure out how to use Matlab's `butter` function to design an 8^{th}-order Butterworth band-pass filter to select the frequency range 35 to 250 Hz; that produces the a_i and b_i coefficients that define the filter. Then use Matlab's `filter` function to apply it to your original data.

b. Plot your filtered time series. Do you see any evidence for a signal now?

4. Matched filtering in the time domain

a. Calculate the correlation between the (original unfiltered) data and the template for all possible offsets of the template. You will probably want to use a '`for`' loop to do this. Store the results of the correlation for each offset in an array (one correlation value per offset). This takes a long time! It will help to pre-create the correlation array, e.g. by doing: `corr=zeros(1,length(data)-length(template));`

b. Plot the correlation array that you just calculated. Do you see any evidence for a signal?

5. Down-sampling [if you have time]

So far, you've looked for a signal whose frequency band is far below the Nyquist frequency. In other words, the sampling rate is much higher than it needs to be, and you can down-sample (or "decimate") the time series to reduce the amount of data you have to work with.

a. Calculate the RMS for your original unfiltered data time, to compare to later.

b. Try down-sampling your time series by simply selecting every 16^{th} sample. (This produces a time series with 262144/16 = 16384 samples with a sampling rate of 16384/16 = 1024 Hz.) Calculate the RMS for this filtered time series; how does it compare with what you found in part a? Do you understand why?

c. Before resampling, you really should use an anti-alias filter to remove all of the noise power at frequencies above the new Nyquist frequency. So, construct a 6^{th}-order Butterworth low-pass filter with a corner frequency of 512 Hz and apply it to your original time series, and then select every 16^{th} sample from the output time series. Plot the resulting time series.

d. Calculate the RMS for this properly down-sampled time series; how does it compare with what you found in part a?

e. Use `pwelch` to estimate the PSD for your down-sampled time series. Note that the frequency scale is different from before because the Nyquist frequency has changed.

f. Down-sample your template too, and repeat the matched filtering (section 4) with the down-sampled data and template. The signal should stand out just as clearly as before.