BEACON Calibration Summary

Kaeli Hughes

November 2018

**Getting Started:**

BEACON analysis relies on one of two analysis packages:

1. Ben Strutt’s BeaconTau data analysis package. This works with Python 3. Installation instructions are available at <https://github.com/beaconTau/BeaconTau> . This requires “import BeaconTau as bt” at the top
2. Cosmin’s beaconroot package. This uses root and Python 2, although could also use Python 3 depending on your root installation. Installation instructions found here: <https://github.com/beaconTau/beaconroot/blob/master/examples/beacon_data_reader.py>

For a single correlation event, either method is equally fast. For 40,000 events, I’ve found it’s quickest to use Cosmin’s tools. Make sure to replace the directory I have that locates the directory with the correct path for your own machine/setup.

**Various Programs I’ve Written:**

* generate\_correlator.py: this is the simplest program of the bunch, and the one that is probably the most user friendly. Currently it is set up to use Ben Strutt’s analysis package. It generates a correlation map like the one Cosmin has on the website. In the “main” section of the code, change the run number and event number variables and a map should pop out.
* SunFinder.py: This was my first pass at a sun-centric coordinate system. Not the most up to date at this point, but I’ve included it in case you’d like to take a look. Currently defaults to Cosmin’s analysis package.
* GalacticFinder.py: This is the most recent program that takes data and plots it in Right Ascension (RA) and Declination (Dec). In the main{} section, you can choose between various settings like: looking at a single event vs evenly sampled events, run number, event number, and polarization
* CreateCorrelationFiles.py: For a given run, generates \*.npy files that contain correlation waveforms and best fit time delays. Can use either analysis package.
* Timing.py: Uses files created with CreateCorrelationFiles.py to cluster events with similar timing delays, then plots them. Useful for creating a catalogue of events. I created this a few weeks ago and haven’t revisited it, though I think there’s a lot of interesting information here.
* Fake\_Correlator.py: takes a fake simulated signal (saved as ‘run191\_pulse126\_average.npy’, which was created with Timing.py) and compares it to a real event.

**Next Steps:**

Here are the things that I would do next, if not for the upcoming South Pole trip:

1. The 48.36 MHz signal. As Cosmin showed, this signal pops up somewhat periodically and is incredibly strong. It mostly looks like CW and I’m pretty sure at this point that it’s the LA Department of Water. Here are the links that have the locations of all the towers in the area:
   * <https://www.radioreference.com/apps/db/?fccCallsign=KJZ563> (Nevada towers)
   * <https://www.radioreference.com/apps/db/?fccCallsign=KXU627> (California towers- Cosmin thinks Tower 1, the Cottonwood Repeater, is the most likely based on splat)

I would like to take the locations of these towers, calculate the expected time delays with respect to BEACON, and compare that to the calculated time delays of BEACON. Even though it’s CW, the wavelength is around 6.2 m, so hopefully there is only 1 answer that makes sense. This could then be used to calibrate the antennas much faster than the galactic center.

1. Improvements to the GalacticFinder python program:
   * Right now it takes about 2-3 hours to run over 40,000 events, which is pretty slow. I think this could be improved in a few ways:
     + Combine the FFT in the filter with the one in the Correlate function.
     + Create all the correlation waveforms and save them to a file, then reference those each time.
   * Once it’s a little faster, run over all events in a 24-hour period (which is ~400,000). Not sure if this would make the map look better or not.
   * Design a regression algorithm that tries placing the antennas in different locations, re-runs the correlation map, and maximizes the value near the Galactic Center.
2. Creating a catalogue of pulses that we can cut on. I’ve started this with the last 3 files listed above, but I think it’s worth creating an extensive catalogue of the pulses that we’re seeing again and again. Some of them are showing up >10,000 times in a run and I feel like it’d be really good to have those recorded.