All scripts support option -h for help.

1. **find_baseband_runs.py**
   A script that trolls through logs directory and prints contents of log files between two specified dates. Needs a directory that contains following folders:

   albatros_config_fpga albatros_dump_baseband albatros_dump_spectra

   -s start_date -S stop_date (YYYYMMDD)

   ```
   python find_baseband_runs.py ~/logs/snap1/ -s 20210729 -S 20210730
   ```

2. **quick_spectra.py**
   A script to plot direct spectra products (pol0 mag, pol1 mag, pol0xpol1 mag and phase) for any given direct data folder.

   Needs a directory that contains the following files:

   pol00.scio.bz2 pol01i.scio.bz2 pol01r.scio.bz2 pol11.scio.bz2

   Ideally, the directory name should be a timestamp (as is the convention for albatros data dumps), so that the output image has the timestamp in its name. This timestamp indicates the starting time for this direct data file. FPGA code is set to create a new file roughly every hour. Usual directory structure is something like:

   ```
   ~/data_auto_cross/snap1/16272/1627202093/
   ```

   Use -l (ell) for logscale plots, which are more useful. -o for specifying output dir. Default output dir is ./ (dir from which code is called).

   ```
   python quick_spectra.py ~/1627202093/ -o ~/outputs/ -l
   ```

3. **plot_overnight_new.py**
   A script to plot direct spectra products over several hours (combining multiple files). It will generate a single plot showing what the data looks like for an entire night's (or several nights') run. It takes a start timestamp and end timestamp and automatically finds all files that lie between these timestamps. (Timestamp format YYYYMMDD_hhmmss)

   Needs a top-level directory that contains sub-directories with "short" 5 digit timestamps. (Each of these sub-dirs has sub-dirs with full 10 digit timestamps.) E.g.

   ```
   ~/data_auto_cross/snap1/
   ```

In a particular direct data file, each data point (for, say, pol00) is averaged over roughly 6 seconds by the FPGA. This may be too high a resolution if you're plotting for several hours. Use **-a [avglen]** to specify how many points to average over.

```
python plot_overnight_new.py ~/data_auto_cross/snap3/ 20210730_000000
20210730_070000 -o ~/outputs/ -a 50 -l
```

The above example calculates direct data products averaged over 5 minutes (50 points) for data collected between Jul-30 12 AM to Jul-30 7 AM. And once again, use -l for logscale plots.

It is the user's responsibility to make sure that a contiguous chunk of data is available between the two timestamps. If there's some discontinuity (e.g. data wasn't collected for 2 hours in the 7 hour window), it'll be obvious from the plots. In this sense, this is a useful diagnostic tool.

4. **Plot_hist.py**

   First switch to branch **newcode.**

   ```
   git checkout newcode
   ```

   If there are no compiled ".so" files in the correlations directory, run:

   ```
   python ./correlations/setup.py
   ```

   (You should still run this if you have some old version of .so files lying around.)

   This script allows you to histogram the whole baseband file. You can specify if you want to histogram pol0, pol1, or both (entire data) using option **-m [mode].** (mode=0 for pol0, 1 for pol1, -1 for both. Default is -1.)

   Default output plot will have counts for each bit value, e.g. 0 - 15 for 4 bit data. You can shift the bins and their values to reflect the positive and negative levels using option **-r.** E.g. -7 to 7 to 4 bit data.

   ```
   python plot_hist.py ~/1627202093.raw -m -1 -o ~/outputs/ -r
   ```

5. **autocorravg.py**

   First switch to branch **newcode.**

   ```
   git checkout newcode
   ```

   If there are no compiled ".so" files in the correlations directory, run:

   ```
   python ./correlations/setup.py
   ```

(You should still run this if you have some old version of .so files lying around.)

The purpose of this script is to let you analyze 4 bit baseband data from a single antenna. It will compute and save pol00, pol11, and pol01 averaged over whatever acclen user specifies.

It requires a parent directory (which has 5 digit timestamp folders) for baseband files, a starting timestamp, and an accumulation length. By default it will run for 560 chunks: that's roughly 1 hour for an accumulation length of 393216. You have the option to manually specify the number of chunks, or an end timestamp. In the latter case, the number of chunks will be calculated using (t_end - t_start)/t_acclen. Both cases are shown below.

You also have the option to control exactly what channels you'd like to examine. In order to specify the channels, use the "-c" option followed by the **indices (not channel numbers)** of starting and end channels. **The channels must be contiguous and the indices must follow numpy convention.** E.g., say you have 100 channels around 40 MHz, and 50 channels around 113 MHz (ORBCOMM), and you'd like to examine ORBCOMM, use: **-c 100 150**

If you wanted to look at a single channel, you'd use something like: **-c 100 101.** This will only unpack the channel at index 100.

Example usage:

```
python autocorravg.py ~/baseband/snap1/ 1627202094 393216 -t 1627205694
-c 100 150
```

```
python autocorravg.py ~/baseband/snap1/ 1627202094 393216 -n 560 -c 100
150
```

How to read the .npz file that's generated?
The npz file stores data and masks for pol00, pol11, and pol01, and the exact channel numbers that were used (as per the indices specified). It can be read as follows.

```
with np.load("~/pols_4bit.npz") as npz:
    pol01 = np.ma.MaskedArray(npz['datap01'],npz['maskp01'])
    # pol00 = np.ma.MaskedArray(npz['datap00'],npz['maskp00'])
    # pol11 = np.ma.MaskedArray(npz['datap11'],npz['maskp11'])
    channels = npz['channels'].copy()
```

The format of the output file is:
`pols_4bit_{time_start}_{acclen}_{nchunks}_{chanstart}_{chanend}.npz`

By default chanstart=0 and chanend=None, in order to process all channels. So if a file is named `…*0_None.png*` it means all channels in the baseband file were included.

6. **autocorravg1bit.py**

   Same as **autocorravg.py** but for 1 bit baseband files. **Important distinction:** the starting index, if you specify a range of channels using "-c", must be **even.**

   Example usage:

   ```
   python autocorravg1bit.py ~/baseband/snap1/ 1627202094 393216 -t
   1627205694 -c 24 37

   python autocorravg1bit.py ~/baseband/snap1/ 1627202094 393216 -n 560 -c
   24 37
   ```

7. **pol01_compare.py**

   This script is meant to compare pol01 real and imaginary parts produced from baseband averaging and corresponding direct file for quick sanity check.

   For 4 bit data, it plots magnitude, phase, and real/imag residuals. For 1 bit data, it plots real/imag parts and real/imag residuals.

   Accumulation length is inferred from file name, so don't mess with default file names.

   Sample usage:

   ```
   pol01_compare.py
   ~/path_to_baseband_npz/pol01_1bit_1627528594_393216_560_0_None.npz
   ~/path_to_direct_folder/16275/1627528594/
   ```