How to Calibrate and Use the ATA-20 Beamformer: Version 2.2

Sofia Sheikh, Wael Farah, Grayce Brown, Miro Saide, and the ATA Team

May 2023

1 Background on the Allen Telescope Array

The Allen Telescope Array (ATA) is a 42-element radio array located in Hat Creek, California (about 5 hours drive north of San Francisco). Each dish in the array has a diameter of 6.1 m, and it currently can record data from 1–11 GHz with the log-periodic "Antonio" feeds that are at the focus of each dish. This is an extremely wide bandwidth, and we can currently digitize 2×672 MHz of it at a time, with those 2 tunings being independently selectable within the band. This will likely increase to 3 tunings some time in Summer 2024 (and 4 tunings at some unspecified date after that).

2 Interferometry Basics

Interferometers operate in a few basic modes, all built upon ways to combine the signals being captured by the multiple elements in the array. Here, we will talk about the two most common modes: the **correlator** mode and the **beamformer** mode.

The **correlator** captures what is known as "visibility" data — frequency, time, polarization and flux data for each *baseline* (pair of antennas in the array) by cross-multiplying each baseline in the array. This visibility data can be used to correlate the beamformer mode (which is what we will mostly talk about here) but it is also scientifically interesting in its own right. For example, you can grid the visibilities in a matrix and perform a 2D inverse FFT to get a sky image which can be used for science of extended sources. The key to using correlation for calibration is that the cross-multiplication allows us to measure the instrumental time delay imparted by different parts of the system — cables, attenuators, amplifiers — such that we can calculate and then correct for the differences in phase. We will talk more about this later in Section 4.4!

The **beamformer** captures a more familiar data product to single-dish observers: time, frequency, polarization, and flux data corresponding to a particular point (or points) on the sky. In the beamforming mode, the signals from each antenna are delayed with respect to one another (based on the results of the calibration) to "phase up" signals corresponding to user-specified locations of synthesized beam positions. This allows the beam to be electronically steered to any location on the sky (although you'll want to take care to use only locations in the direction that the telescope is actually pointing/sensitive!).

3 Introduction to Observing

In order to perform observations with the ATA, you'll have to log onto the observing VNC session, which is a single universal session for all observers. For current login, port, and password information, or help setting up your VNC software (e.g., VNCViewer), please contact an ATA team member.

The VNC session will have several "workspaces" or desktop screens, with different useful terminals, GUIs, and data displays on each. The exact setup can change over time, but usually the delay engine terminals, antenna status monitor, obscontrol terminal (for checking e.g., if a source is up), and terminal for running observing scripts are on Workspace 1. The pipeline monitor and terminal to set the backend/post-processor are on Workspace 2. And the terminal window to run the flux calibrator pipeline with CASA is in Workspace 3. An example of Workspace 2 is shown in Figure 1.

However, don't worry too much about each of these individual components for now. What's most important is to know how to switch between the workspaces, which can be done with the buttons on the top

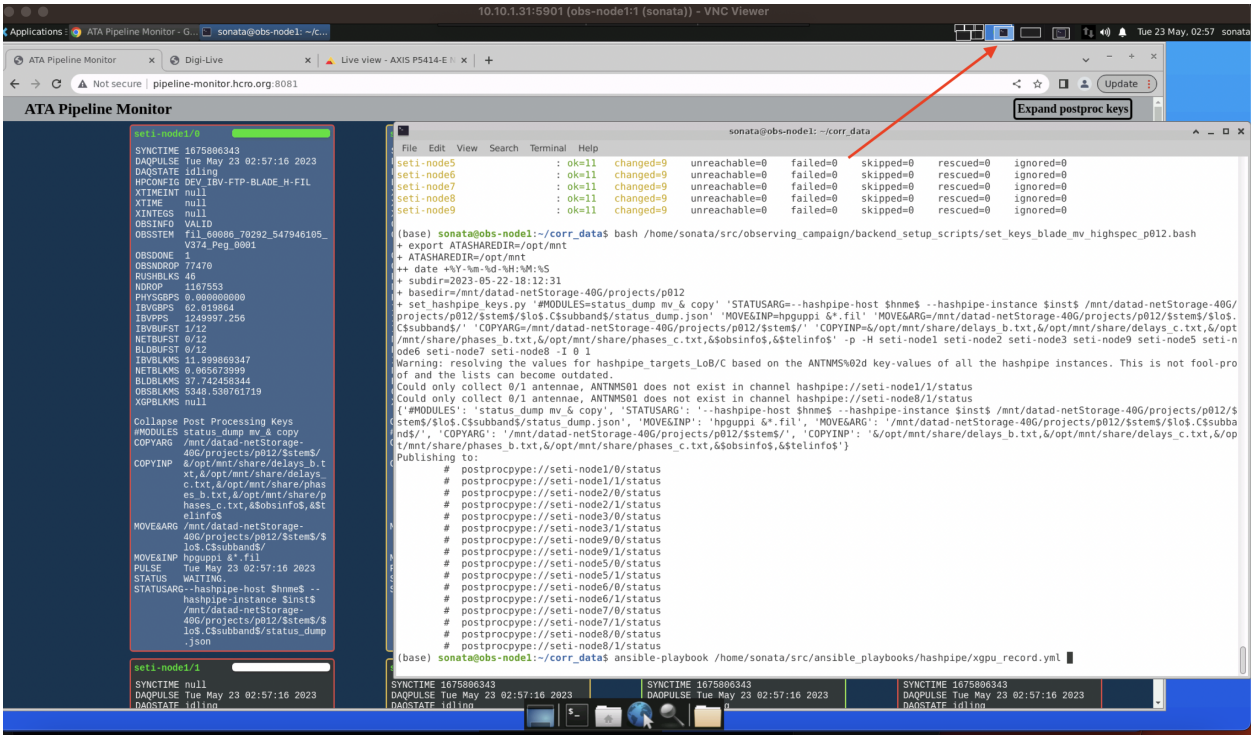

Figure 1: An example of one of the VNC workspaces (in this case, workspace 2). The red arrow indicates which workspace we're in, which can be changed by clicking those buttons at the top right of the screen. New terminals can be opened by clicking the black and white icon at the bottom center of the screen.

right in Figure 1 (red arrow pointing to current workspace), or by using the shortcut Ctrl-Cmd-[arrow key] to move left or right.

Make sure no one else is using the VNC (which allows multiple simultaneous users) by asking in the #active-ata-observations channel in Slack, and pausing for a minute to see if anyone else is typing in terminal windows or moving the mouse around.

In general, if anything seems unusual or broken during the observing process, post the issue you're having to #active_ata_observations and an ATA team member will likely be able to help you out in real time.

When running an observation with the beamformer, we must first *delay* and *phase calibrate* the beamformer, then we can run our *observation script* in Python. Funnily enough, the phase calibration has a lot more steps than most actual post-calibration observations!

4 Calibrating the Beamformer

Quasars are extragalactic objects: supermassive black holes at the center of distant galaxies. Conveniently for us, they are also bright, steady, broadband flux sources, somewhat like reliable beacons in the sky. It's mildly amusing that we use such immense, mindblowing objects for the humble step of beamformer calibration.

Calibrating the beamformer with an observation of a flux calibrator source (i.e., a quasar) has the following steps:

1. Load correlator backend and post-processor
2. Perform observation of flux calibrator
3. Copy calibration data to a local directory

4. Run auto-calibration pipeline
5. Apply calculated phase solutions
6. Re-observe the flux calibrator
7. Get System Effective Flux Density (SEFD)

4.1 Load correlator backend and post-processor

Before we start, it's worth understanding the difference between a **backend** and a **post-processor**. The **backend** is the part of the observing pipeline that defines which mode you're using to take data (e.g., correlator or beamformer). When running an observation, your files will write, by default, to the `seti-nodes` that are in charge of taking the data. However, the `seti-nodes` are not meant for long-term storage. If you only set the backend, therefore, the `seti-nodes` will fill up with data and then run out of memory, breaking the rest of your observation. To avoid this outcome, we must also set at least a basic **post-processor**. The post-processor defines any steps that are done after the data is taken on the `seti-nodes`, for example, critically, moving the data off of the `seti-nodes` and into long-term network storage. Usually, the post-processor will be written to put your data into your project directory (p###); for the correlator, the data will be transferred into a directory called `corr_data` (for now in the `datac` storage directory, though this is subject to change). However, for non-calibration observations, this could include automatically running analysis software like `turboSETI` (for... SETI) or `heimdall` (for FRBs).

Great: let's give it a go! For this step, you'll want to move to the workspace with the "ATA Pipeline Monitor" window (Firefox page with dark blue background). Then, you'll need to activate the correlator backend with the following command as `sonata@obs-node1` (usually in the existing terminal window in the same desktop).

```
ansible-playbook /home/sonata/src/ansible-playbooks/hashpipe/xgpu_record.yml
```

This command is used often, so you can also usually get to it by hitting the up arrow to see previous commands if you don't want to type it out. However, if you do this, make sure the command contains `xgpu_record.yml`, because that's the correlator-specific backend (the other backends look similar, but are not the same).

You'll see a bunch of things print in the terminal window (starting with `PLAY` and `TASK`). This step takes about a minute to run, so be patient! At the end of the command, you'll see it print `PLAY RECAP` — here, just make sure that all of the lines say `failed=0`. This shouldn't fail, though (famous last words).

So now the correct backend is set for calibration, but we also need to set the correct post-processor for calibration. We can check the post-processor by clicking `Expand Post-Processing Keys` in any of the SETI node boxes on the ATA Pipeline Monitor window (it's a clickable piece of text, even though it doesn't look like it). If it says `#MODULES skip`, then there's no post-processor loaded. This is bad — we'll want to add the correct post-processor for correlation.

To do this, enter the following command (same terminal window):

```
/home/sonata/src/observing_campaign/backend_setup_scripts/set_keys_uvh5_mv.py
```

Remember, this is for correlation observations *only*. There are different post-processors for FRB observations, SETI observations, pulsar observations, etc. We will talk about these other options later (in Section 5).

After you run that command, you can check a few status options in any of the `seti-node` boxes in the ATA Pipeline Monitor window to make sure it worked, except nodes 1.1 and 8.1, which will always show NULL. Those two `seti-node`s are not currently used to record data (because there are only 7 nodes needed in each band, leaving an odd node out for LOb and LOc). If you really want to check on the metadata keys in detail, you can use the following items as sanity checks:

- DAQPULSE is steadily increasing (showing active connection to the machines)
- HPCONFIG ends in *...xGPU-UVH5* (showing that visibility data will be saved in a .uvh5 file format)
- IBVGBPS (input GB/second) is about 60 (ensuring that the data rate from the `seti-node`s into the network interface is normal)

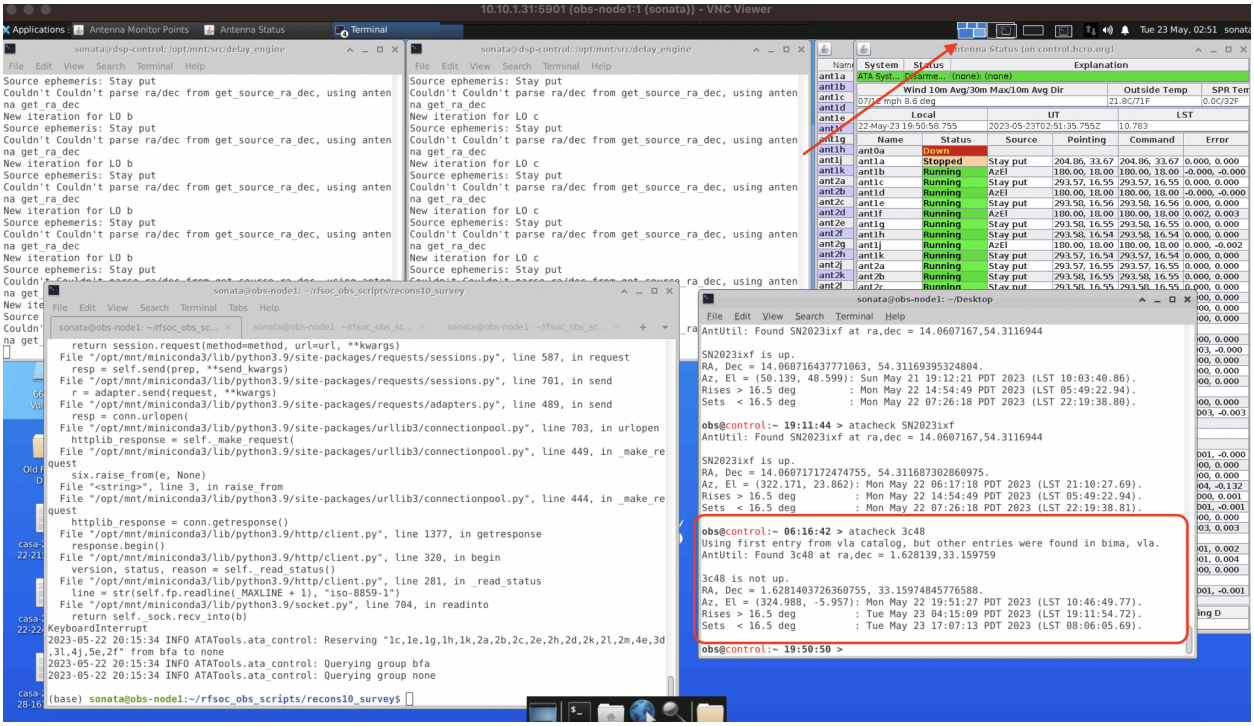

Figure 2: An example of Workspace 1 (indicated with the red arrow), showing a source check on 3c48. As printed in the output in the red box, the source was not above the observing elevation limit at the time of the command.

- IBVPPS (input packets/second) is about 1.24 million (same data rate check as above; a lower value would indicate an issue with the network or digitizers)
- OBSNDROP (number of dropped packets in the observation) is 0 — for the beamformer this number actually increases at the start of observation during normal operation, which is fine unless the value reaches above the millions

Finally, if you notice that the PROCESSORS key has errors ("Error: Null"), this is totally normal. In a recent update, the number of workers that the post-processor deploys was expanded from 1 to 4, to be able to process up to 4 simultaneous finished observations in parallel.

Alright, the correlator backend and post-processor are now loaded and we're ready to observe!

4.2 Perform observation of flux calibrator

For this step you will go back to the workspace with the antenna status panel and work in the window which says `sonata@obs-node1`. Make sure you're in the `rfsoc_obs_scripts` directory.

Now let's pick our flux calibrator (quasar). In summer, the best calibrator quasar that is up during the day is 3c48, and the best ones that are up at night are 3c286 and 3c147. You can check if a source is up by typing, e.g., `atacheck 3c48` into the `obs@control` window, as in Figure 2. This will also provide you with the RA/Dec coordinates of the source, the current azimuth and elevation of the source, and, importantly the rise and set times of the source.

> NOTE: If you want to be able to normalize and properly flux calibrate your observation, you will need to use one of the calibrators from Perley and Butler [2017]. The following calibrators from that paper are both observable by the ATA and present in the ATA source database:
>
> - 3C48

- 3C123
- 3C138
- 3C147
- 3C196
- 3C286
- 3C295
- 3C353
- 3C380
- Cassiopeia A

Do not use 3C84, as it is not actually steady enough to be considered a flux calibrator! If, for some reason, you have already made this mistake (e.g., if you took data in Summer 2023 or before, when the source was accidentally in the catalog), contact the ATA team for calibration workarounds.

We will want to run a script for a correlator observation corresponding to the calibrator we choose, e.g. `observe_3c48.py`. There are two options for how to create this script:

1. Edit an existing script in the `rfsoc_obs_scripts` directory
2. Use the script in `/home/ssheikh/Observing_Tools/calibrator_script_generator.py` to make an observing script for a particular calibrator. Warning: do not use, Sofia needs to update!

Either way, we'll need to make sure the script we choose is tuned for our needs. Open up the script with a text editor (vi/nano/emacs/etc.), and check the following things:

1. Make sure the `source` variable is set to the correct string (matching the object's name in the ATA catalog; generally starting with '3c' lowercase)
2. Set `freqs` and `freqs_c` to the center frequencies you want for your low and high frequencies, respectively (for your main observation). LOb (freqs_b) should be set to the lower frequency, and have `nofocus=True` in the `set_freq` call. LOc (freqs_c) should be set to the higher frequency. Remember that you'll get a 672 MHz chunk of spectrum from each LO, centered on the LO frequency you input here.
3. Make sure the `autotune` and `snap_if.tune_if_ants_lo` lines are uncommented for calibration, and commented out otherwise.
4. Set the integration time in seconds (`XTIMEINT=30` for 30 second integrations) in the definition of the dictionary `d`.
5. Check that the recording will start with `hpguppi_record_in.record_in` near the end of the script.
6. Set the total recording time (`obs_time`). By default, the calibration observation script records for 10 minutes (600 seconds) — this is generally appropriate for calibrating on 3c sources, but would be far too long an observation for a maser line or Mars technosignature observation (which would be $\sim$ 1 minute).

If all of that looks good, run the calibration script with e.g.,
`python observe_3c48.py`.

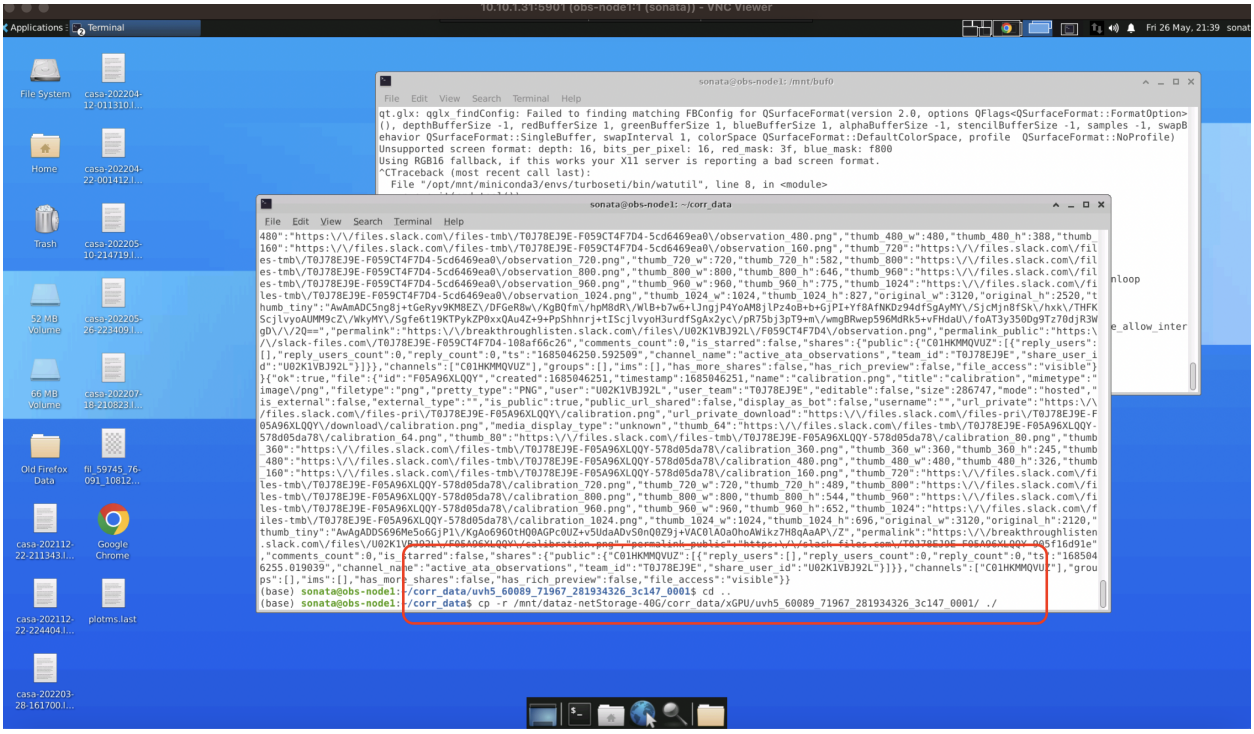

Figure 3: An example of moving and copying calibrator data in the obs-node1 terminal on the third workspace on the VNC.

4.3 Copy data

Once the observation ends, the data should end up in `/mnt/datac-netStorage-1G/corr_data/xGPU/` (for now, this will be moved to a location in `/mnt/primary/` sometime in February 2024) and start with `uvh5` (denoting UV-plane visibility data stored in h5 format). On `obs-node1`, usually in a dedicated terminal in Workspace 3, we want to move this data elsewhere with the following commands:

- `cd ~/corr_data`
- `cp -r /mnt/datac-netStorage-1G/corr_data/xGPU/uvh5_XXXXX... ./`, where you determine which folder you want by looking at the ObsID string in any of the nodes shown in the Pipeline Monitor (XXXXX will always be today's MJD). Make sure the source name matches too. Example command: `cp -r /mnt/datac-netStorage-1G/corr_data/xGPU/uvh5_59752_75727_23852050_3c84_0001./`
- Now move into that folder that you just copied with `cd uvh5_XXXX_....`

An example of the terminal in Workspace 3 is shown in Figure 3.

4.4 Run calibration pipeline

To run the calibration pipeline, all you have to do is run `~/scripts/auto_calib_casa.bash` . This script will print a lot of outputs and take a few minutes to run — this is normal. Generally, the script is splicing the 7 sub-bands of each node together; then flagging radio frequency interference (RFI); then normalizing by autocorrelations to properly measure sensitivity; then solving for the best phase and delay solutions; then producing diagnostic plots to show the efficacy of those solutions.

Then, a complicated plot will appear in the VNC, an example of which is shown in Figure 4. Details on how to read this plot for observational purposes are provided in the caption to Figure 4 — for more details on the actual physics, see Section 8 but briefly, a successful calibration would have all but a few of the 40 sub-sub-plots on the lefthand side showing flat orange and purple lines overlaid around 0.

Figure 4: An example of the 4 sub-plot, 80-sub-sub-plot "calibration check" figure that appears after running the `auto_calib_casa.bash` pipeline. You want all of the sub-sub-plots on the lefthand side to have orange and purple lines that are flat-ish and aligned at 0 phase over frequency — these are the post-calibration "best-fit" results, implying that there are now no phase offsets across the bandwidth. The top two sub-plots are for LOb, and the bottom two are for LOc. It's fine if some of the sub-sub-plots show gaps (from radio frequency interference, or RFI) or if the orange and purple (x and y polarizations) are not perfect mirrors due to RFI, you just don't want wrapping diagonal lines or vertical offsets. The bottom right panel of every subplot is removed, as this is the auto-correlation panel and would provide only redundant information (see Section 8). A missing panel indicates an antenna that is down ('1b' in this example) — note this in the channel before continuing, if present. In addition '1e' shows a large phase scatter in LOc after calibration due (in this case) to an uncorrected 2 ms delay — if you see this, you can try the steps in Section 8.7. Despite these two antennas showing irregularities, this is an example of a successful, and necessary, calibration.

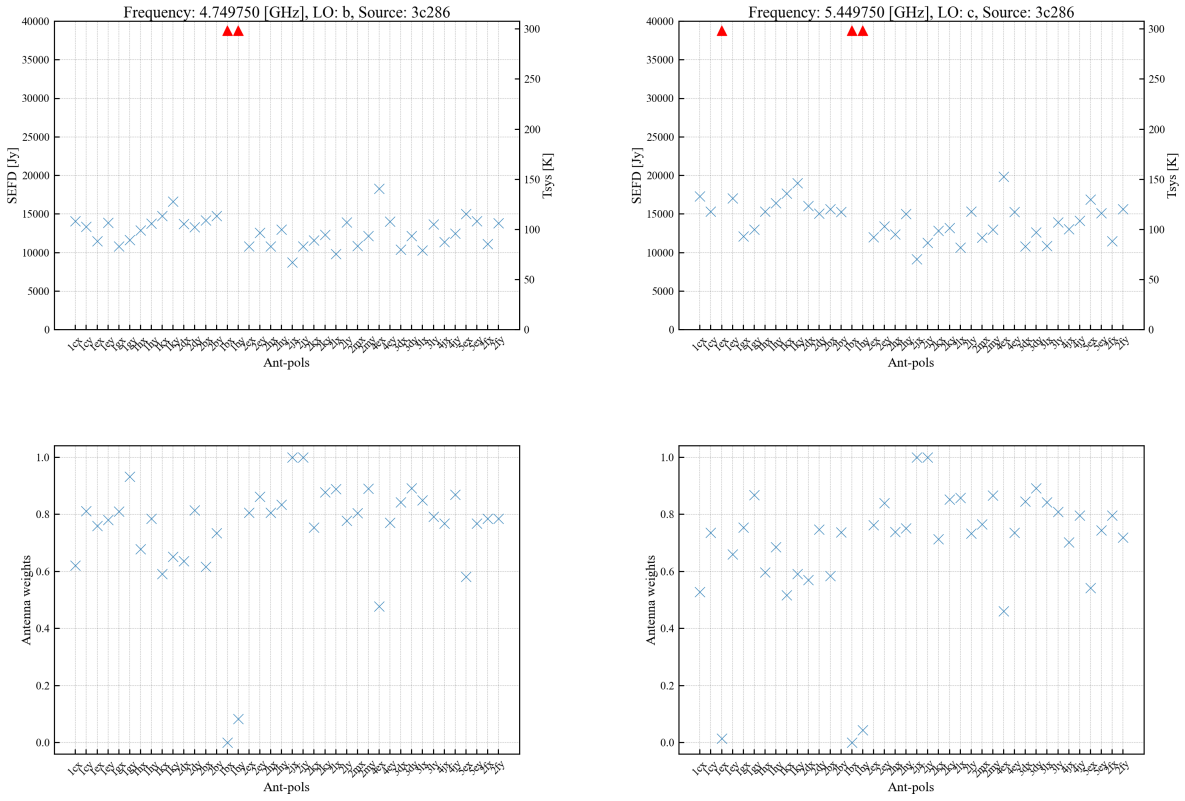

Figure 5: An example of the calibration results figure that appears after Figure 4. The top row (LOb on the left, LOc on the right) is the calculated SEFD (a measure of sensitivity) for each antenna-polarization combination. Ideally, all of the points are blue x's that are between 5000–15000K — more sensitive antennas will show lower values of SEFD. At RFI-heavy frequencies or frequencies above ∼8 GHz (as in this example), you might see these values increase. Sometimes, you'll see red arrows indicating that a particular ant-pol is off of the axes. If it's just a single antenna's worth (2 adjacent arrows) that's probably fine — any more than a few and you should investigate. The bottom row of plots shows something that should look like the inverse of the top row - these are the antenna weightings based on the SEFDs in the top row. In this plot, larger numbers (closer to 1) are indicative of the best antennas with the most sensitivity. This is an example of a successful calibration at a frequency that is in the lower sensitivity part of the ATA's frequency coverage.

When you close out of that plot, you will see another plot appear, an example of which is shown in Figure 5. Details on how to read this plot are provided in the caption to Figure 5, but briefly, we would like the SEFDs in both LOs to be between 5000–15000 Jy for a low frequency observation; it is fine if antennas stray from these values, but you should consider a lower frequency (or debugging) if you see more than a few red arrows in the upper plots. Close this plot as well.

4.5 Apply calculated phase solutions

To apply the phase solutions calculated by the pipeline, all you have to do is run `~/scripts/apply_solutions.bash` . This also pushes the plots to the #active_ata_observations Slack channel for documentation and live feedback if needed.

4.6 Optional: Re-observe the flux calibrator and run calibration pipeline

Now that the system is properly calibrated, we could do one more correlator observation of a quasar (we can pick the same one) to get some slight additional refinement and a reliable measurement of the System Effective Flux Density (SEFD) value. This is most important for large SETI surveys or surveys that care strongly about absolute flux.

In this observation, we can use the same source, frequencies, and observing time, but make sure to *remove the autotune and IF-tune* from the observing script (if either are removed, the `set_freqs` call will also be removed). Re-run the correlator observation with the same `python observe_3c84.py` -style command, and then re-run the step in Section 4.3 for the new directory.

Make sure you're in the same directory that you copied over, then run the command from Section 4.4.

5 Running an Observation with the Calibrated Beamformer

5.1 Load in the backend you want

For any beamformed observations, we will be using the Breakthrough Listen Accelerated DSP Engine (BLADE) beamformer under the hood. This software was developed by ATA team member Luigi Cruz. Detailed notes on each beamformer mode are contained in the Appendix:

5.1.1 SETI Backend - real-time, high-resolution beamformer

The correct backend to load for most SETI observations will be:

`ansible-playbook /home/sonata/src/ansible_playbooks/hashpipe/beamformer_mode_h_record.yml`

You should usually run this command in Workspace 2, in the same terminal that you ran the backend/post-processor commands from the calibration.

When you input this command, it will ask you for the number of beams. For most small projects, this will be 2: one on-source and one off-source. Then you will be asked about the polarization options: 1 will provide total intensity (should be the choice for most projects as of Q1 2024), while 4 will provide 'xx', 'yy', 'xy', and 'yx' products. It will then ask about the channelizer: most project will want to select 1 for "disabled" for now. If you think you want additional channelization, run it by an ATA team member. Next, you'll be asked for an accumulation rate N_{int}, which is a proxy for the channel size of your observation $\delta\nu$ as follows:

$$\delta\nu = \frac{1}{\Delta\nu^{-1} N_{block} N_{int}} \tag{1}$$

Where $N_{block} = 8192$ samples for a non-channelized observation, which corresponds to a $\Delta\nu = 0.5$ MHz.

An input N_{int} value of **"61"** will give you 1 Hz channels, and is our default for SETI searches. Similarly, the backend setup will then ask for a time integration factor — with channelization disabled, an input value of **16** will provide 15 second integrations, which is our current SETI standard. Finally, you will be asked whether you want output files produced in SIGPROC or FBH5 format — FBH5 is the default for SETI searches (default as of Summer 2023). The backend should then complete its loading process without additional queries.

5.1.2 FRB/Pulsar Backend

The correct backend to load for most Fast Radio Burst (FRB), pulsar, or other transient/high-time resolution observations will be:

`ansible-playbook /home/sonata/src/ansible_playbooks/hashpipe/beamformer\_mode\_a\record.yml`

You should usually run this command in Workspace 2, in the same terminal that you ran the backend/post-processor commands from the calibration.

You will be asked to select the number of beams: for most FRB/pulsar studies, 1 beam will be sufficient. Then, you will be asked whether you would also like to produce an incoherent beam (summed but not

phased), which provides sensitivity across the whole field-of-view. This option is not recommended for most projects, and you should answer "False". Polarization options are the same as in Section 5.1.1: 1 for total intensity data, 4 for full polarization data. Channelizer options are also the same as in Section 5.1.1: 1 is disabled, and should be correct for most projects. If you do disable the channelizer, then the next input, which asks for an integration size N_t, can be converted to your actual sampling time with the following equation

$$t = N_t * 2\mu s \tag{2}$$

Where t is the sampling time in μs. For example, a user input of $N_t = 32$ will produce a sampling time of 64 μs, which is the standard time resolution for ATA pulsar/FRB projects. Finally, you will be asked for a file format: the standard filterbank format for pulsar/FRB projects is SIGPROC.

5.1.3 Critically-Sampled Voltage Backend

If you need voltage data from a beamformed product, consult the ATA team before using `beamformer\_mode\_b\record.yml`. This is an extremely memory-heavy mode and should be used with caution.

5.2 Load in the post-processor you want

5.2.1 SETI Post-processor

The post-processor that you use for SETI work on the ATA will likely be some variant of `set_keys_blade_mv_highspec` appended with a project number (e.g., p004), e.g.,

`/home/sonata/src/observing_campaign/backend_setup_scripts/set_keys_blade_mv_highspec_p004.py`

The new SETI post-processor primarily just serves to shift data from the buffers to a permanent network storage directory associated with a particular project. Replace the project code with your own project code and the post-processor should run correctly, assuming you have already created a post-processor script for your project. If you would like more complicated post-processing, or don't have a version of the default highspec one for your project yet, consult with the ATA team.

5.2.2 FRB Post-processor

The post-processor that you use for FRBs work on the ATA will likely be some variant of `set_keys_blade_mv_highspec` appended with a project number (e.g., p031), e.g.,

`/home/sonata/src/observing_campaign/backend_setup_scripts/set_keys_blade_mv_p031.py`

As with the SETI post-processor, the FRB post-processor just serves to shift data from the buffers to a permanent network storage directory associated with a particular project. Replace the project code with your own project code and the post-processor should run correctly, assuming you have already created a post-processor script for your project. If you would like more complicated post-processing, or don't have a version of the default highspec one for your project yet, consult with the ATA team.

5.3 Run your observing script

This part is straightforward — just use python to run your observing script e.g., `python observe_kepler842b.py`

If you do not have an observing script written, check out the "Writing Observing Scripts — ATA" guide (separate Overleaf) for guidance, or contact the ATA team.

6 Checking the Calibration at the End of the Observation

Now, once the ATA Pipeline Monitor shows that the post-processing is finished (by saying "idling", usually takes less than a minute for a simple data-moving post-processor) you may want to do one more calibration of your flux calibrator by following the steps in Section 4.6. This is more critical for large surveys, greater than 6 hour observations, or projects that require very accurate fluxes.

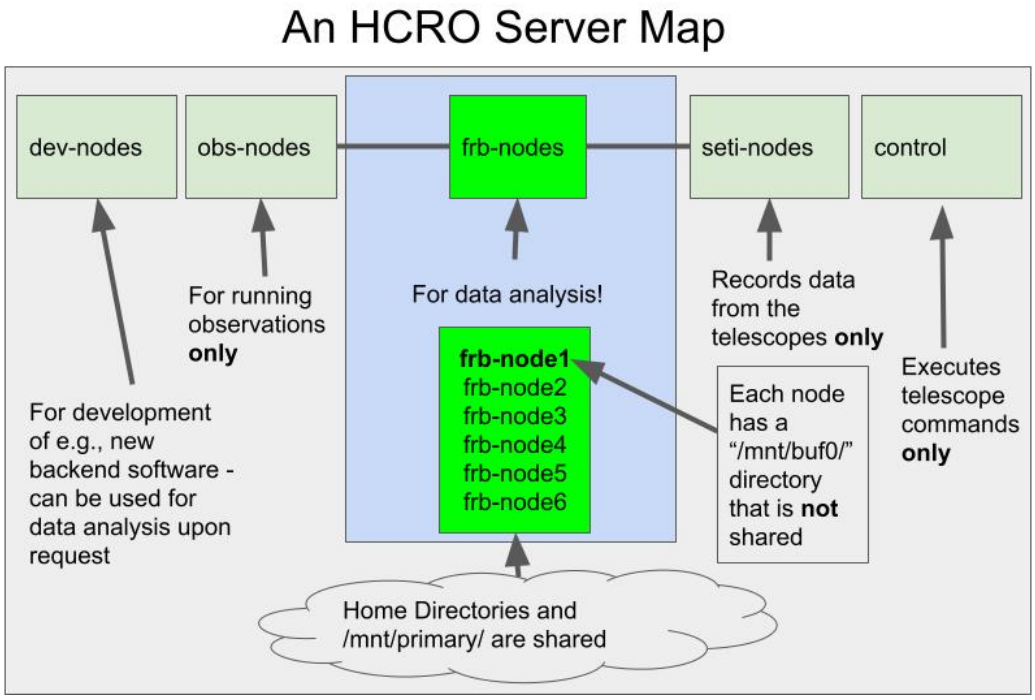

Figure 6: Structure of the HCRO server ecosystem.

7 After the observation: Compute, Data, and Storage

We are currently (February 2024) in the middle of reorganizing our network storage. Figure 6 shows the current organizational structure of the server system:

Once you have taken data, it will likely end up in a project directory in the "primary" storage directory, e.g., with a path like `/mnt/primary/projects/p040/`. If you took data before February 2024, your data may be located within sub-directories in the "primary" storage node, based on the specific data disk the data were on in the past, e.g., `/mnt/primary/datad/projects/p040/`. The original network storages will also still be available for now (`dataa`, `datab`, `datac`, `datad`, `dataw`, `datax`, `datay`, and `dataz`), but only with a -1G (1 GB/s) connection, instead of the previous -40G and -100G extensions.

For working with data, make sure to log in to any of the 6 `comp-node`[1-6] for data processing. If you would like a "scratch" area for temporary data storage, you can use `/mnt/scratch`.

8 Appendix

8.1 What the 80-sub-sub-panel calibration plot actually means

In Section 4.4, and specifically Figure 4, we provide an operational description of how to read the calibration output plot from the `auto_calib_casa.bash` pipeline. However, that description does not explain the physical intuition behind this plot: we provide that information here.

Assume that there is a series of bright, perfect sine waves at each frequency in our bandwidth being produced by our calibrator source. If you imagine this wavefront hitting the array, some dishes (that are closer to the source) will encounter the signal before other dishes. In addition, those signals have to be recorded and sent down fibers of varying-lengths to the signal processing room (SPR). Even though all of the dishes are looking at the same radiation source, both of these effects will cause the incoming radiation to arrive "out-of-phase" at the SPR. Our goal, in the process of calibration, is to re-align all of the phases to that they're adding constructively, and all of the telescopes work together to record high sensitivity data on the source.

Back to Figure 4: Each 19-panel subplot shows the phase-versus-frequency delays for 19 antennas versus the reference antenna. Here, if you zoom in, you'll see that every panel mentions "5e" — this is our current reference antenna in the 20-element ATA. Each panel, therefore, shows the phase delay between the signal at the reference antenna and the signal at another antenna, for every frequency. There is no point in showing the 20th panel — 5e correlated with itself — because we already know that it will be perfectly in-phase with itself, and that's not particularly useful information. If you see a "phase ramp," a wrapping linear slope across frequencies in one of the panels, that is indicative of some residual time delay between the two incoming signals (and a frequency-dependent velocity of analog signals in passive components of the signal chain). If you see a "phase offset," a constant offset from zero in one of the panels, that is indicative of a delay in the cables between the LO mixers and the digitizers.

8.2 How to manually run `plot_sefds`

Currently, the `plot_sefds` function is automatically run, and the output ported to Slack, within the `auto_calib_casa` function. However, if you need to run it outside of that pipeline, here's the manual instructions.

At this point, we can get the SEFD using a script, still within the same observation directory. We will run `~/scripts/plot_sefds.py` to plot out the SEFDs for each antenna. This script requires 3 arguments:

- `cal.b.G` or `cal.c.G` file (depending on the LO you want to plot): The gain table generated by the auto-calibration CASA script. NOTE: these are only scaled properly (and therefore should only be trusted) if `normalize_by_autos` was called successfully in `auto_calib_casa.bash`.
- `flux_jy`: The flux density of the source at the center frequency of the band, in Jansky. Again, make sure to use the center frequency associated with the appropriate LO. To get this value, you can use a utility script in `/home/ssheikh/Observing_Tools/flux_density_retriever_pb2017.py`, which is called with a string associated with your object name (assuming it's a flux calibrator present in the Perley and Butler (2017) flux calibrator catalog) and your center frequency in GHz as in the following example: `python /home/ssheikh/Observing_Tools/flux_density_retriever_pb2017.py ''3C286'' 6.0`
- `freq_ghz`: The center frequency in GHz of the observation, for the chosen LO — same as used for the flux density command above, used for plotting purposes.

Once you have those arguments ready, you can call the plotting script with:
~/scripts/plot_sefds.py cal.G flux_jy freq_ghz

8.3 Beamformer modes

Currently, the BLADE beamformer supports 3 separate (but fairly similar) modes. Each mode can be fine-tuned by the observer to align with the desired science case.

- `Mode A:` Produce "filterbank" files of "detected" total intensity data, either in stokes I, or full coherence polarization. The user can control:
  - Number of synthesized beams
  - Whether or not to produce incoherent sum
  - Number of output polarization (1=Stokes I; 2=XX,YY; 4=XX,YY,XY,YX)
  - Pre-beamformer channelizer FFT-length (1=no channelization)
  - Integration length. This is in unit of $t = N \times M$, where $N = 2\mu$s (i.e. the PFB sampling rate), and M is the pre-beamformer channelizer provided above.
  - Filterbank output format; `SIGPROC` is default and is the format needed for the `dspsr` and `heimdall` post-processors.

  Use this mode for pulsar and FRB observations.

- `Mode B:` Produce "GUPPI" files of beamformed voltages. The user can control:
  - Number of synthesized beams
  - Output bit rate (4=cfloat16; 8=cfloat32)
  - Pre-beamformer channelizer FFT-length (1=no channelization)

  Use this mode for any offline processing requiring complex beamformed voltages. Offline SETI is one example.

- `Mode H:` Produce "filterbank" files of "detected" total intensity data, either in Stokes I, or full coherence, at a high-spectral resolution. The user can control:
  - Number of synthesized beams
  - Number of output polarization (1=Stokes I; 2=XX,YY; 4=XX,YY,XY,YX)
  - Pre-beamformer channelizer FFT-length (1=no channelization)
  - Post-beamformer channelizer FFT-length. This is a unit of 8192. E.g. a value of 61 will make the backend perform a $61 \times 8192 = 499,712$ point FFT, resulting in a 0.5MHz/499712 $\sim$ 1Hz resolution output file
  - Integration length in units of $1/f$, where f is the spectral resolution calculated above.
  - Filterbank output format; default is `FBH5` which is needed for `turboseti` and `seticore`.

  Use this mode for SETI and maser line studies.

8.4 How to switch out an antenna in the active array

[TBD]

8.5 Setting up the Observer's Room monitors

The screens in the Observer's Room can be accessed by either plugging in a mouse and keyboard into the computers running them, or by using a VNC.

The screen in the corner that is oriented vertically shows the pipeline monitor. It can be accessed at the VNC Server address `obs-screen0.hcro.org`, using the sonata password. If this monitor needs to be refreshed, log into the VNC, and simply refresh the browser page. It should be at the address `pipeline-monitor.hcro.org:8081` . Expand Post Processing Keys for the first row of cells.

The two screens in landscape orientation on the other side of the room are on the computer attached to the wall behind the screen on the left. They can be accessed at the VNC Server address `obs-screen1.hcro.org`, using the sonata password. The screen on the left shows the Grafana Cryo Health dashboard. The screen on the right shows the Digi-Live monitors, the Antenna Status monitor, and the Room Alert temperature monitors.

To set up the Grafana Monitor, open a browser and go to `grafana.hcro.org` and log in as sonata. Open the Cryo Health dashboard.

To set up the Digi-Live monitors, open `10.10.1.31:9000/1` and `10.10.1.31:9000/2` in browser windows. If they don't load, run the following commands [Where are these commands run?]:

```
sudo su gsingh
stopdigilive
startdigilive
```

[This may change soon?]

To set up the Antenna Status monitors, log into a terminal window using `ssh obs@control -X -Y` and use the comand `atastatus`. The Antenna Status and Antenna Monitor Points windows should pop up.

To set up the Room Temperature Monitors, [How do you set these up?]

8.6 Errors

Error: ATATools.ata rest.ATARestException: Command Failed - Status: Not Calibrated
Solution: The antennas were parked using the command *park.csh 'all ants'*, restarting the backend to *UVH* recording and restarting the post processor. The python observation script was restarted after all antennas were in thier parked position and the system worked with no errors.

8.7 Q&A

Q. In Section 4.1, it was stated that switching backends shouldn't fail. What if it does?

A. If it does, the usual culprit is a `seti-node` machine that isn't communicating with the network. In this case, a reset of the `seti-node`s has worked well in the past. TO-DO: Add the commands to do this.

Q. There are a lot of red arrows in the SEFD plot — how do I fix it?

A. 1) Try another calibration. 2) Zero your phases before starting (sometimes errors creep in over time, if the phases aren't reset to zero every now and then) — sudo bash `~/scripts/zero_phases.bash` 3) Consider the weather on site, and whether you might need to switch to another frequency.

Q. There is a single antenna-LO-polarization combination that is showing large scatter in the frequency-phase diagnostic plot across calibration (like the purple polarization in 1e versus 5e in Figure 4) — how do I fix it?

A. One possible explanation for this behaviour is that CASA is one "polyphase filterbank delay" off of the real solution for that particular antenna-LO-polarization combination. To understand what this means, it will be helpful to first read the explanation in Section 8. Have you done that? Great, welcome back.

If you encounter a single antenna-LO-polarization combo with large scatter, especially one that does not rectify with an additional calibration cycle, it could be indicative of an unsuccessful fitting by CASA. One particular common issue happens when an antenna was already misbehaving in a previous observation, leading to a bad solution. In these cases, even if the antenna has since been fixed, CASA will use the bad solution as an initial guess, and may end up in a local maximum for the "best solution", instead of the global maximum best solution. This is due to discretization in the signal chain, as described below, but is theoretically similar to picking up a harmonic instead of a primary in signal processing. The polyphase filterbank in the signal chain has a 0.5 MHz channel width, which leads to a corresponding timescale of $1/(0.5\text{ MHz}) = 2000$ ns. When CASA is already unmoored from the correct solution, it can sometimes latch onto a solution that is one polyphase filterbank delay unit away from the correct one, i.e., 2000 ns from the correct solution.

To test whether this is the issue, you can check the `delays_b.txt` or `delays_c.txt` files (whichever one corresponds to the panel where the issue is occurring). If you open that file with a text editor, you will see column-separated data where the first column indicates the antenna ID, the second column indicates the delay in nanoseconds in the x-polarization, and the third column indicates the delay in nanoseconds in the y-polarization. You should see that the lowest values are in group 1 antennas, which is because the delays were originally scaled for the old reference antenna, "1c." Remember, these delays are due to e.g., differences in the length of the fiber connecting the antennas to the control room. And, reasonably, antennas in the same group number are physically grouped on the field. So Group 1 antennas should have the lowest delays (as they were essentially normalized to zero originally), and Group 5 should have the highest delays, up to about 1500 ns, as they are furthest from the signal processing room. The difference between the x-pol and y-pol on the same antenna is usually quite small (a few to a dozen nanoseconds), as they have nearly the same signal path to the signal processing room.

So, if you see one line of the `delays.txt` file where one value is $\sim$2000 ns larger or smaller than it's supposed to be based on its group / the other polarization for that antenna, a bad CASA solve (to the tune of 1 polyphase filterbank delay) is likely responsible! You can fix this issue by simply editing the text file to remove the 2000 ns offset, and then try re-calibrating.

8.8 Notes on old backends, post-processors, and observing scripts

8.8.1 Changes to `auto_calib_casa.bash`

[Added 2022] If you looked for it, deep in a series of unnecessary print statements, you would see a line that says "normalizing by autos" — if this prints, the pipeline has normalized all of the visibilities by dividing all cross-correlations by the auto-correlations of their baselines; thus, the values are properly scaled to be able to measure the sensitivity.

8.8.2 Switch from bash to Python Post-Processors

In August 2023, we retired the `bash` post-processor and moved to a Python-based post-processor, thanks to Ross Donnachie. Before that date, post-processors were called with the following syntax:

`bash/home/sonata/src/observing_campaign/backend_setup_scripts/postproc1.0/set_keys_uvh5_mv.bash`

Which was updated to:

`/home/sonata/src/observing_campaign/backend_setup_scripts/set_keys_uvh5_mv.py`

In the new (current) Python syntax.

The Python postprocessor is more robust and generic, and should give more control and flexibility to users wanting to process data when they are written to disk. The new postprocessor scripts are executable (no need to pre-pend "python" to the call).

Ross notes that "the post-processor logs to /home/sonata/logs/postproc/*, each instance has its own file. One can tail -f one of the log files there to track the progress of a post-processor or tail -n 50 to look at the last 50 log entries. They have timestamps and are pretty verbose, so should be nice and informative."

8.8.3 Old observing script features

In the current setup, the active antenna list is queried behind the scenes, so there's no need to input which antennas are active manually. In the past, this was done with a Python list (e.g., ['1a', '1b', ...]).

8.8.4 OLD SETI Backend - offline beamformer

This is the OLD SETI Backend — we are moving to a different model now, which can do real-time, high-resolution beamforming (see Section 5.1). For documentation, the old mode was called with the following command.

`ansible-playbook /home/sonata/src/ansible_playbooks/hashpipe/beamformer_mode_b_record.yml`

8.8.5 SETI Postprocessor - OLD

Post-processor for OLD SETI Pipeline (kept for documentation): `bash /home/sonata/src/observing_campaign/backend_setup_scripts/set_keys_rawspec_longint_catch_mv.sh`

This post-processor performs rawspec to make the raw voltage data into a spectrum, then moves the newly-created filterbank data to a new directory. While this is happening, data-taking must pause for about as long as the original observation took (which is why we moved away from it).

References

Richard A Perley and Bryan J Butler. An accurate flux density scale from 50 mhz to 50 ghz. *The Astrophysical Journal Supplement Series*, 230(1):7, 2017.

Revision History

Revision	Date	Author(s)	Description
1.0	June 2022	S. Z. Sheikh	Created guide for Summer 2022 REU

1.1	August 2022	S. Z. Sheikh	A new release of `pyuvdata` incorporated the `normalize_by_autos` function in the main python distribution. The `pyuvdata_test` conda environment that worked around this issue at HCRO is no longer needed, and has been removed from the guide.
1.2	September 2022	S. Z. Sheikh	Moved the manual SEFD discussion to the appendix
1.3	October 2022	S. Z. Sheikh	Integrated information about the new SETI real-time beamformer mode
1.4	January 2023	W. Farah	Minor modification to backend calls. Add appendix on beamformer modes
1.5	February 2023	W. Farah	Minor modifications to backend calls.
1.51	May 2023	S. Z. Sheikh	Moved obsolete calls to appendix, highlighted to-do items in red
2.0	June 2023	S. Z. Sheikh	Incorporated plots and text from Miro Saide
2.1	July 2023	G. C. Brown	Added a guide to setting up Observer's Room monitors to the appendix.
2.2	February 2024	S. Z. Sheikh	Updated with new Python post-processor, added data storage section, more information on beamforming and calibration plots