Pointing calibration

Wael Farah

April 2021

The aim of this document is to describe how a pointing calibration observation is performed on the Allen Telescope Array. I will go through the scheduling of the observations and how pointing errors are estimated. I will then describe how to fit the offsets and produce a *pointing model* that can be loaded unto the antennas to correct for the pointing offsets. The content should hopefully be DSP-hardware-independent enough such that these observations can be performed using any signal processing backend. A minimum level of expertise in the ATA-specific observation and source-tracking logic is required as a prerequisite.

1 Introduction

In a perfect world, a telescope commanded to point at a specific position on the sky will always do so. In reality, however, this is never the case. Pointing errors, due to different mechanisms such as encoder "zero-points", faulty encoders, telescope's weight, non-perpendicularity between axes, etc..., are always present and have to be compensated for in order for an instrument to accurately point at sources. These offsets can simply be static offsets (index errors), but can also significantly vary depending on where the instrument is pointing. A commonly used procedure is to fit a **pointing model** to these offsets, and then apply it, in real time, when astronomical observations are taking place.

The ATA dishes are AltAz telescopes; Equatorial mounts are beyond the scope of this document. The AltAz 8-parameters pointing model that was found to properly describe the measured offsets of the ATA dishes is:

$$\Delta\text{el} = \text{IE} - \text{AN} \times \cos(\text{az}) + \text{AW} \times \sin(\text{az}) + \text{ECEC} \times \cos(\text{el}) + \text{ECES} \times \sin(\text{el}) \quad (1)$$

$$\Delta\text{az} \times \cos(\text{el}) = -[\text{CA} + \text{IA} \times \cos(\text{el}) + \text{NPAE} \times \sin(\text{el}) + \text{AN} \times \sin(\text{el}) \sin(\text{az}) - \text{AW} \times \sin(\text{el}) \cos(\text{az})] \quad (2)$$

The term az × cos(el) is also known as the "cross-elevation", a better representation of "sky coordinates". The terms are described below (from the `tpoint` manual, page 14):

- IE: index error in elevation
- IA: index error in azimuth
- CA: nonperpendicularity of elevation and pointing axis
- AN: NS misalignment of azimuth axis
- AW: EW misalignment of azimuth axis
- NPAE: nonperpendicularity of azimuth and elevation axis

- ECES: El centering error (sin component)
- ECEC: El centering error (cos component)

The terms in the above pointing model are also often called the "TPOINT coefficients" because of the name of the software used to fit them, which will be described in a following section.

The pointing models can be obtained on the "data" machine, under the directory:

`obs@data:/opt/atasys/data/Pointing`

2 Performing observations

Fundamentally, to perform pointing calibration, it is required that a measured set of offsets be obtained across as many pointing directions as possible. With a 6.1m dish, however, this is tricky as there are not enough astrophysical sources in the sky that are bright enough for such observation to take place. GPS satellites, on the other hand, are bright and have a large sky filling factor: at any single point in time, it is expected that at least 6 GPS satellites are visible in the sky. GPS satellites drift at around twice the sidereal rate, so tracking them with ATA dishes is easily doable. The catch is that it is required to accurately know the sky position of the satellites, to sub-meter accuracy. Luckily, this information is available online (https://www.igs.org/), and users should not worry about it as the ATA database is updated a few times a day with every available GPS ephemerides. Whenever the instrument is commanded to track a GPS satellite, those ephemerides (also called "ultra-rapid" ephemerides) are used to generate the sky track. We will use the GPS satellite signal at 1575 MHz for our observations.

As any knowledgeable radio-astronomy observer should hopefully know, the sky response of a receiver mounted on a telescope is a point spread function with FWHM $\sim 1.22 \times \lambda/d$[1], a frequency dependent entity. Thus, as we only have a "single-pixel" on the sky to obtain the offset measurement, a series of "cross" pattern measurements around the true position of the GPS satellite has to be performed. This is described in Fig 1. The power, at 1575 MHz, is measured for every point of the 10-point cross. The set of 5 points in each coordinate are then used to measure the offset by: subtracting the power of the outermost ("background") 2 points, and fitting with a Gaussian function. The position at where the Gaussian function peaks will then be the measured offset for the given coordinate.

Practically, performing the above observation requires a special mode where the antennas should keep tracking a GPS satellite, but at the same time offset in elevation and cross-elevation. Historically, this has been done using a "compound trajectory" (see this), a complicated approach to the problem at hand. An alternative that I proved worked is to perform the offsetting in the ephemeris file that gets sent to the antennas. The process is to generate a trajectory ephemeris file at the position of the GPS satellite, and then modify it by adding the "cross" pointing offsets (note that the ephemeris file lists El/Az, rather than El/X-el, which represent the coordinate system where the "cross" is performed. `ataephem` and in term all the `python` control utilities are now able to perform offsetting in the X-El and El coordinate system). The observing script then passes the modified ephemeris file to the antennas to command them to perform the "cross" pattern.

The current observation script tries to prioritise satellites that are setting, then chooses whatever other satellite is available. Once all satellites within the Elevation limits are observed, it either decides to wait for ~ 20 minutes, or chooses a random target to re-observe. The point of this scheme is to measure the pointing errors at as much sky positions as possible; the more the measurements

[1] A handy equation for the ATA is: FWHM $\sim 3.5/\nu_{\mathrm{GHz}}$[deg]

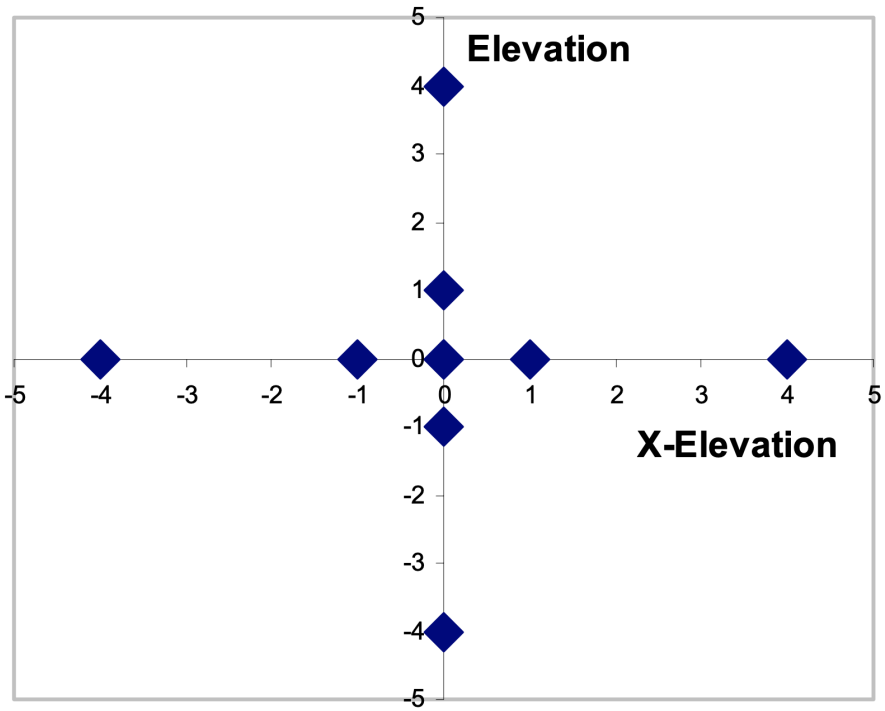

Figure 1: Plot taken from ATA-memo61. The scale on the plot are in units of HWHM, which is ~ 1.11 deg in the case of centre frequency of 1575 MHz.

are distributed on sky, the better. It's also good to let the observing script run for a few hours, ~ 100 data points are usually good to obtain a good fit.

This observing script is specifically written to perform these observations with the production SNAP board design, using the spectrometer mode and the `psrdada` FRB software backend.

3 Preparing `tpoint` input

`tpoint` is a software used to fit a pointing model, just like the one stated above, to measured pointing offset data. Because the ATA is a Az/El mount system, the input ASCII file that we will need to provide `tpoint` with is in Az/El coordinates. Therefore the format of the ASCII file our observing script should write is: "Observation record Format 4", as defined in page 41 of the manual. The most important columns of the ASCII input file are the first 4: Az/El of the "true" direction of the satellite, and the raw Az/El of the antennas. We do not have direct access to the raw Az/El, as they are embedded in the antenna control boards, so we will have to compute them live when running the calibration observations. In other words, we have to remove the effect of the current pointing model (read the "word of advice" on page 41 of the `tpoint` manual). This logic is incorporated in this function. Practically, how to construct the `tpoint` ASCII file is listed here:

- Write a boilerplate header containing the current pointing model parameters that was used in the observation. TPOINT doesn't see those, they are only for user reference.
- Keep track of the observed satellite az/el at each point of the cross pattern. Average the numbers at the end to obtain the averaged "true" position. Those Az/El values go into the first 2 columns of the ASCII file.
- Perform the Gaussian fit in the two coordinate directions, as described above. The measured offsets using this fit are the pointing errors; if the antennas are very well calibrated, those values should approach 0.
- The formula to remove the current pointing model and obtain the "raw" coordinates is: `removeTPOINT(AvCoord + OffCoord)`, where `removeTPOINT()` is a function that removes the effect of the pointing model, `AvCoord` and `OffCoord` are the average coordinates and measured offsets, respectively. Those values go into the next 2 columns of the ASCII file.

4 Running `tpoint`

Running `tpoint` is explained in detail in the `tpoint` manual. I will only summarise the steps here to make it easier for someone not familiar with the software to be able run it, but users are encouraged to read through the manual for a better understanding. A copy of the `tpoint` program exists on "control" machine. To run `TPOINT`, log in as "obs", and run the following:

```
obs@control:$ source /hcro/opt/tpoint/tpoint.sh
```

This will set up all the necessary environment variables to start up the software. Once that is done, a simple `tpoint` on the command line should launch the program. The software will return a command line prompt, where `TPOINT` commands can be typed in. First thing we have to do is load in the ASCII file that our observing script produced earlier. Let's call it "ant1f.dat"

```
INDAT ant1f.dat
```

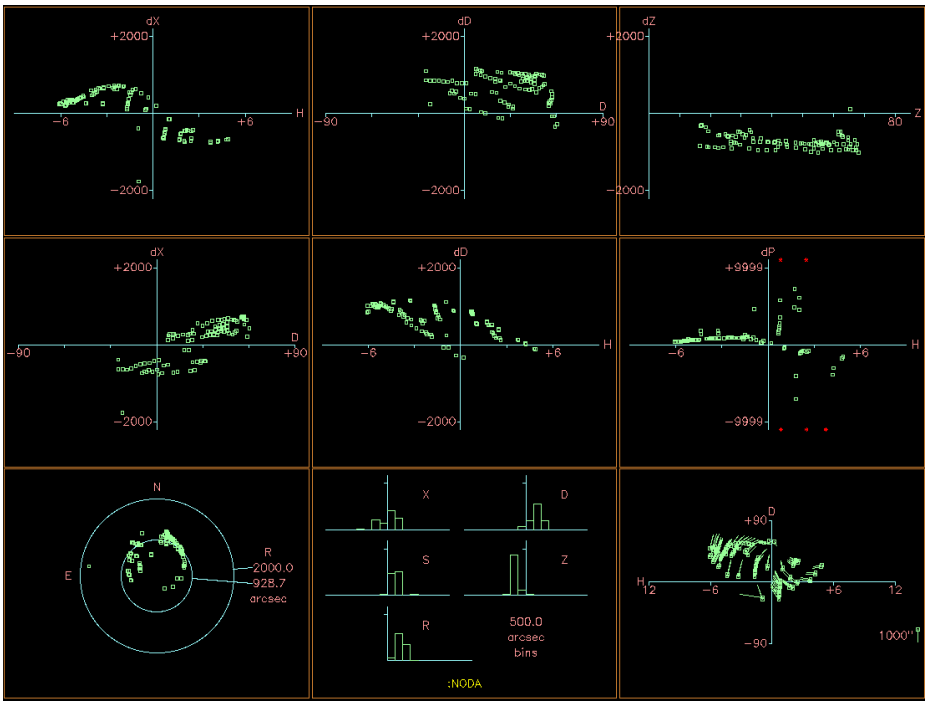

Figure 2: TPOINT's `.E9` before performing any fitting.

First thing we can do is produce a plot of the errors. The command to do so is: `.E9` (make sure X11 is forwarded for this plot to be displayed). This should hopefully display a plot similar to the one in Fig 2. The first 2 rows of the plot show subplots with errors, in either directions, in units of arcseconds, as a function of several coordinates. An explanation of those coordinates can be found on page 28 of the `tpoint` manual. Another important subplot is the bottom left, which can be thought of as a "bullseye" for pointing; the closer the points to the centre, the better our accuracy is. That subplot also shows the pointing RMS.

Next, we have to specify to `TPOINT` what parameters of our pointing model we would like to fit. For this, we do:

```
USE IA AN AW CA NPAE IE ECES ECEC
```

Those are the parameters that are defined in the pointing model above. Next, we begin our fitting by running the command `FIT`. This will perform a single fitting iteration. `FIT` will display the coefficients that we asked it to fit, along with the "change" in the values since the last `FIT` command, the "value" of parameter, and "sigma" which represents the error on the fit (all in units of arcseconds). Basically, the goal is to execute `FIT` multiple times such that the "change" for all the parameters becomes 0, which then represents a convergence of the model. Before doing that, however, some outliers would always exist. Thankfully, `TPOINT` tells you about them. It will display something like: "Observation #5 is an outlier candidate.". To remove the outlier, you can run:

```
MASK 5
```

Which will mask data point number 5 from the dataset. You can always rerun `.E9` to check how the errors look post the last fit/mask iteration. The `FIT` command also prints out the "Sky RMS" which is the RMS (in units of arcsec) on the pointing offsets after the fit, a number to keep track of. We aim to have an RMS value below 60 arcseconds, although I've seen RMS values of around ~ 30 arcseconds.

Once a few `FIT` and `MASK` iterations have been performed, a final `.E9` plot should look like the one in Fig 3. Once we are happy with the fit RMS and with the `.E9` plot, we can output the model into a text file by using:

```
OUTMOD ModelFile.mod
```

which will produce a file called "ModelFile.mod" in this case, with the model parameter values and the errors on them. The very last step involves editing the model files on the "data" machine (as listed above) with the newly obtained coefficients, and running an `atareloadpm ANTNAME` to load the newly edited pointing model to the antenna servers. Once all of this is done, congratulations on successfully calibrating the pointing of an ATA dish!

To summarise:

To summarise the above procedure, setting up and running `tpoint` requires:

1. `obs@control:$ source /hcro/opt/tpoint/tpoint.sh`
2. `obs@control:$ tpoint`

Then, to fit pointing models, you'll need:

- `INDAT TPOINT.FILE.dat`
- `USE IA AN AW CA NPAE IE ECES ECEC`
- `FIT`
- `MASK SOME.DATAPOINT.NUMBER`
- `FIT`
- `MASK SOME.OTHER.DATAPOINT.NUMBER`
- `FIT`
- ...
- OUTMOD ModelFile.mod

And finally:

1. edit the pointing model files on the "data" machine.
2. run `atareloadpm ANTNAME` to load the pointing model unto the antenna server.

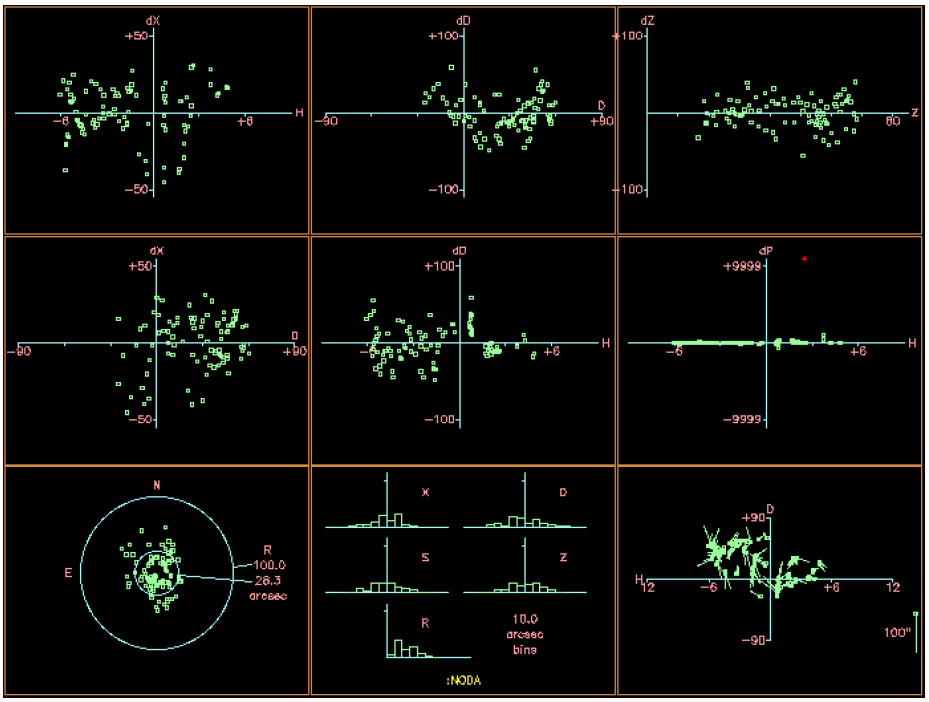

Figure 3: TPOINT's `.E9` after performing recursive fitting. The plots show a low RMS of 28.3 arcsecond.

5 Results with the 12 available antennas

A calibration observation took place mid April, 2021, that involved all the available 12 antennas that are connected to the SNAP boards at the time: 1a, 1f, 1c, 2a, 2b, 2h, 3c, 4g, 1k, 5c, 1h, 4j. In the plots, I show 2 set of measurements, one that was done while using an old pointing model, and the other with a new model. The new model was computed using the data collected when the old model was in place. The plots show the Az/X-El and El errors, in arcseconds, as a function of elevation in degrees. I also include a table showing the sky RMS that `tpoint` provides after performing the fitting.

Antenna	1c	1f	1h	1k	2a	2b	2h	3c	4g	4j	5c
Sky RMS	36.2	32.5	25.8	26.5	17.4	28.8	59.8	25.1	43.9	31.5	33.2

NOTE1: Some "before" pointing data points were off-the-chart, so I replaced them with the plot's maximum limit (500 arcsec) to make sure all the plots are scaled similarly.

NOTE2: antenna 2h shows sparse data points because of a failure in the digital backend while recording. Moreover, antenna 5c shows that the "after" pointing errors are off by $\sim$ 4 arcmins. Since the "before" data that were used to fit the pointing model were already on the order of a fraction of a degree, I anticipate a next iteration of pointing calibration will refine these values. The next pointing calibration observation is scheduled for the first week of May 2021, and that is expected to further fine-tune the models for all the working antennas.

NOTE3: Az offsets are usually "amplified" by a cos(el) effect. Therefore, it's better to evaluate the error in Az by looking at the X-El, rather than Az; I only added the Az subplot for completeness.

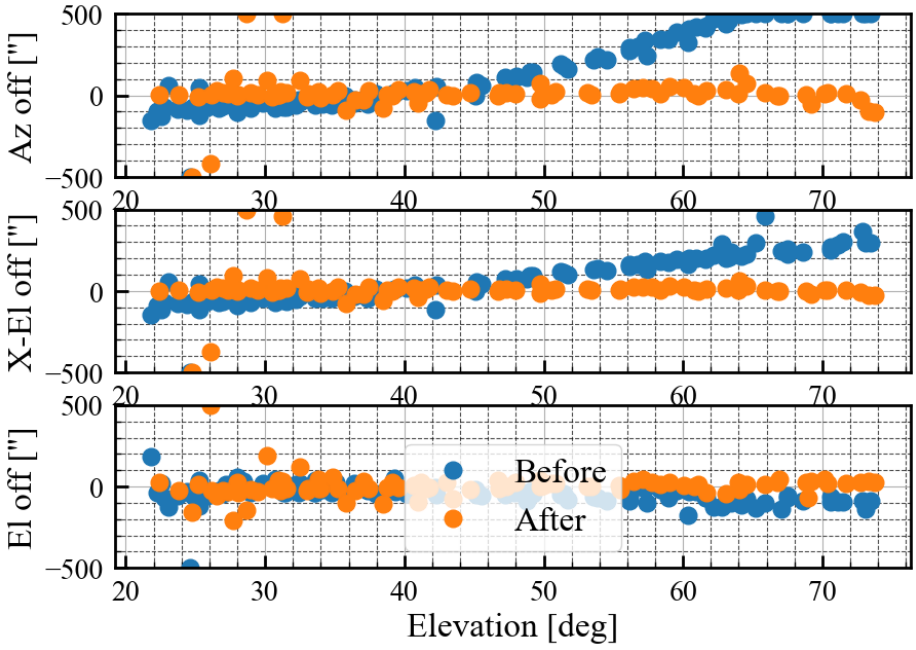

Figure 4: Antenna 1c

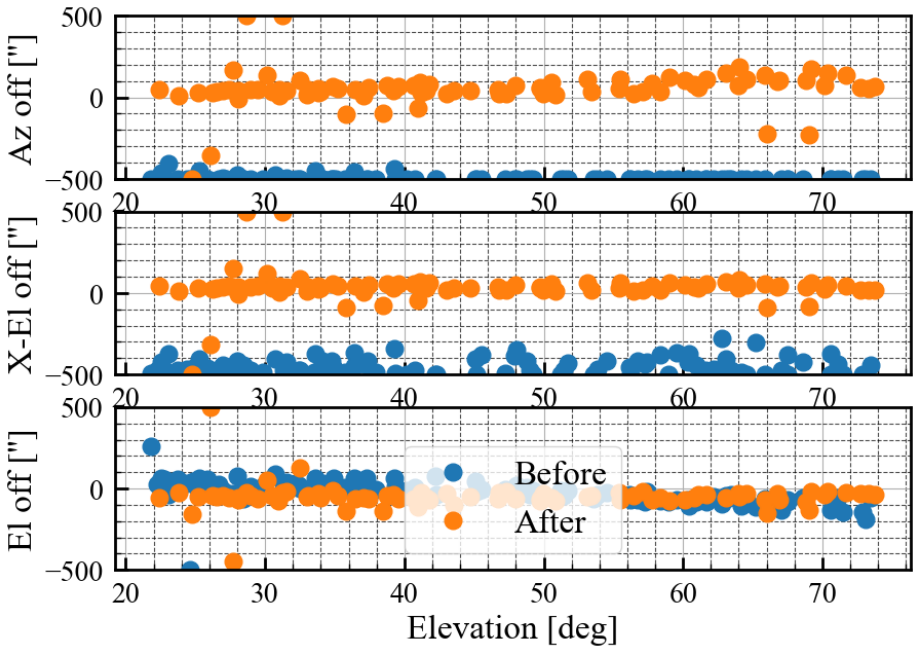

Figure 5: Antenna 1f

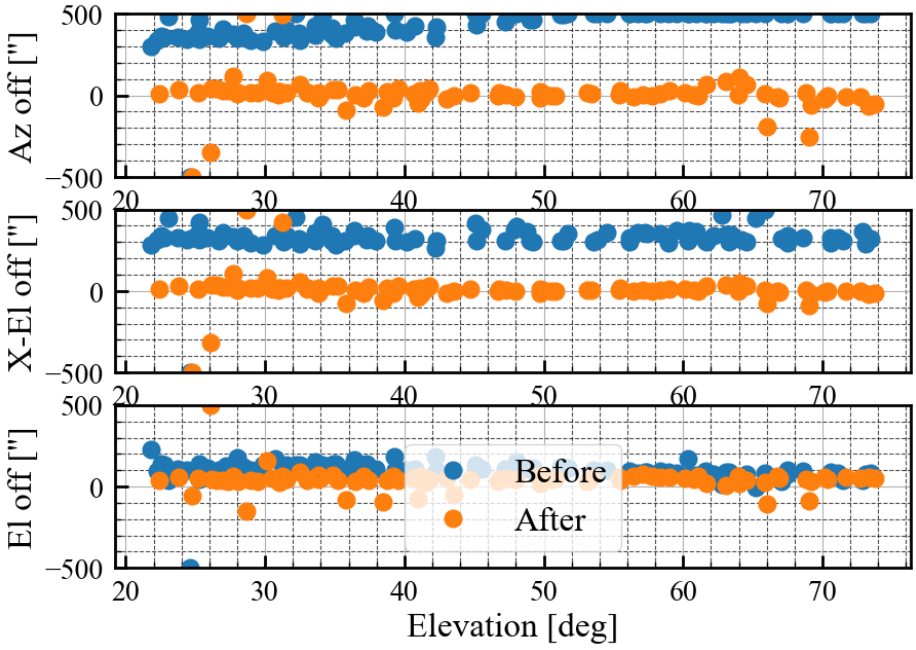

Figure 6: Antenna 1h

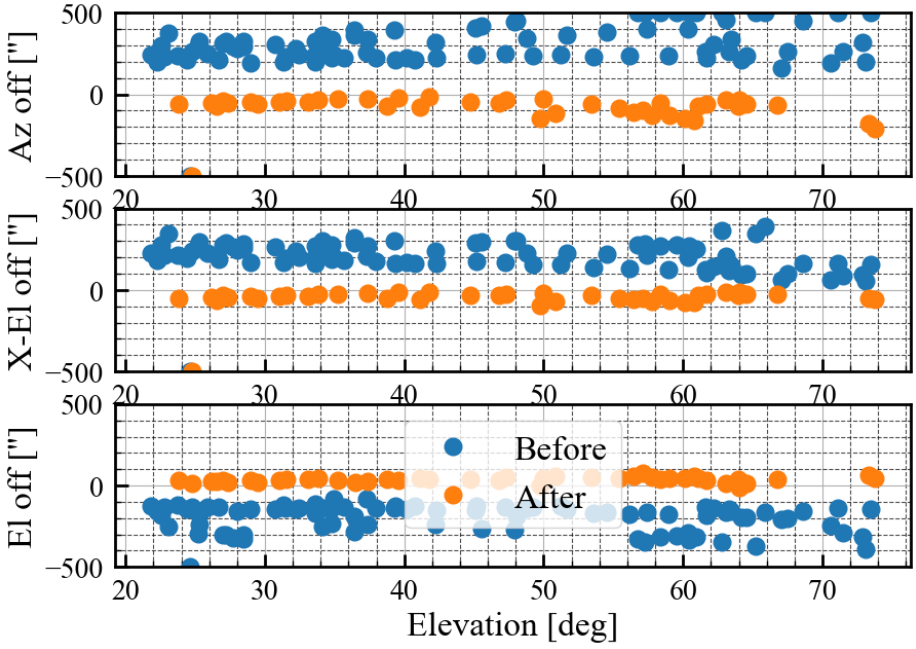

Figure 7: Antenna 1k

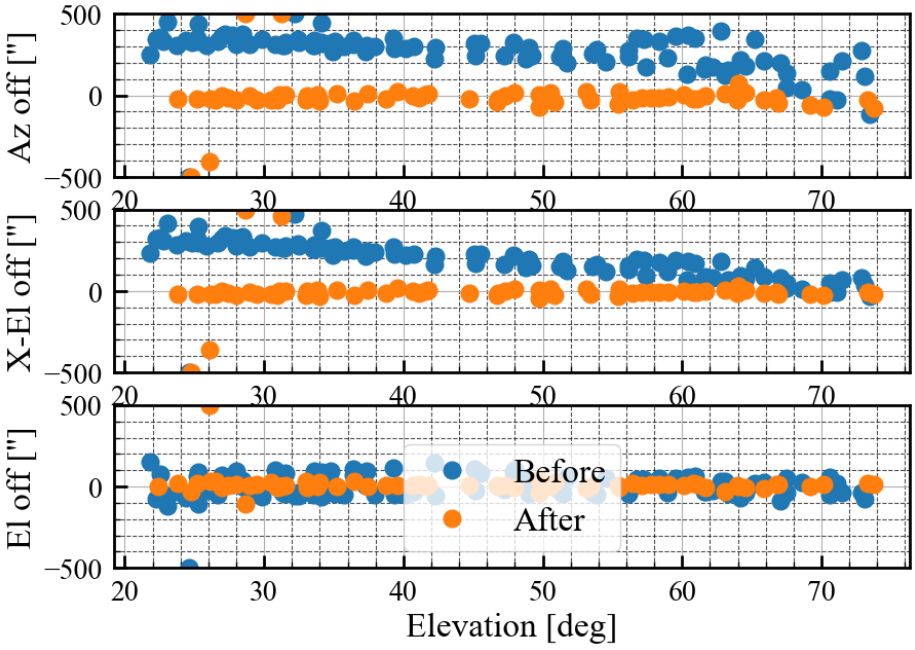

Figure 8: Antenna 2a

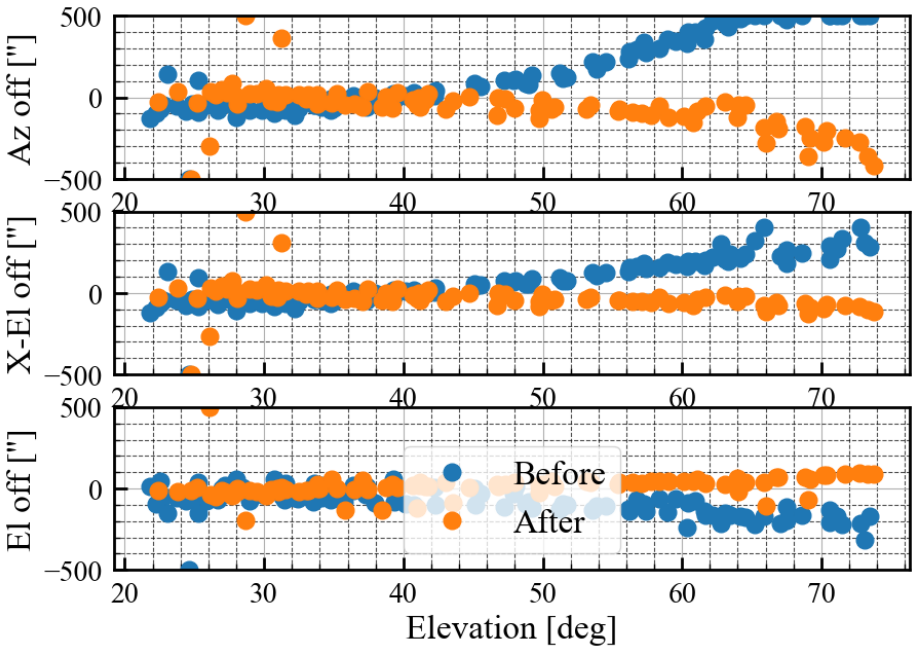

Figure 9: Antenna 2b

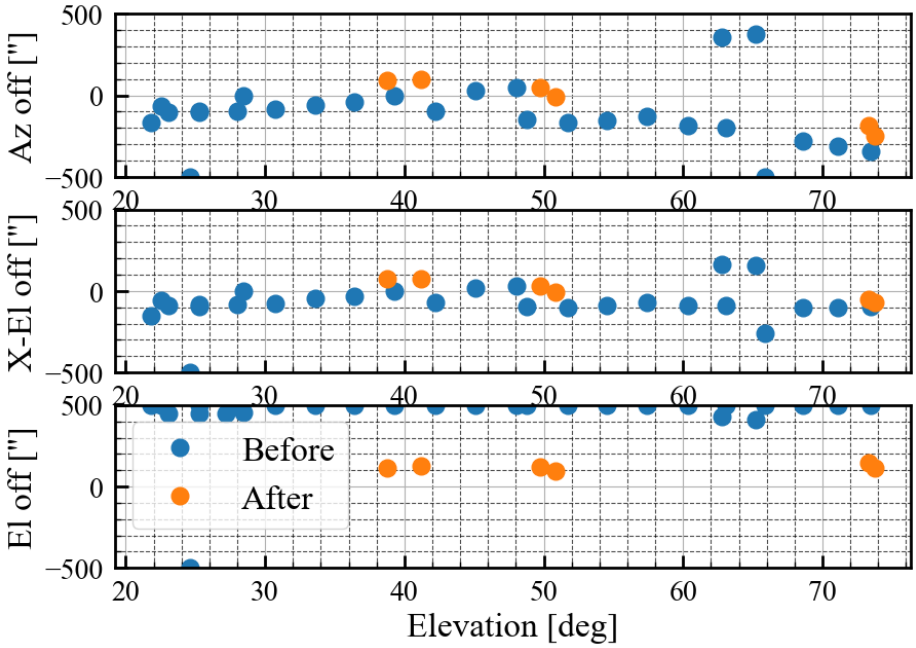

Figure 10: Antenna 2h (see NOTE2)

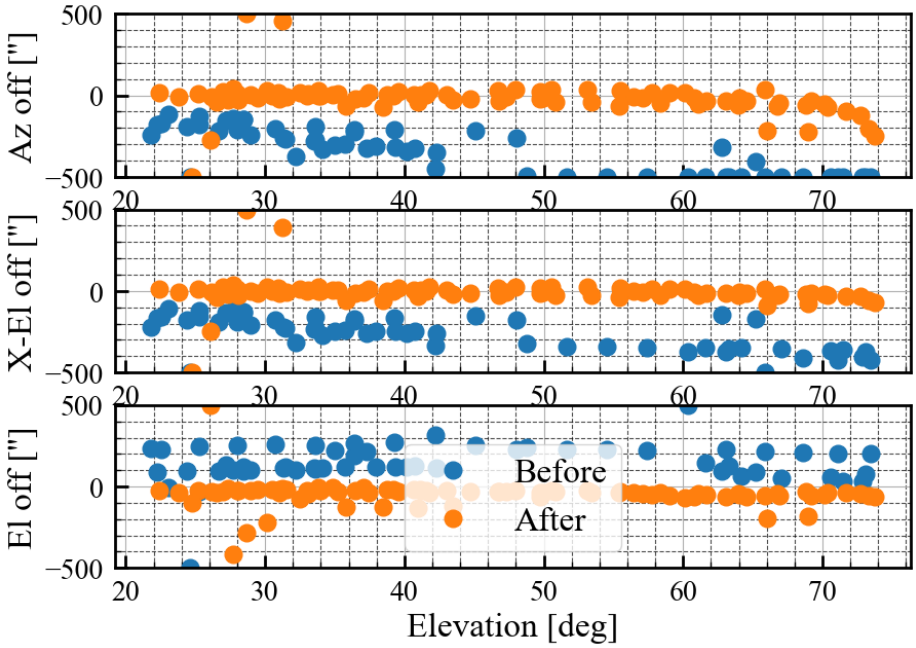

Figure 11: Antenna 3c

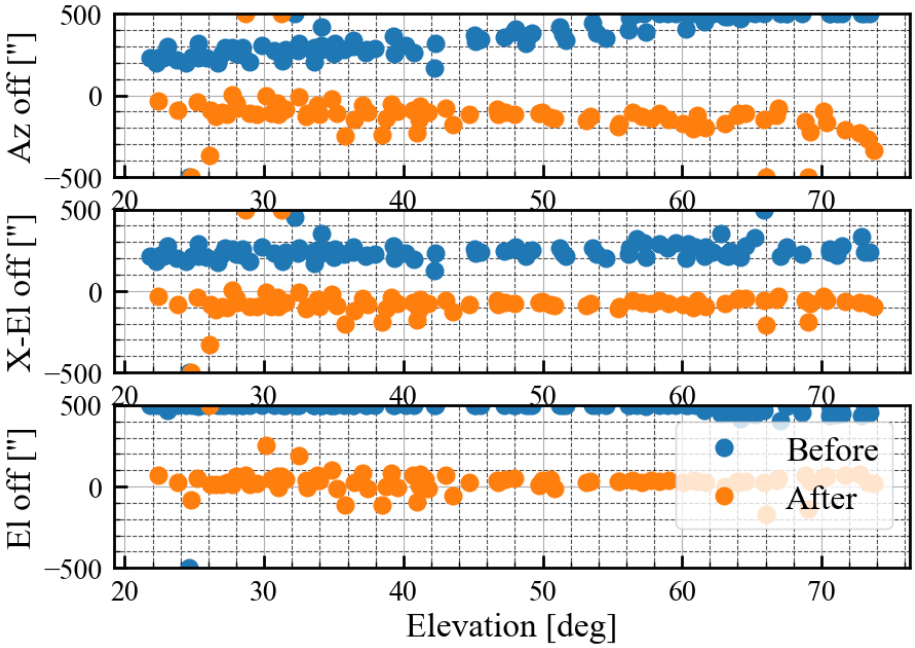

Figure 12: Antenna 4g

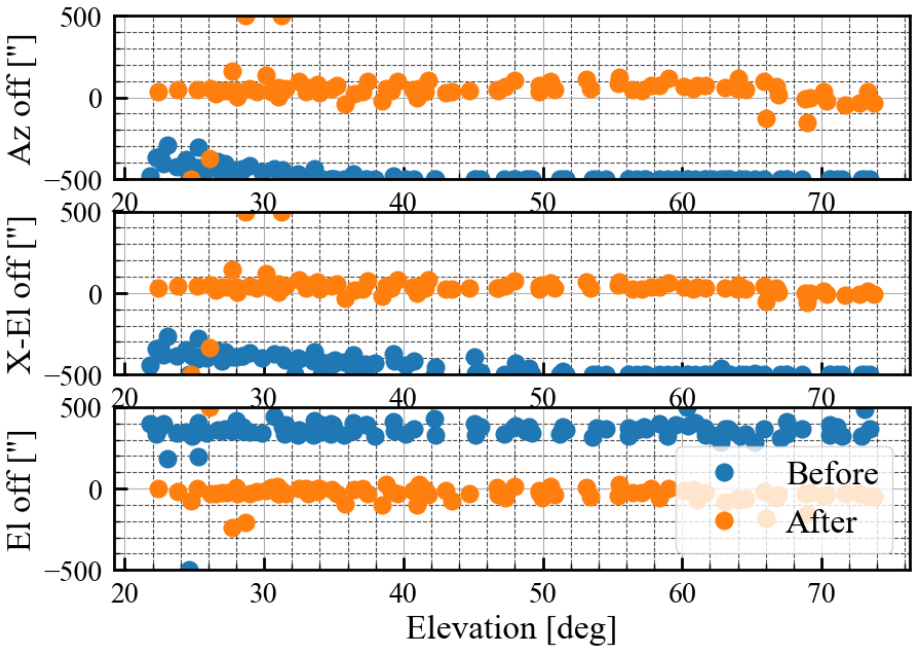

Figure 13: Antenna 4j

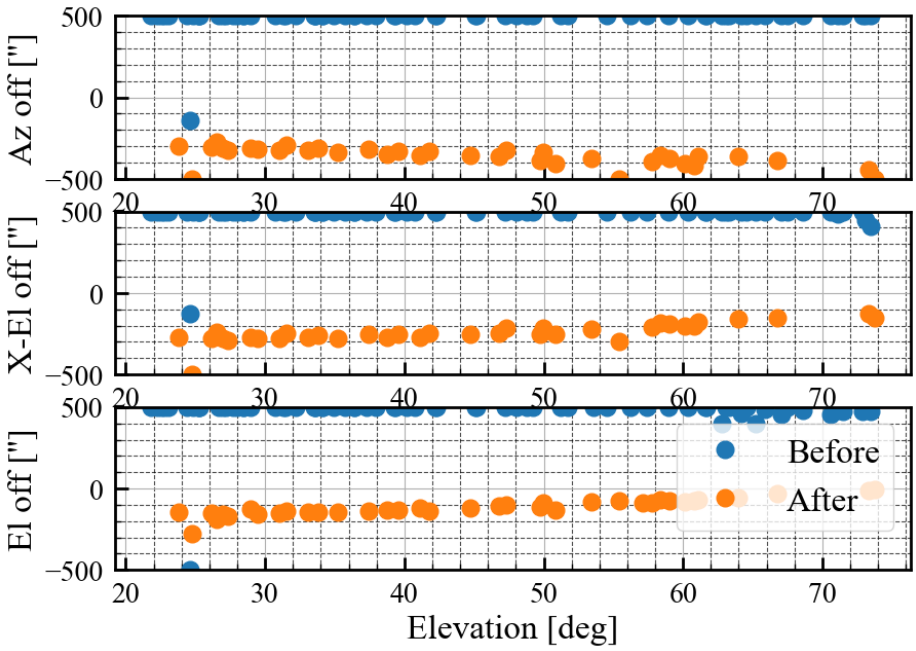

Figure 14: Antenna 5c (see NOTE1 and NOTE2 above)