Commissioning of the COSMIC

Savin Shynu Varghese

June 13, 2023

Outline

1. turboSETI and seticore benchmarking on setigen data
2. Fringes between antennas
3. Delay calibration
4. Understanding the correlated data products
5. Gain calibration
6. Offline calibration and first images from COSMIC
7. Online calibration testing
8. Commissioning the BLADE beamformer with another python beamformer code
    a. Tests with methanol Maser
    b. Test with simulated data
9. Commissioning the BLADE beamformer search mode using setigen data
10. Observations of Voyager1 to finalize the commissioning
11. RFI flagging to improve calibrations

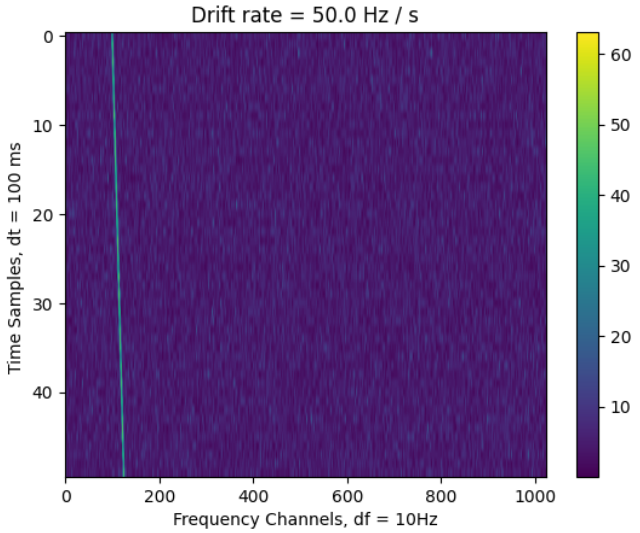
1. turboSETI and seticore benchmarking on setigen data

The goal of this test is to understand the de-doppler SETI search runtimes of turboSETI and seticore softwares using the data from COSMIC pipeline. Since we did not have finalized dynamic spectra (filterbank/h5 file) from the pipeline in the beginning of the commissioning phase, a simulated data similar to the VLASS data product was generated using the setigen software.

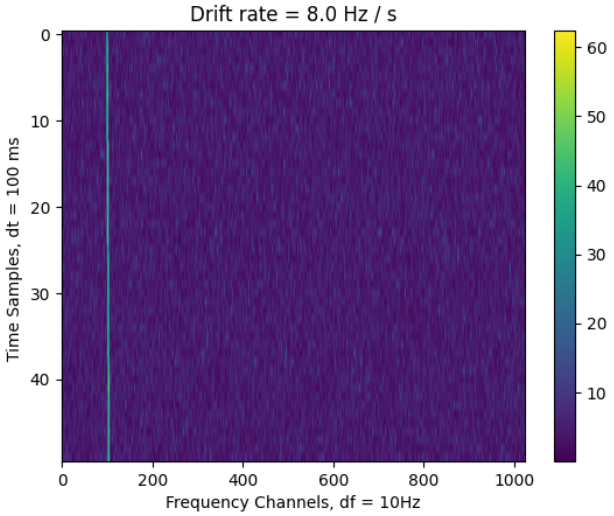
Generation of simulated data:

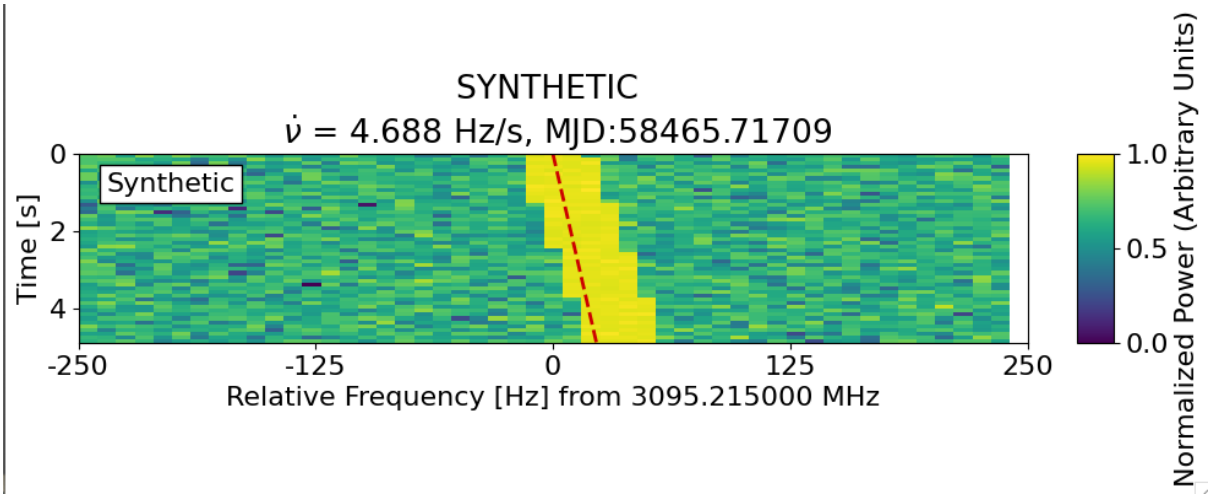
Expected VLASS data product:
Observing length = 5s
Time resolution = 100 ms
Number of time samples = 50
Frequency = 2-4 GHz (Here using some random value of 3.09 GHz)
Frequency resolution = 10 Hz
Number of fine frequency channels = 3,200,000 (if 32 MHz of data is processed in each GPU compute node)
Dynamic spectra/spectrogram = 50 (t_samples) * 3200000 (frequency channels) array

In the setigen software, first a frame is created with the above listed parameters. Then a Chi square noise (high resolution BL data follows this) is added to the frame. Then signals with preferred SNR, drift rate and width are injected into the frame. An example of an injected signal with drift rate 50 Hz/s is shown below. Once the signals are injected, the frames are saved into h5 files and runtime tests are carried out in turboSETI.



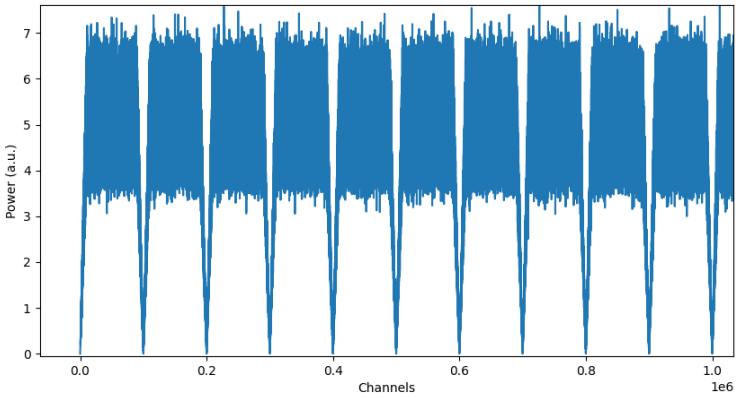
One issue: For the initial testing, a dynamic spectra with 50 time samples and 1024 frequency channels was injected with a 8 Hz/s drift rate signal. Then turboSETI was used to confirm that we are actually detecting the injected signal. Even though, a strong signal was detected at the injected frequency, the drift rates of detection were different from injection. Attaching the plots of injected signal and the recovered signal from turboSETI in the form of hit.



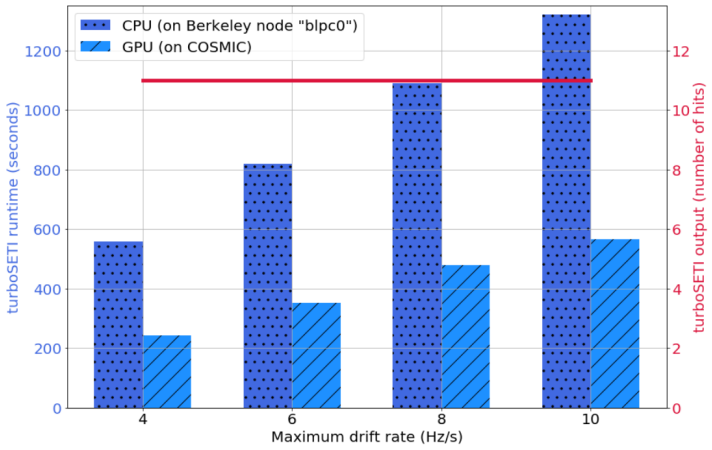


Testing from a VLASS perspective:

The VLASS data would have 32 coarse channels each with 100,000 fine channels. A signal with SNR between 10-50 was injected into the middle of each coarse channel with drift rates going from 5-55 Hz/s. A bandpass response was added to each coarse channel to simulate the actual data. This created a 1.1 Gb h5 file. The plot below shows the bandpass response of each coarse channel.



From a recent turboSETI test conducted using the Meerkat data, dedoppler search runtimes on both CPU and GPU increased as a function of drift rate which is shown below. Other parameters such as SNR and frequency channels does not seem have an effect on the runtimes.



Runtimes are calculated for 2 processes: dedoppler drift search and filtering and plotting events (Both Python versions)
Turboseti search parameters:
SNR = 6, n_coarse_channel = 32,

Run times for the dedoppler drift search in CPU and GPU are shown in the table below.

Max Drift rates (Hz/s)	GPU runtime (+/-1.2 s)	CPU runtime (+/-1.2 s)
10	3.6	6
20	3.6	6
30	3.6	6
40	3.6	6
50	3.6	6

60	4.2	6

The GPU and CPU run times seem to be independent of the max drift rates used for the search. The drift rate looks like an independent parameter when the observation times are smaller.

Finding events and plotting time:

Max Drift rates (Hz/s)	CPU runtime (s)	No of hits
10	0.46	1
20	3.49	9
30	6.71	18
40	10.22	28
50	10.42	29
60	10.51	29

The runtimes were also tested with an observational data from Meerkat and compared with the setigen generated file (using observed parameters). Both the files are identical based on their header information. The turboseti takes longer time for observed data and 3 times lesser for the simulated data.
Maybe the simulated data is a simplified version of observed data? (Not many RFIs, etc..)

Runtimes of turboSETI and seticore:

Benchmarking of setigen generated dynamic spectra (.h5 file) with seticore and turboseti:
Comparing only the GPU run times. The seticore software only has a GPU version. During these tests, no process were running on any of the GPUs. seticore shows segmentation fault error messages sometimes. The search was done on two different files
First, the filterbank file were generated directly from setigen:

```
(turboseti) svarghes@cosmic-gpu-0:~/benchmark_test/test_seticore$ watutil -i data_setigen_ds32.h5

--- File Info ---
DIMENSION_LABELS :   ['time' 'feed_id' 'frequency']
       az_start :                                0.0
      data_type :                                  1
           fch1 :                     1999.5 MHz
           foff :       1.0172526041666666e-05 MHz
          ibeam :                                 -1
     machine_id :                                 20
         nbeams :                                  1
          nbits :                                 32
         nchans :                            3145728
           nfpc :                              98304
           nifs :                                  1
    rawdatafile :                          Synthetic
    source_name :                          Synthetic
        src_dej :                       -28:22:59.16
        src_raj :                           17:47:15
   telescope_id :                                  6
          tsamp :                           0.098304
   tstart (ISOT) :          2018-12-13T17:12:37.000
    tstart (MJD) :               58465.71709490740 6
       za_start :                                0.0

Num ints in file :                                 50
      File shape :                  (50, 1, 3145728)
--- Selection Info ---
Data selection shape :                  (50, 1, 3145728)
Minimum freq (MHz) :                           1999.5
Maximum freq (MHz) :                  2031.4999898274739
```

At the same time, raw voltages were generated for a single antenna with the expected configurations from a single COSMIC node: 32 coarse channels (1 MHz bandwidth and 1 us resolution). Then conducted upchanelizaton using rawspec to achieve high time and frequency resolution (0.1 seconds and 10 Hz resolution) similar to VLASS. The results on this data product are different.

```
(turboseti) svarghes@cosmic-gpu-0:~/benchmark_test/test_seticore$ watutil -i synth_guppi_sing_ant_sig.rawspec.0000.h5

--- File Info ---
DIMENSION_LABELS :   ['time' 'feed_id' 'frequency']
       az_start :                                0.0
      data_type :                                  1
           fch1 :                     1999.5 MHz
           foff :       1.0172526041666666e-05 MHz
          ibeam :                                 -1
     machine_id :                                 20
         nbeams :                                  1
          nbits :                                 32
         nchans :                            3145728
           nfpc :                              98304
           nifs :                                  1
    rawdatafile : synth_guppi_sing_ant_sig.0000.raw
    source_name :                          SYNTHETIC
        src_dej :                           0:00:00
        src_raj :                           0:00:00
   telescope_id :                                 -1
          tsamp :                           0.098304
   tstart (ISOT) :          2000-01-02T00:00:00.000
    tstart (MJD) :                           51545.0
       za_start :                                0.0

Num ints in file :                                 50
      File shape :                  (50, 1, 3145728)
--- Selection Info ---
Data selection shape :                  (50, 1, 3145728)
Minimum freq (MHz) :                           1999.5
Maximum freq (MHz) :                  2031.4999898274739
```

Benchmarking of seticore and turboseti on h5 file generated directly by setigen:

Max Drift rates (Hz/s)	seticore (s)	turboSETIi (s)
10	20	6
20	18	6
30	14	6
40	27	6.6
50	22	6
60	17	6

Benchmarking of seticore and turboseti on h5 file generated by channelization of raw voltages:

Max Drift rates (Hz/s)	Seticore (s)	Turboseti (s)
10	8	10.8
20	7	11.4
30	7	10.8
40	7	10.8
50	7	10.8
60	7	10.8

At some drift rates, few of the turboSETI hits are missing in seticore and vice versa. Other than that hits from turboseti and seticore agree with each other. Overall the seticore seems to be slightly faster over the turboSETI GPU version.

2. Fringes between antennas:

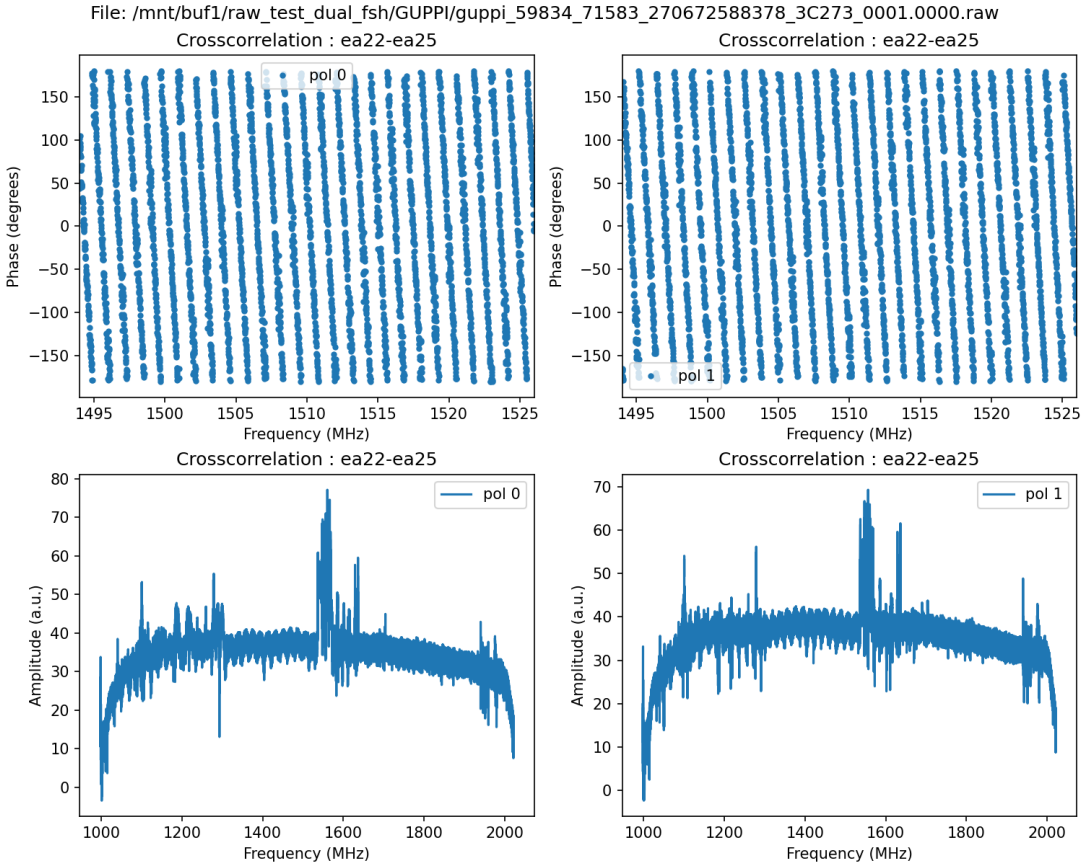
The hashpipe sotware is used to conduct reatltime flow of data and searching for the technosignatures. After setting up the basic hashpipe pipeline to collect data from FPGAs to GPU machines, we conducted observations with a small number of antennas. At this stage, we mainly utilized the raw mode to collect and save voltages as Guppi raw files. The first step was to get coherence from a pair of antennas. For that purpose, we cross correlated the voltage data for each baseline and averaged them for a short duration of time (~1 sec). If there is a fixed time delay between a pair of antennas, that will show up as fringes in the phase vs frequency plot for a baseline. The slope of the phase and how fast the phases are changing across the frequency depends on the delay values. If the delay values are higher, the fringes will change faster across frequency and slope would be higher. At the same time, if the delay values are smaller, they will appear as a slow slope across frequency.

The FPGAs conduct the channelization of the voltage data to 1 MHz frequency coarse channels. If the delay is << (1/1MHz = 1 us), then we would be able to see fringes with the coarse channelization. However, if the delay > 1us, then a step of upchannelization is required to see the fringes. The commissioning started in the VLA C configuration and a total delay greater than the 1 us is expected for several baselines. Here, an upchannelization factor of 128 was used to get around 8 kHz frequency channels. In order to observe coherence, a targeted observation of a bright 3C source across 1-10 GHz was conducted during the test times which will

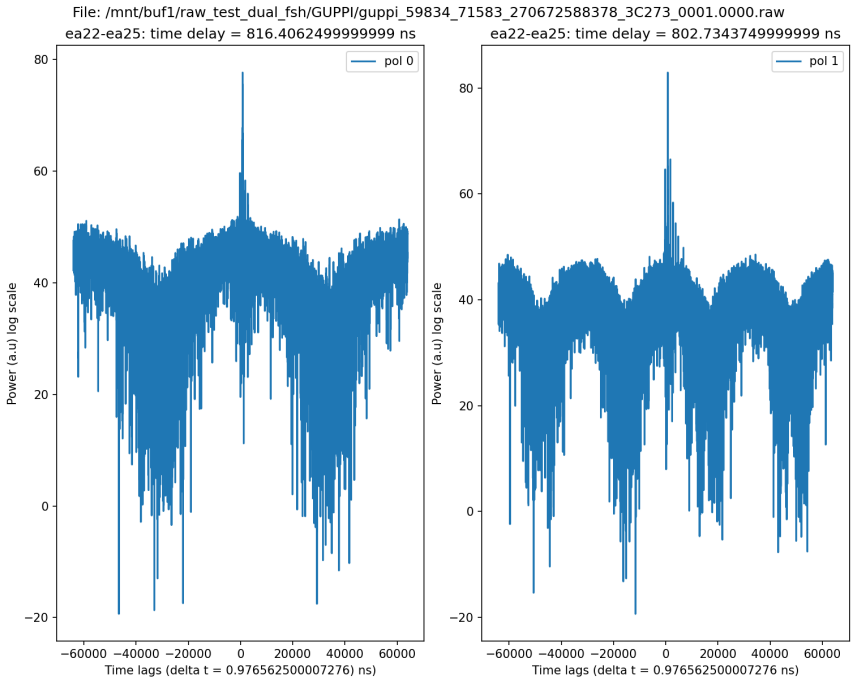
record 5 sec observations as voltage files. The observation were done in 4 ways to verify the system:

1. antenna fshift on | fpga unshift on
2. antenna fshift on | fpga unshift off
3. antenna fshift off | fpga unshift on
4. Antenna fshift off | fpga unshift off

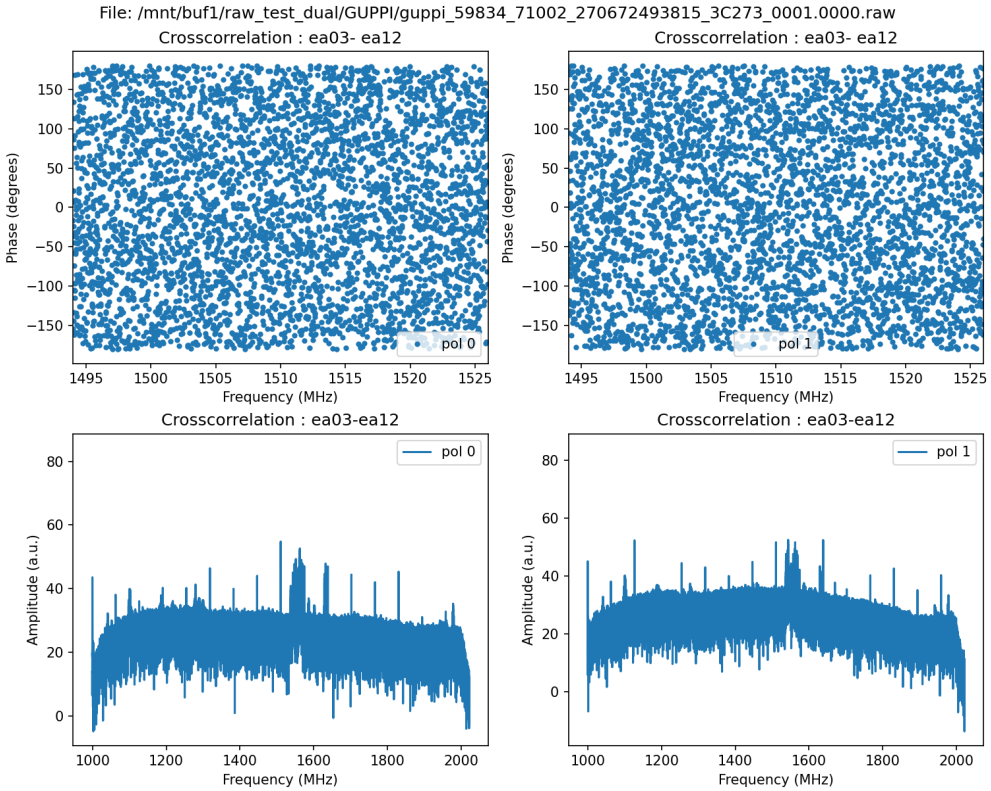
Technically, we should see strong fringes in test1 and lesser strong fringes in test4. Also, we should see decoherence in tests 2 and 3. These tests were carried out this way to make sure that the unshifting of frequencies in the FPGAs were working properly.
The images below show the fringes in phase vs frequency for a baseline ea22-ea25 in test1 using 3C273. The corresponding amplitude responses are also shown in the bottom panel for each polarization.

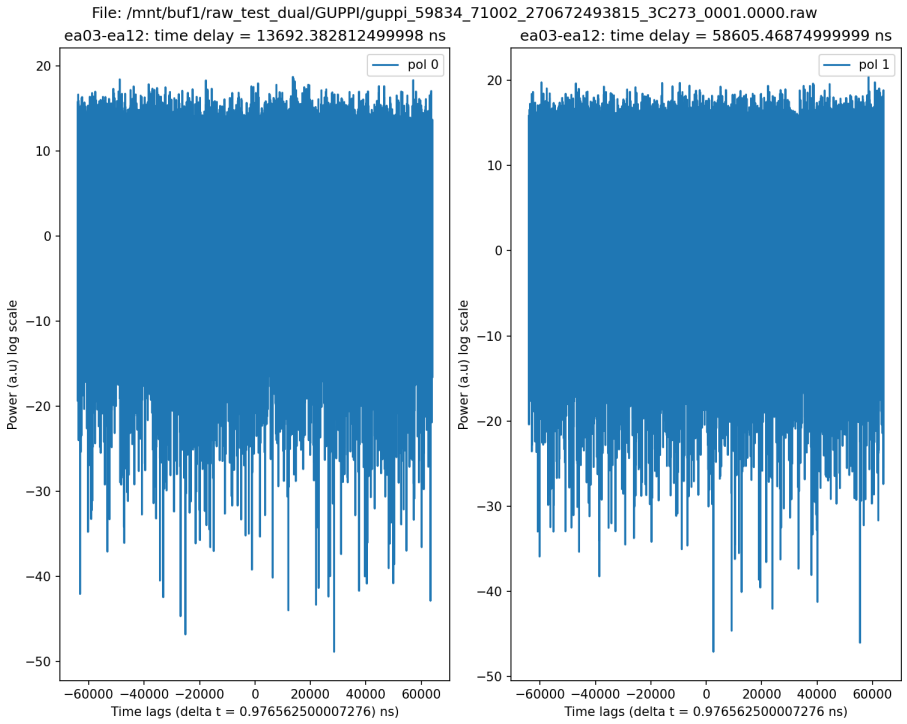


An inverse FFT of the upchannelized data would give the delay peak corresponding to each baseline. The figure below shows the power as a function of different delay lags and peak delay value is noted in the title of the plot for each polarization. In this case each polarization is measuring around ~800 ns delay.



Similar results from test 2 shows decoherence on baselines and inverse FFT of the data would look like noise without any delay peaks



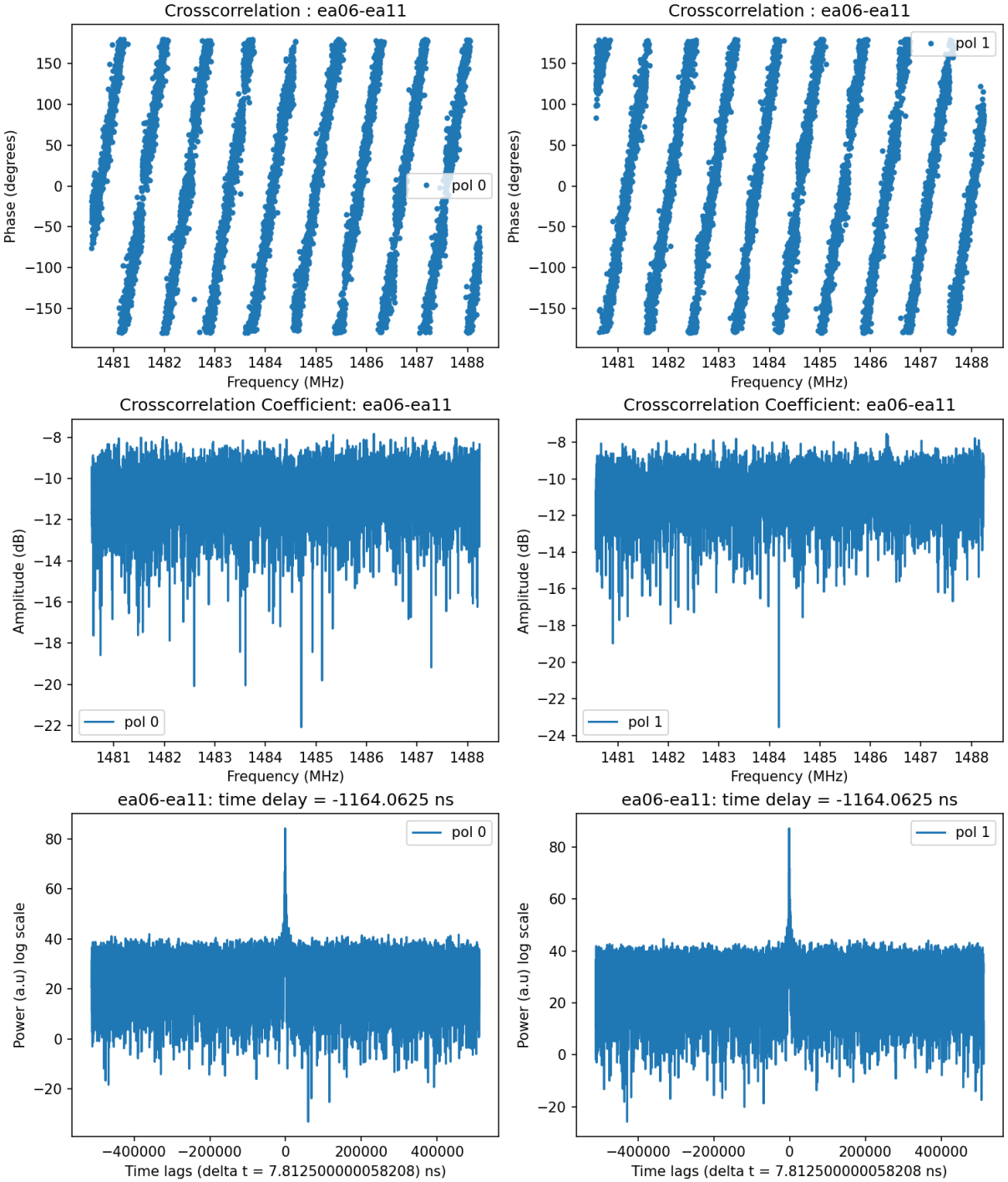


On a regular VLA operation process, we follow test1 where the antenna fshift is on from the NRAO side and a corresponding unshifting is also on from the cosmic side.

Another example of coherence from a regular operation mode is shown below from observations of 3C273 at L band. Top panel shows the phase vs frequency in which fringes can be seen clearly. The middle panel shows the correlation coefficient and the bottom panel shows the corresponding time delay of the baseline.

The codes used for the upchannelization, cross correlation, delay calculation and plotting of the fringe patterns and amplitudes are available at
https://github.com/savinshynu/VLA_COSMIC/blob/master/obs_rawdata/upchan_coherence_opt2.py
https://github.com/COSMIC-SETI/COSMIC-VLA-CalibrationEngine/blob/rfi_mitigation_and_arrayconfig_update/upchan_coherence.py

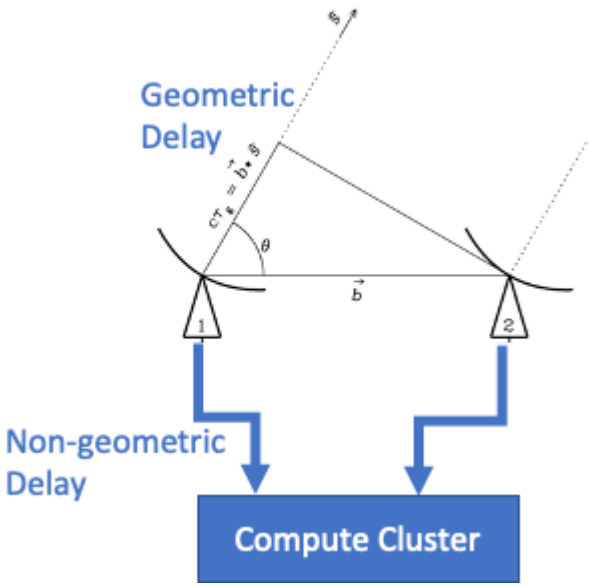
File: guppi_59892_70638_38187331456_3C273_0001.0001.raw_1480.560-1488.240



3. Delay Calibration

After observing coherence between all the baselines in an observation, the next important step is to conduct the delay calibration. COSMIC receives a copy of the digitized voltages from the antennas directly and the signal pathway is different from the WIDAR. The delay calibration consists of 2 steps

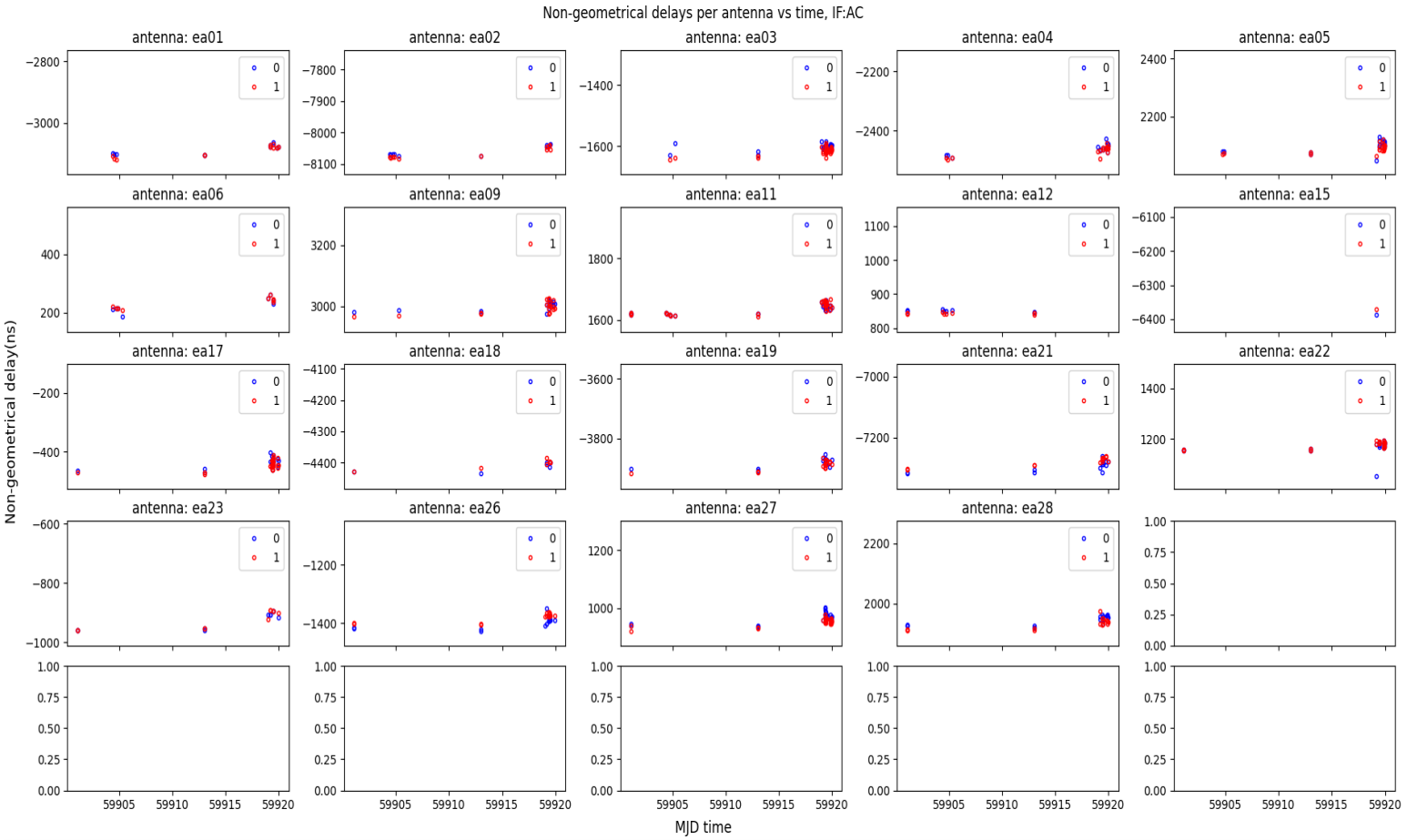
- Estimation of fixed or non-geometric delays mostly associated with the fibers, electronics, etc
- Estimation of geometric delays which can be easily calculated using accurate models

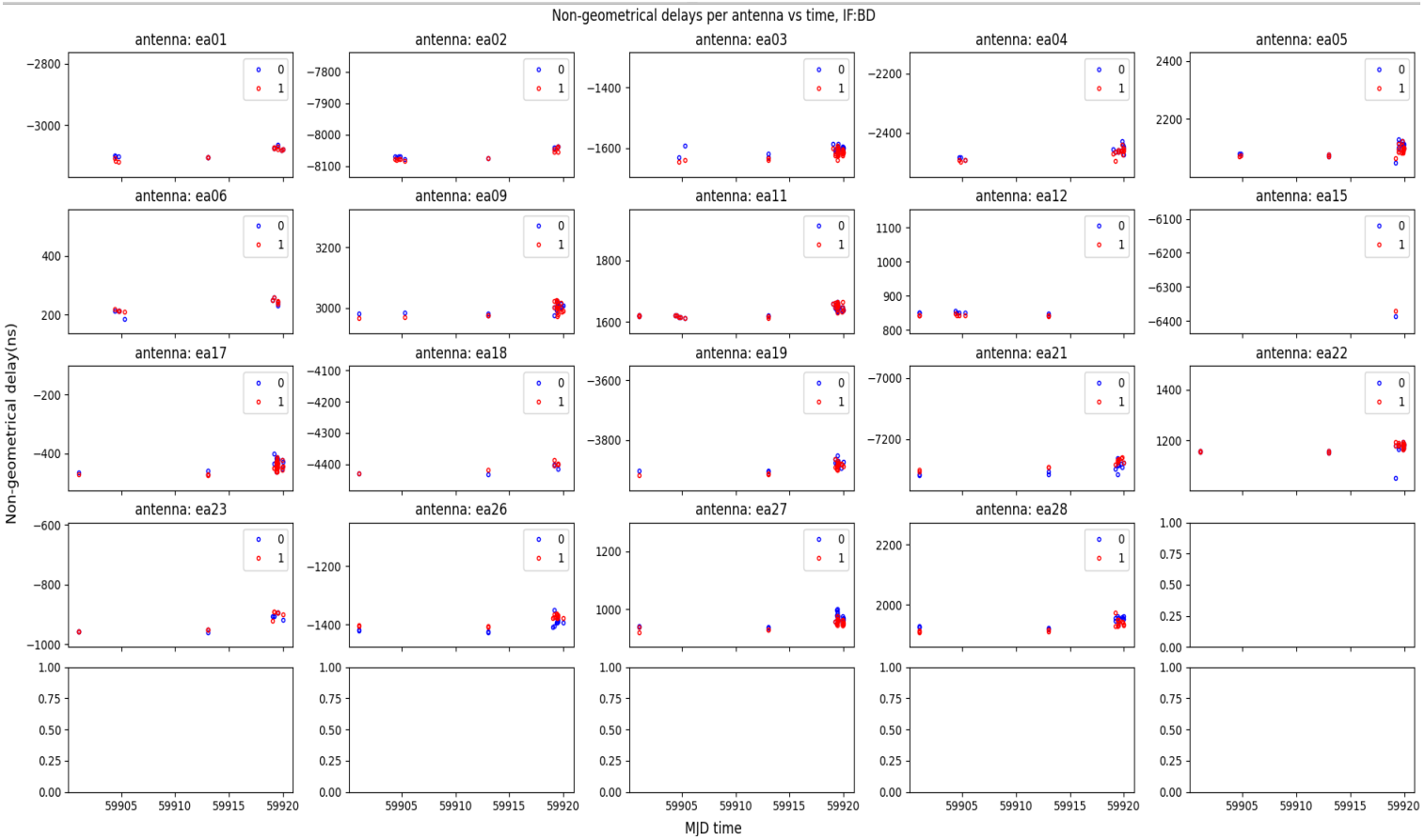


Here, the non-geometric delays should be constant as a function of time and we expect up to 10-15 ns difference for different receivers. At the same time, the geometric delays will change as a function of time, baseline and source direction.

The total delays are calculated using the inverse FFT of the cross correlated data for each baselines. The geometric delays are calculated using a well tested model used in CASA. The geometric delays are subtracted from the total delays to measure the fixed/non-geometrical delays for each baseline or each antenna wrt to reference antenna. These fixed delays are measured/updated during the start of each VLA reconfiguration and updated in the FPGAs. A sum of the fixed delay values and the geometrical delay based on the VLA model is compensated for each antenna in the FPGAs. This will ensure delay tracking of sources in the observation.

We conducted observation of bright 3C sources and other calibrators over a few weeks to measure the fixed delay values and measure their consistency. If the fixed delay values are not consistent over time, the delay tracking will fall apart. The plots below show the consistency of measured fixed delay values from different observations of source across multiple bands in the AC and BD tunings. An average of these delays measured across multiple days are written into a csv file which is later uploaded to the FPGAs for delay tracking. This process is repeated in a partially automated manner during each configuration change. The codes for fixed delay calculations are embedded in
https://github.com/COSMIC-SETI/COSMIC-VLA-CalibrationEngine/blob/rfi_mitigation_and_arrayconfig_update/upchan_coherence.py


Non-geometrical delays per antenna vs time, IF:AC
antenna: ea01
antenna: ea02
antenna: ea03
antenna: ea04
antenna: ea05
antenna: ea06
antenna: ea09
antenna: ea11
antenna: ea12
antenna: ea15
antenna: ea17
antenna: ea18
antenna: ea19
antenna: ea21
antenna: ea22
antenna: ea23
antenna: ea26
antenna: ea27
antenna: ea28
Non-geometrical delay(ns)
MJD time


Non-geometrical delays per antenna vs time, IF:BD
antenna: ea01
antenna: ea02
antenna: ea03
antenna: ea04
antenna: ea05
antenna: ea06
antenna: ea09
antenna: ea11
antenna: ea12
antenna: ea15
antenna: ea17
antenna: ea18
antenna: ea19
antenna: ea21
antenna: ea22
antenna: ea23
antenna: ea26
antenna: ea27
antenna: ea28
Non-geometrical delay(ns)
MJD time

4. Understanding Correlated Data Products

Once the delay tracking is enabled, the next important step of the commissioning was to move towards the correlated mode which cross correlates the voltages and outputs them in the UVH5 format.
Important tests with the correlated data:

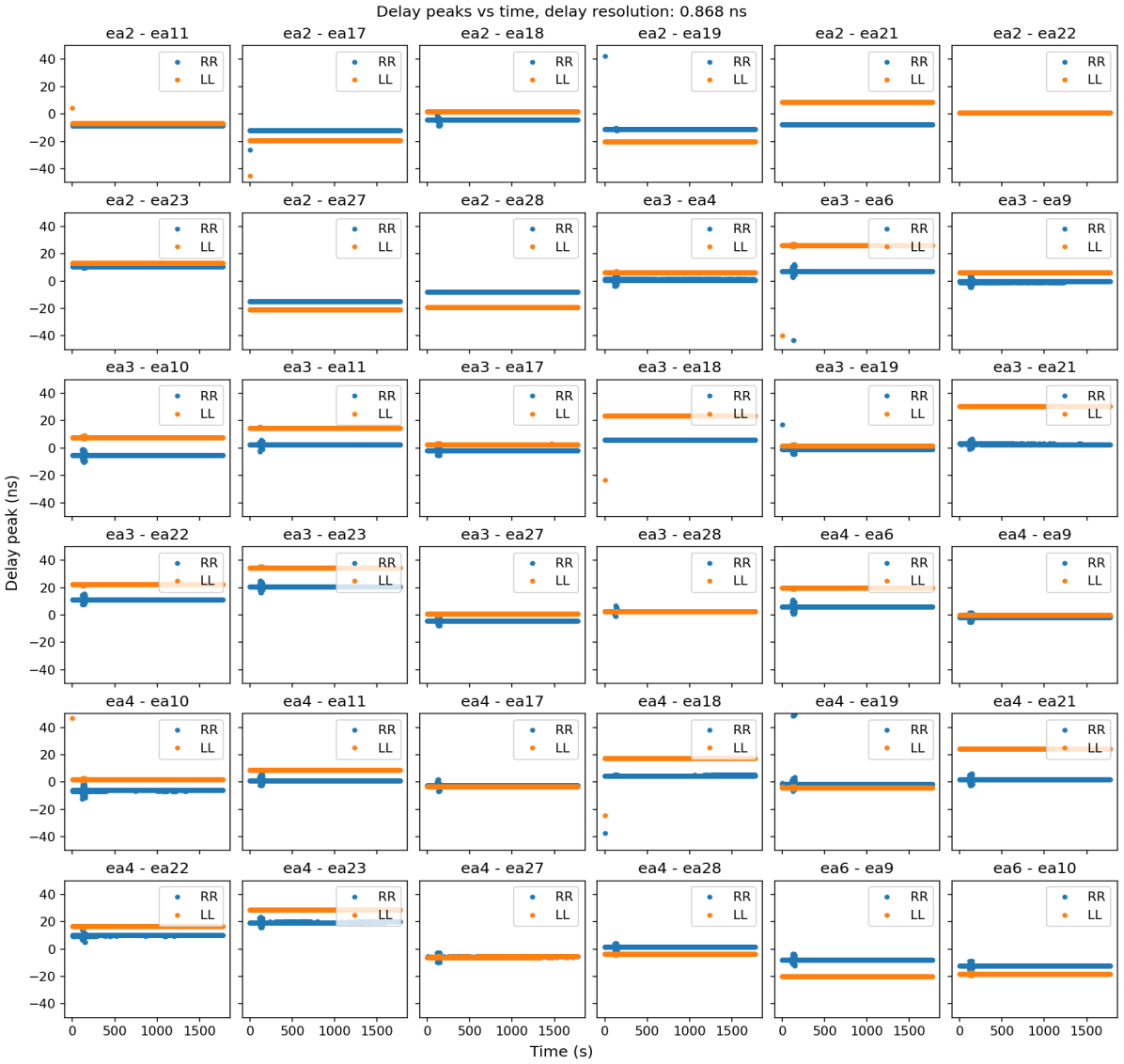
1. Delay tracking as a function of time
2. Checking fringe rotation in the data
3. Phase tracking as a function of time/removing fringe rotation

In order to conduct testing of the correlated data, observations of bright 3C radio sources were carried out in both tunings (AC and BD in 8 bit mode) for shorter (2 min) and longer (30 min) durations. For diagnosing the UVH5 files, a python script was written and available at the link below
https://github.com/COSMIC-SETI/COSMIC-VLA-CalibrationEngine/blob/rfi_mitigation_and_arrayconfig_update/calibrate_uvh5.py and
https://github.com/savinshynu/VLA_COSMIC/blob/master/calib/calibrate_uvh5.py .

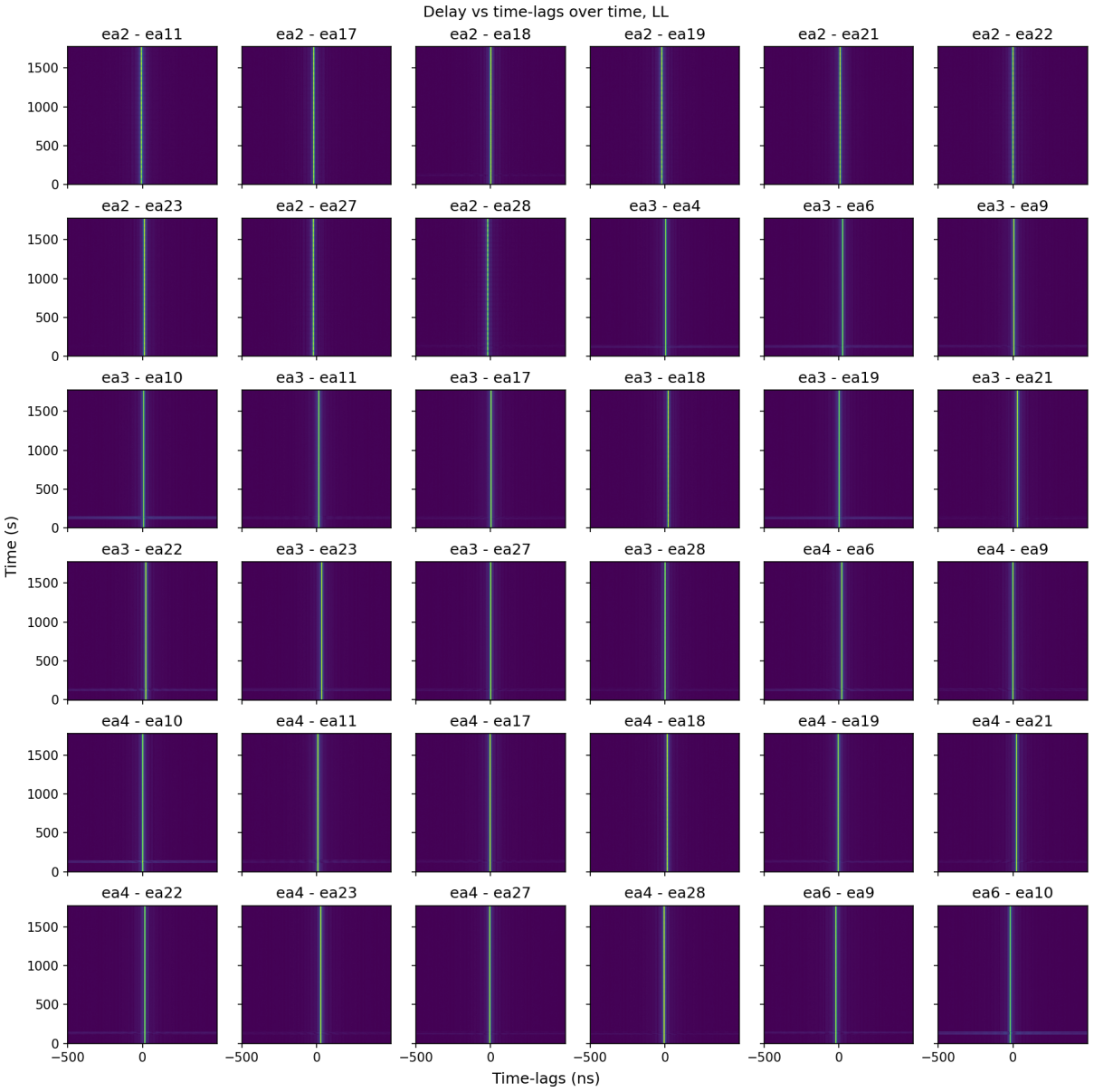
The calibrate_uvh5 class has several methods which can collect the data from uvh5 files and makes a different number of diagnostic plots.

Testing delay tracking:

The plot below shows the residual peak delay measured as a function of time for different baselines showing consistency of the delay tracking. The observations of 3C48 shows RR and LL data products. Once the delay tracking is applied the residual delay values are less than 50 ns which are later corrected in FPGAs in subsequent calibration observations.

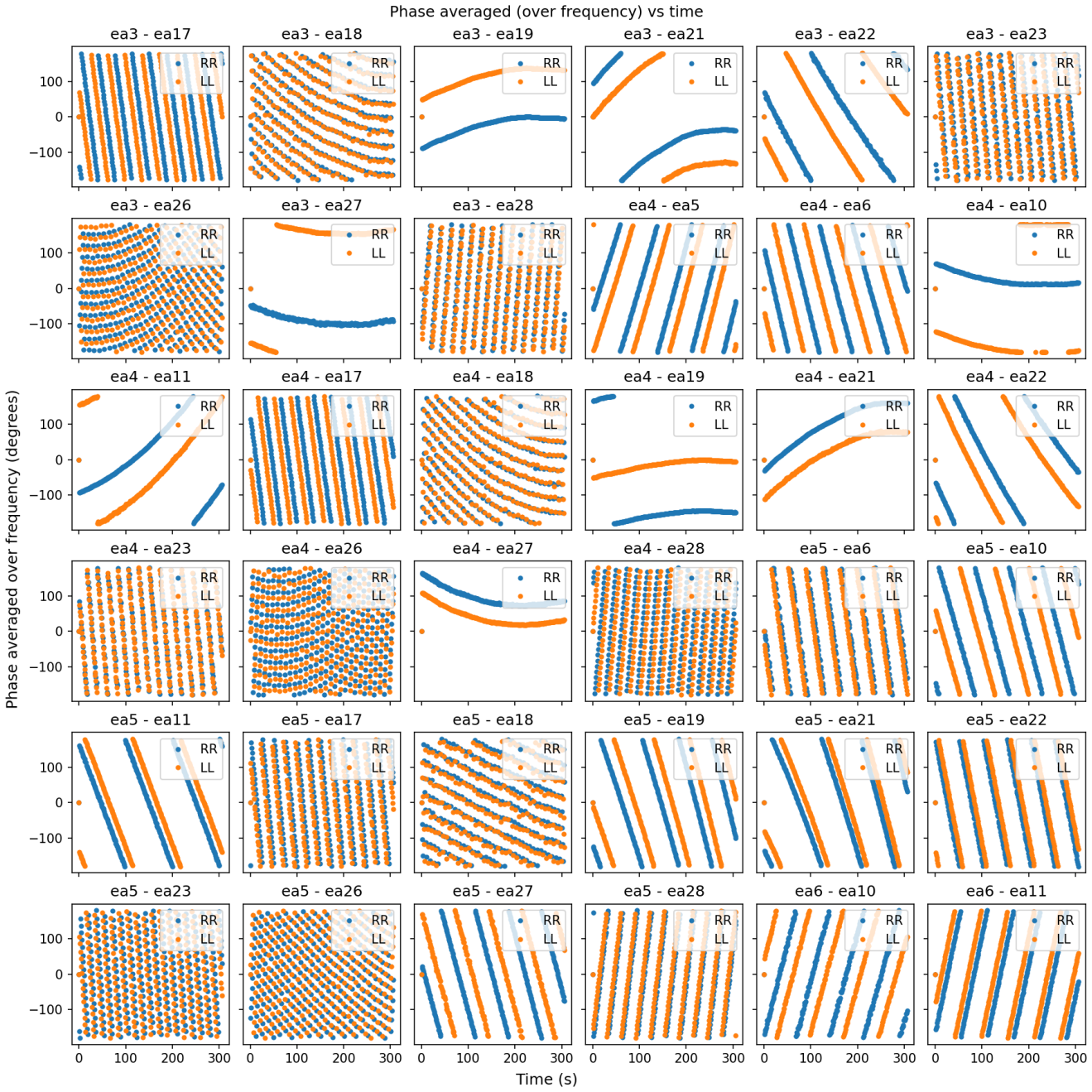

Delay peaks vs time, delay resolution: 0.868 ns
ea2 - ea11
ea2 - ea17
ea2 - ea18
ea2 - ea19
ea2 - ea21
ea2 - ea22
ea2 - ea23
ea2 - ea27
ea2 - ea28
ea3 - ea4
ea3 - ea6
ea3 - ea9
ea3 - ea10
ea3 - ea11
ea3 - ea17
ea3 - ea18
ea3 - ea19
ea3 - ea21
ea3 - ea22
ea3 - ea23
ea3 - ea27
ea3 - ea28
ea4 - ea6
ea4 - ea9
ea4 - ea10
ea4 - ea11
ea4 - ea17
ea4 - ea18
ea4 - ea19
ea4 - ea21
ea4 - ea22
ea4 - ea23
ea4 - ea27
ea4 - ea28
ea6 - ea9
ea6 - ea10
RR
LL
Delay peak (ns)
Time (s)

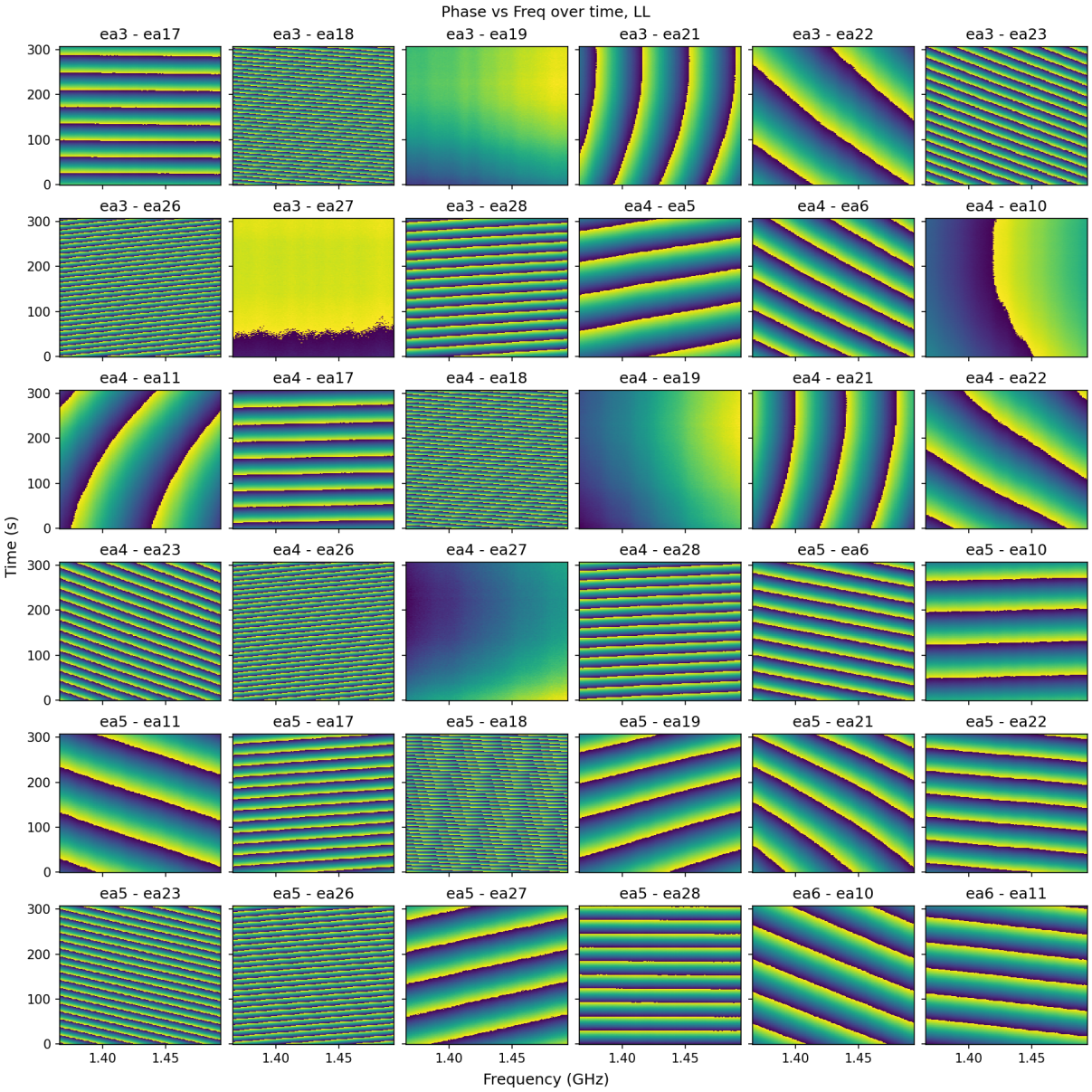
The plot below shows the delay waterfall (delay measured for each time integration) plot for baselines showing bright and constant delay across time.



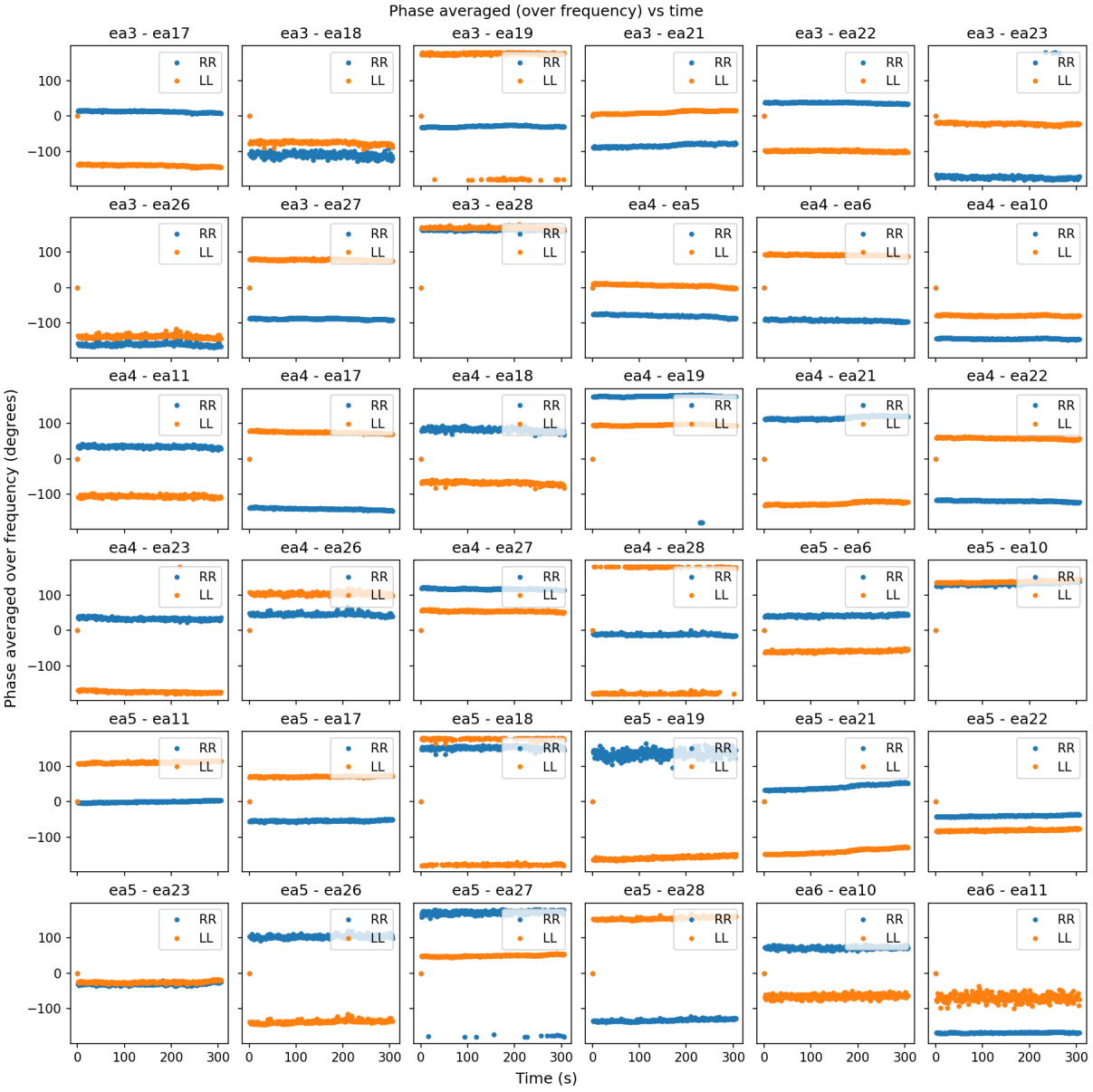
Even though the delays are calculated at the sky or RF frequencies, the antenna electronics operate at the IF frequencies and they are applied at IF frequencies. This introduces a frequency independent phase shift over time. This causes a shift of phase fringes over time commonly known as fringe rotation or fringe oscillation. This needs to be corrected in the voltage domain before the correlation happens.

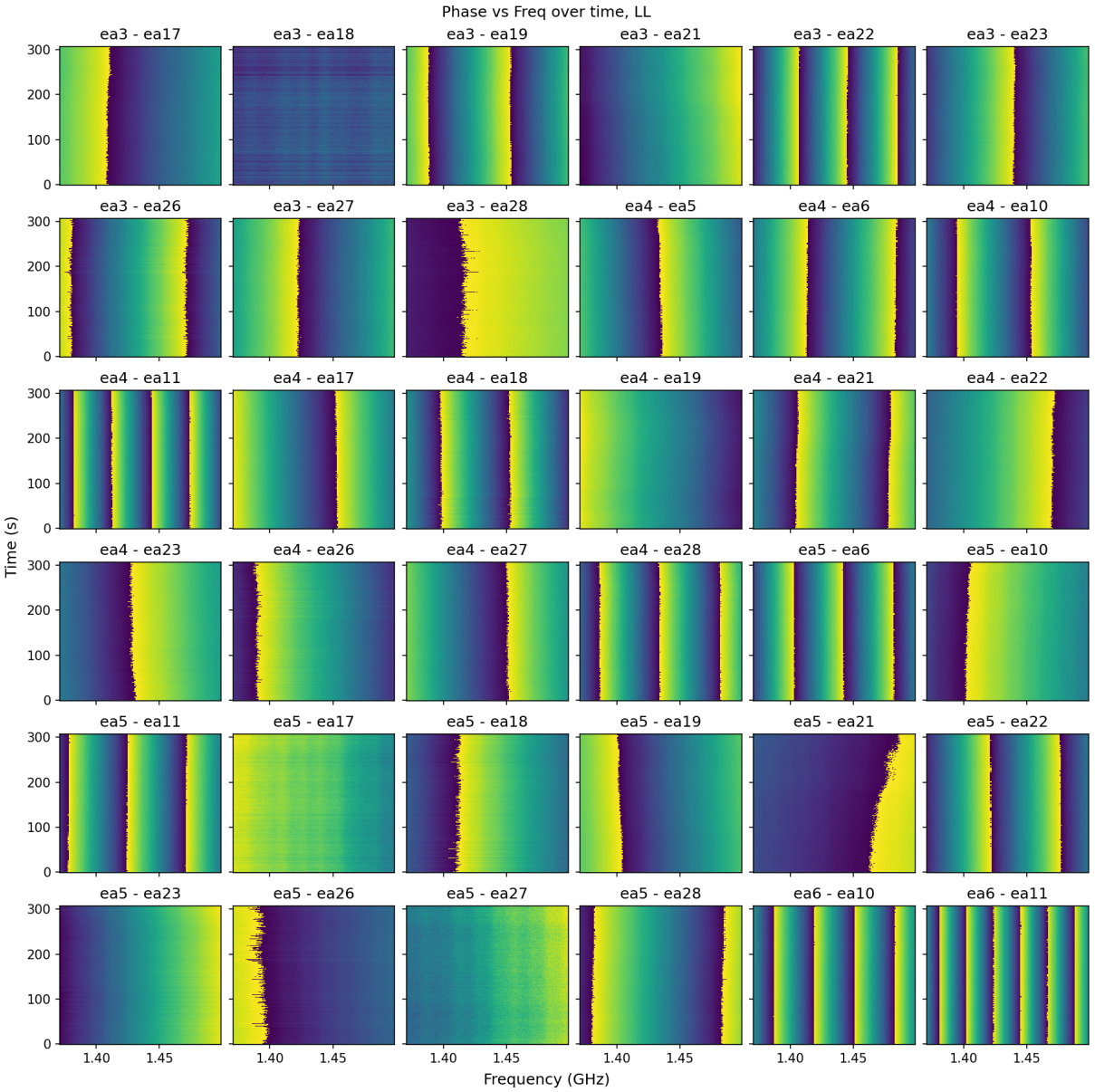
The plot below shows the phase averaged across frequency as function of time and the phase waterfall plots before stopping fringe rotation or the without phase tracking. The phase changes as a function of time and this is problematic as we need phase stability to conduct calibration and imaging.




Phase vs Freq over time, LL
ea3 - ea17
ea3 - ea18
ea3 - ea19
ea3 - ea21
ea3 - ea22
ea3 - ea23
ea3 - ea26
ea3 - ea27
ea3 - ea28
ea4 - ea5
ea4 - ea6
ea4 - ea10
ea4 - ea11
ea4 - ea17
ea4 - ea18
ea4 - ea19
ea4 - ea21
ea4 - ea22
ea4 - ea23
ea4 - ea26
ea4 - ea27
ea4 - ea28
ea5 - ea6
ea5 - ea10
ea5 - ea11
ea5 - ea17
ea5 - ea18
ea5 - ea19
ea5 - ea21
ea5 - ea22
ea5 - ea23
ea5 - ea26
ea5 - ea27
ea5 - ea28
ea6 - ea10
ea6 - ea11
Time (s)
0
100
200
300
1.40
1.45
Frequency (GHz)

The plots below show the averaged phases over frequency and phase waterfall as a function of time with phase tracking / correcting the fringe rotation. We can clearly see consistent phases over time in the averaged plot and the waterfall plots.





Once the delay tracking and phase consistency is achieved, the correlated data is good to derive the calibration and can be used for imaging purposes.

5. Gain Calibration

There are different types of calibration when it comes to correlated data. The data product at this point is mostly delay calibrated. Still there could be delays on the order of couple of ns which needs to be corrected. The first priority of COSMIC is detection of an ETI signal and the only calibration that we focus at the moment is the complex gain or 'phase' calibration. For the same reason, we are not conducting any bandpass or the amplitude calibration in real time. However the correlated data of amplitude and bandpass calibrators are saved in case we need to calibrate the bandpass response and amplitudes post ETI detection.

The gain calibration is usually antenna based and mainly accounts for the time variable factors associated with the instrument (power level setting within the receivers, changes in antenna gains, etc) and the atmospheric effects (troposphere at high frequencies, ionosphere at low frequencies, water content, solar flares, etc).

Here we are using a modified version of the Sdmpy package which was initially written to calibrate the VLA Science Data Model (SDM) datasets. This codebase was modified to ingest COSMIC data and derive calibrations. The derive_gains method in the calibrate_uvh5 class of https://github.com/COSMIC-SETI/COSMIC-VLA-CalibrationEngine/blob/rfi_mitigation_and_arrayconfig_update/calibrate_uvh5.py is used to derive the antenna gains. The cross correlation / visibility matrix is modeled as the outer product of the gains from the antennas. First the data matrix is converted from baseline axis to antenna axis. Then an eigen value decomposition is used to calculate the eigen vectors and eigen values which is later used to get the gain values per antenna, per time, per frequency and per polarization.

Deeping dive into the mathematics:
The data matrix is converted to a reduced matrix using the low rank approximation technique. The largest eigen value and the corresponding eigen vector from the eigen value decomposition of the data matrix is used to create the reduced data matrix. The autocorrelation values in the data matrix before decomposition are zero filled for this process. However they are replaced with the values from an iterated reduced matrix (3 times).
Every complex matrix can be decomposed into its eigen values and eigen vectors using singular value decomposition and an approximate reduced matrix can be generated using highest eigen values and the corresponding vector as shown below.

$V_{obs} = U * \lambda U = U_1 * \lambda_1 U_1$ (reduced matrix)

$V_{ij} = \lambda_{ij} u_i * u_j$ (1) (single baseline visibility)

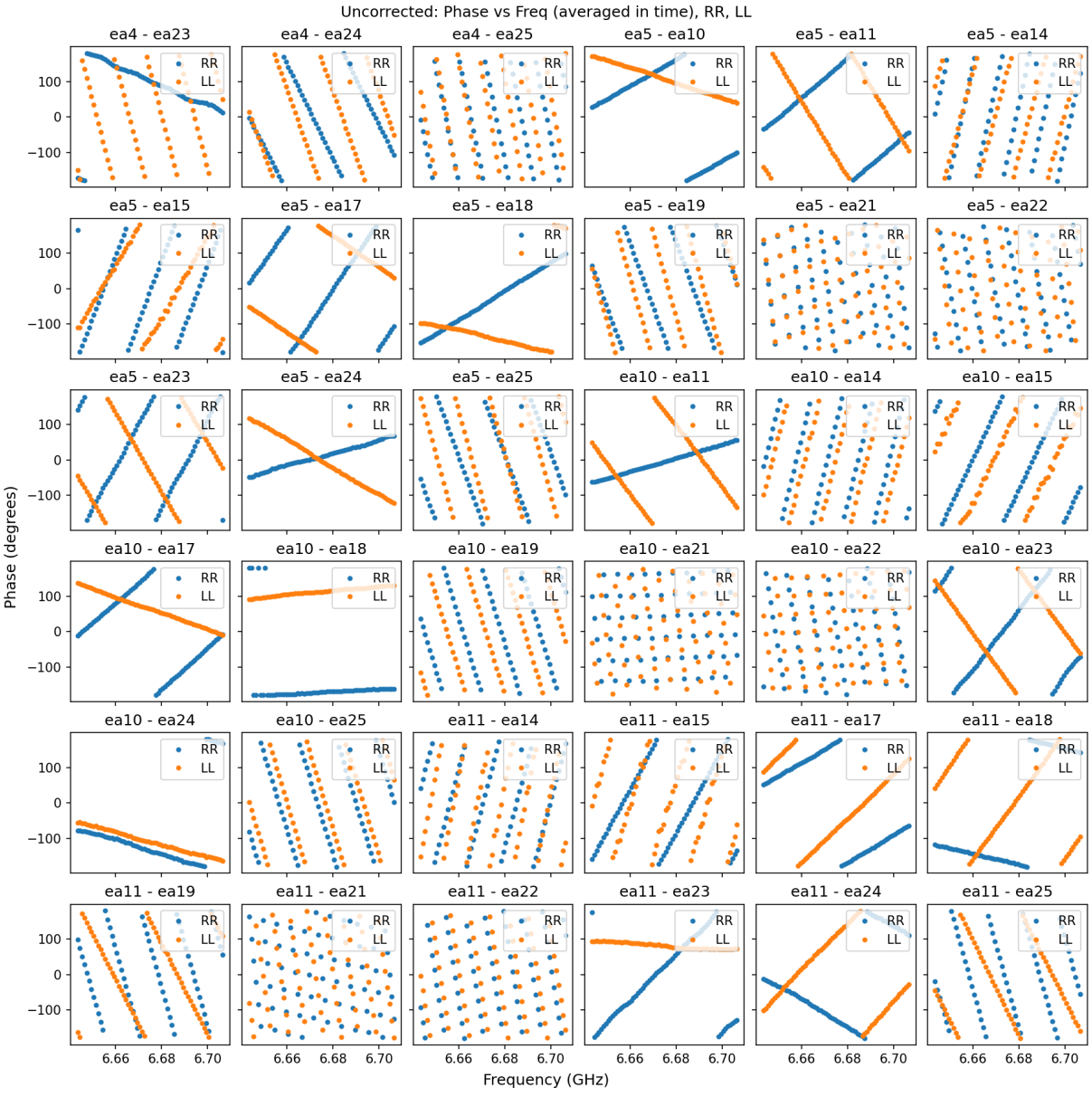
If we have a point source, their visibility for a baseline can be written as

$$Vij \quad = \; So\, Gi\; Gj\, * \quad = \sqrt{So}\, Gi\, \sqrt{So}\, Gj\, * \qquad (2)$$

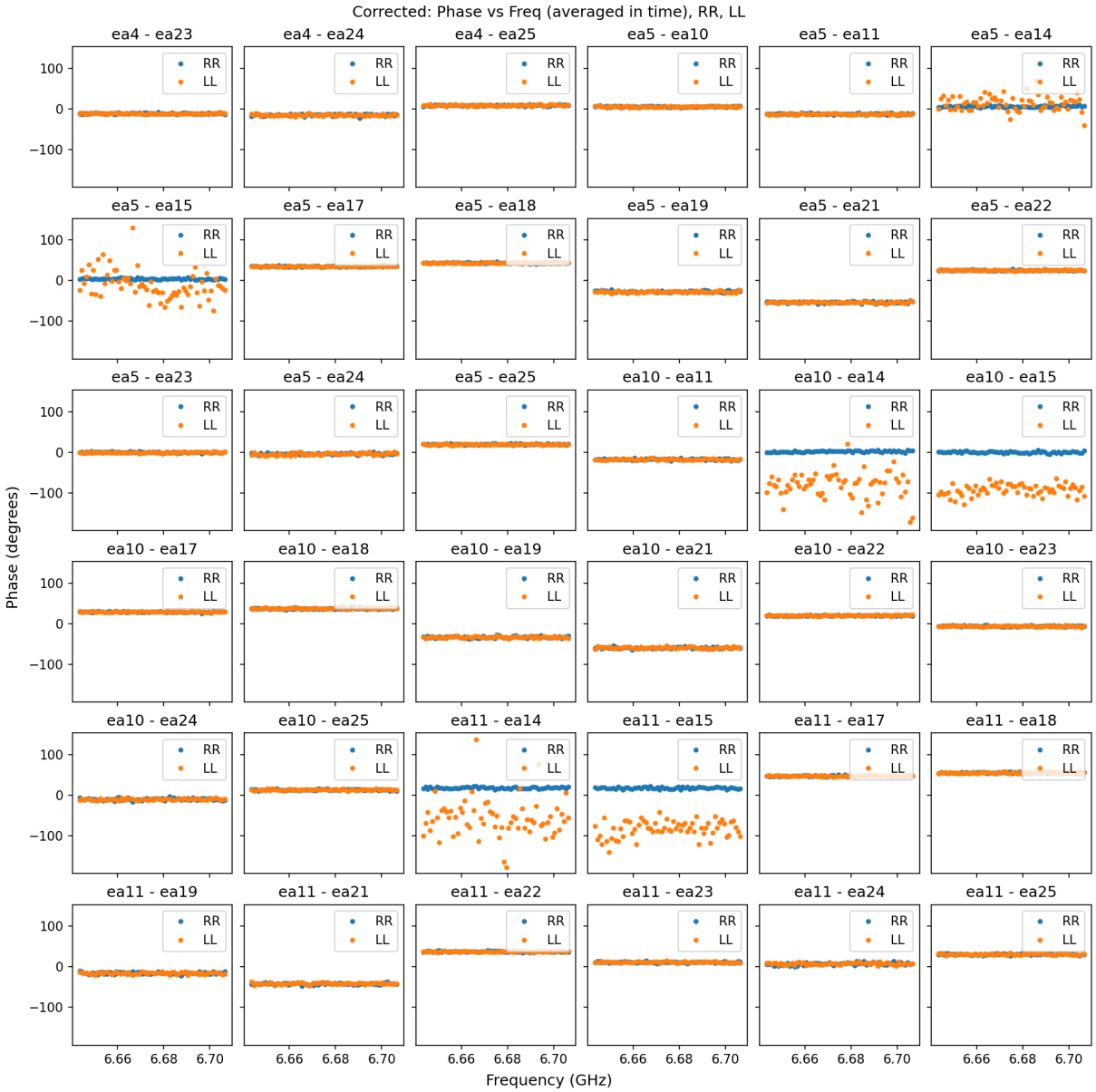
Where V is the visibility, So is the flux density and G are the gains of each antennas
Or in other words, a single baseline response to a point source ideally has zero phase and constant flux density. Any variation from it can be modeled as a gain variation for an antenna multiplied by the constant flux density parameter. If we compare equations 1 & 2, gains of each antenna are equal to the reduced eigen vectors.

I guess there is a simple way to explain this whole thing: More questions ask Paul Demorest.
Does this work ? Yes, it works. The plot below shows the phase vs frequency for different baselines before and after gain calibration.

Before Calibration:



The plots phase vs frequency plots after calibration has flat and zero phases which is shown below.



6. Offline calibration and First images from COSMIC

Before online calibration, the gain calibration algorithm was tested using a back to back observation of 2 close calibrators. For this purpose, we conducted a 3 minute observations of 3C286 and J1310+3220 bright sources.

Tests with gain calibration code:

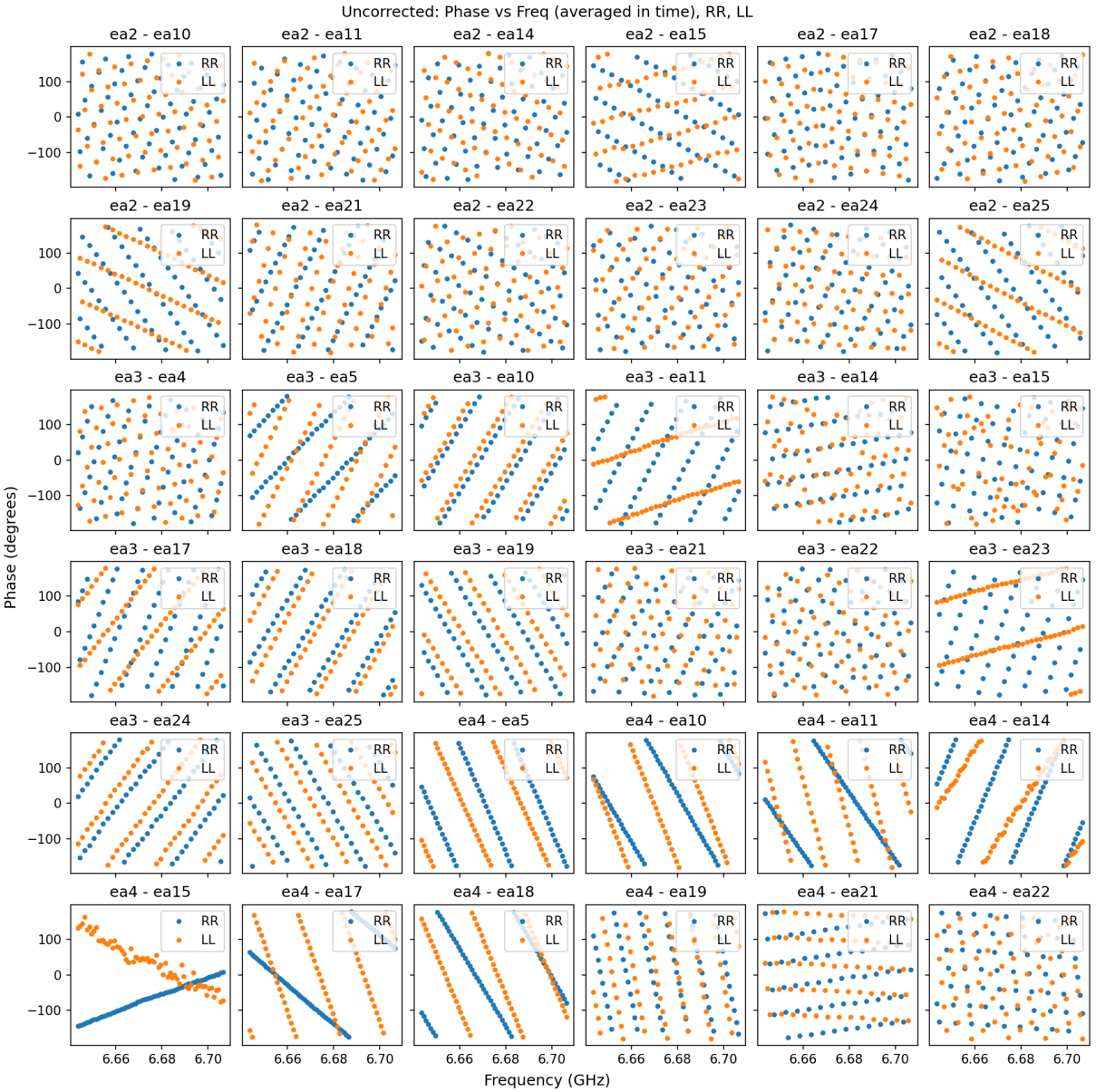
1. Derive the gain calibration from 3C 286 and apply it back to the same source.

2. Apply the derived gain calibration from 3C 286 to the J1310+3220
3. Look at the phase solutions in each case
4. Tests with CASA
    a. Conduct calibration and imaging of each datasets independently in CASA

Test1:

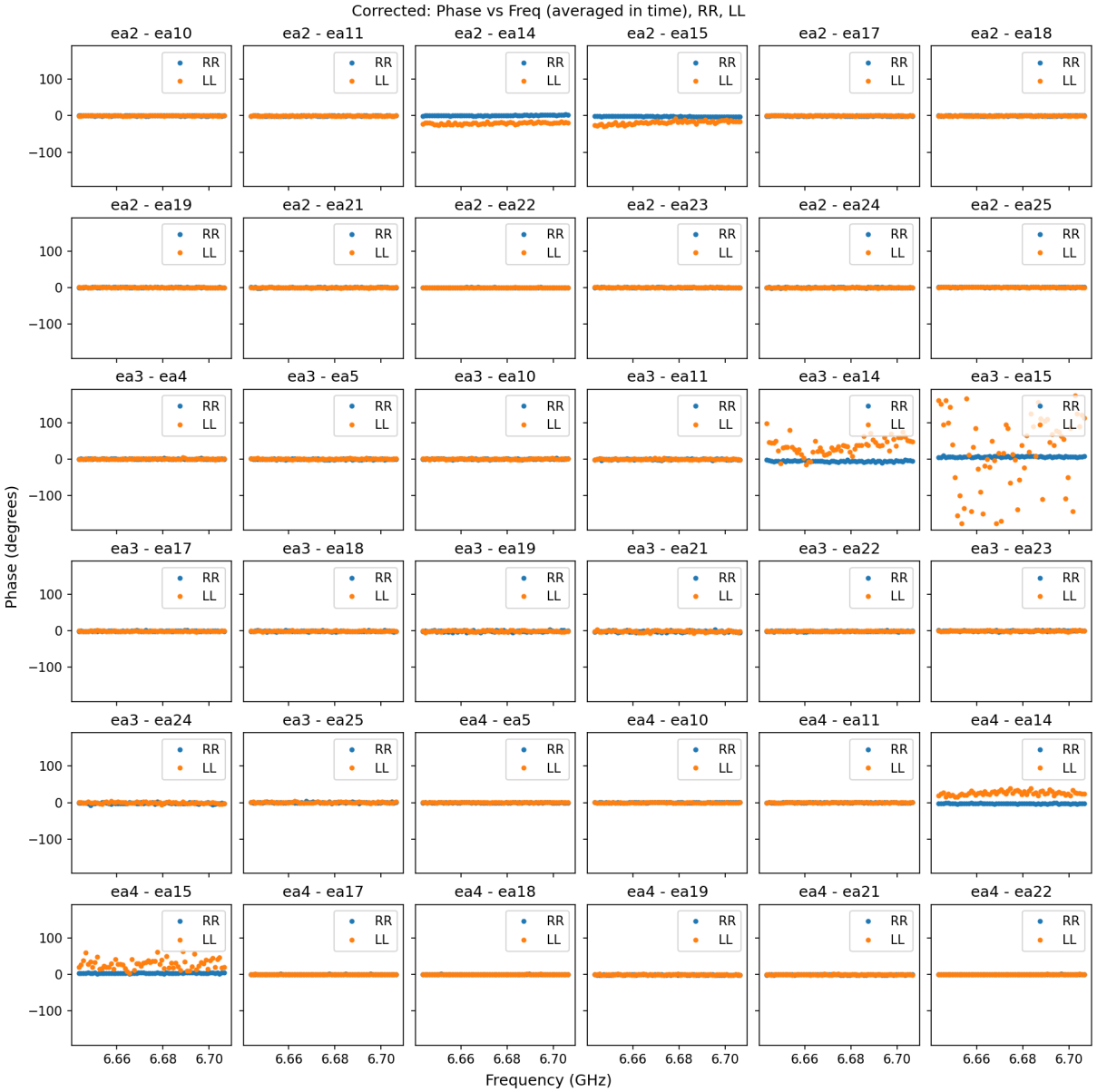
Plot below shows the phase vs freq for each baseline averaged over time before calibration on 3C 286 showing good fringes.

Before Calibration:



Now the gain calibration derived from 3C286 is applied on the same dataset. The phases vs frequency plot after applying the correction is shown below for all the baselines. The phases are flat and zero for both RR and LL in most of the baselines. The phases deviates from zero in baselines with bad fringes which could be due to RFI or data recording issues in certain antennas.

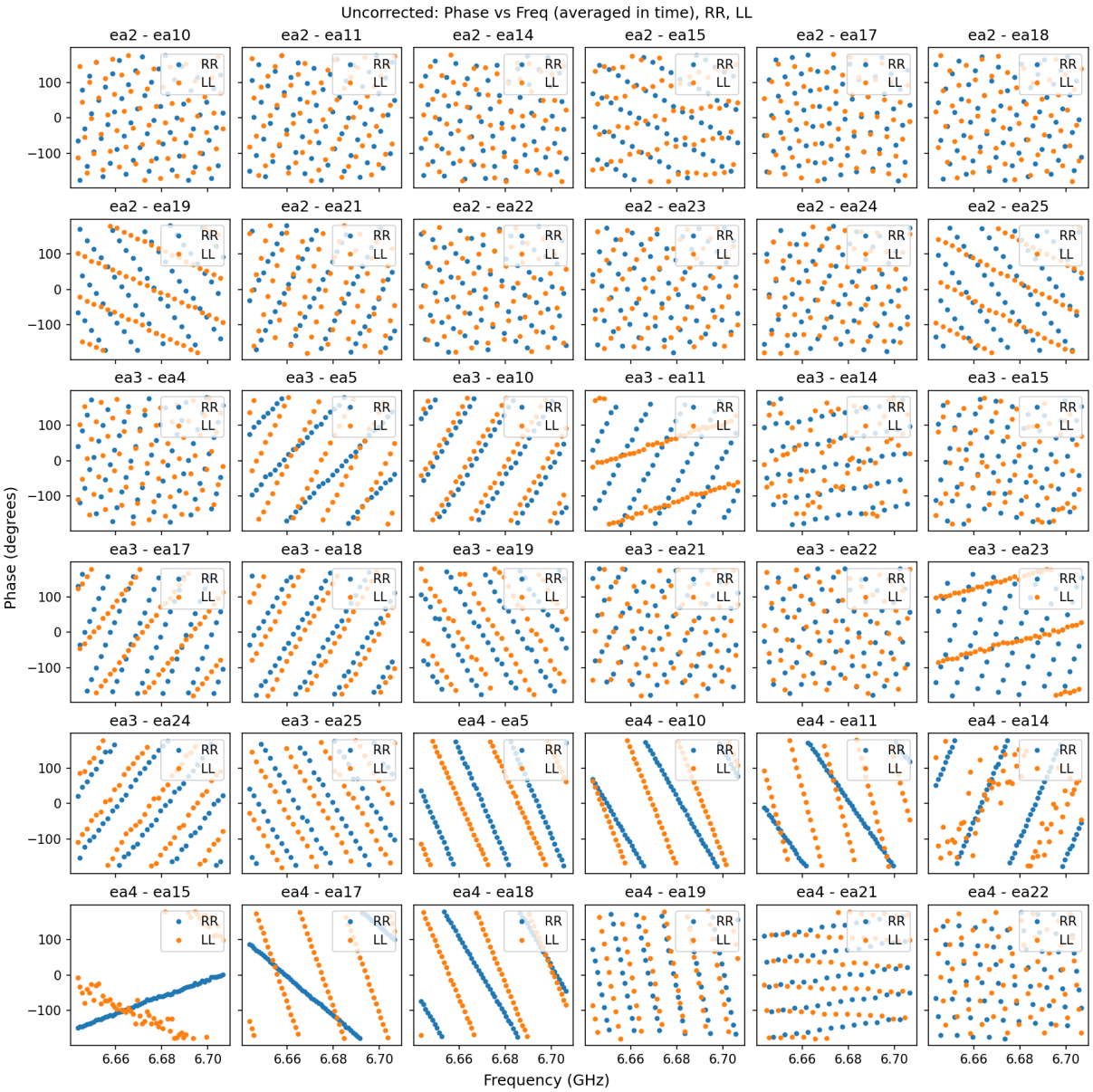
After Calibration:



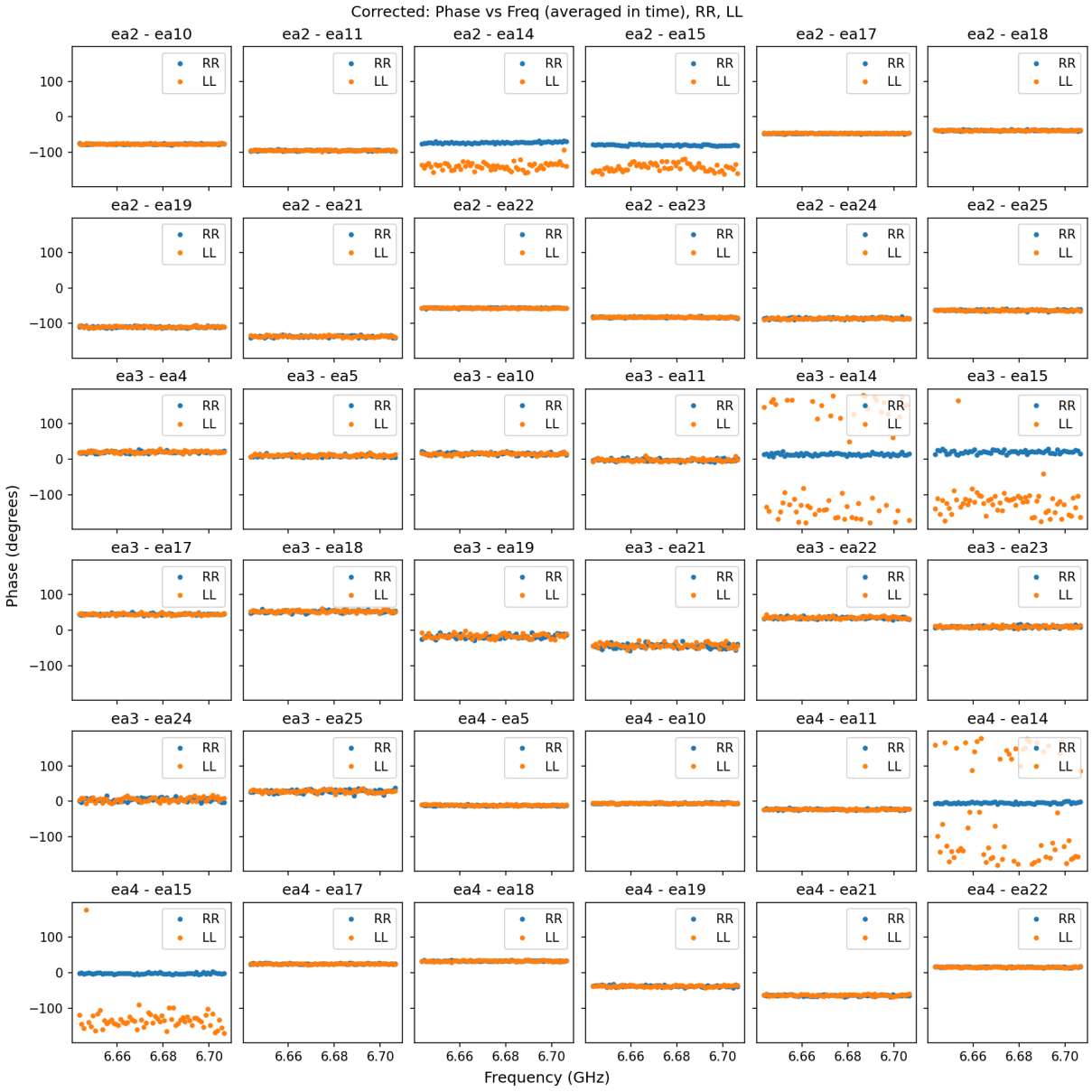
Test 2:

Plots below shows the phases vs freq for J1310+3220 before calibration and after applying the derived calibration from 3C 286.

Before calibration:

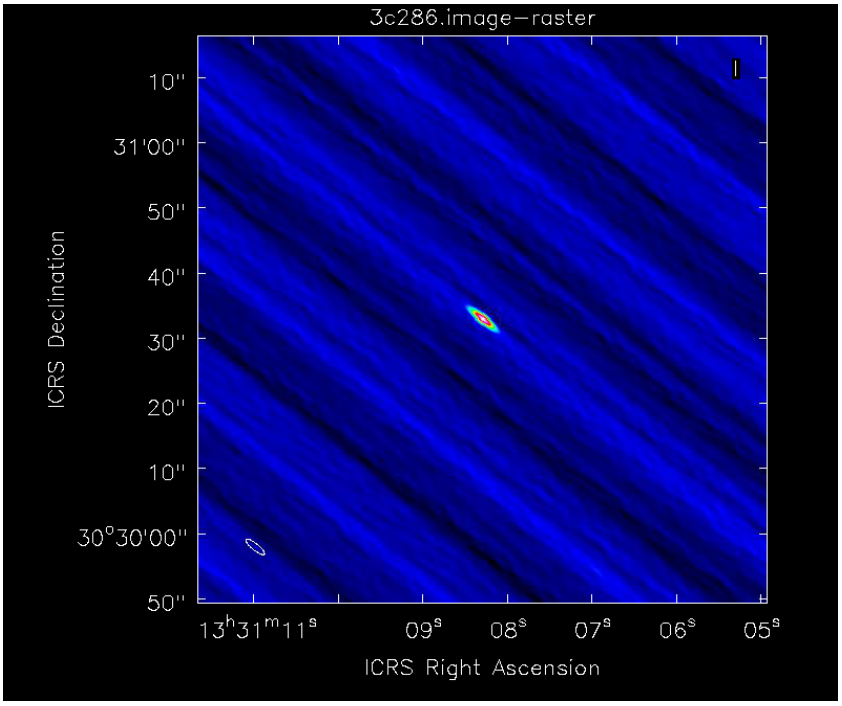


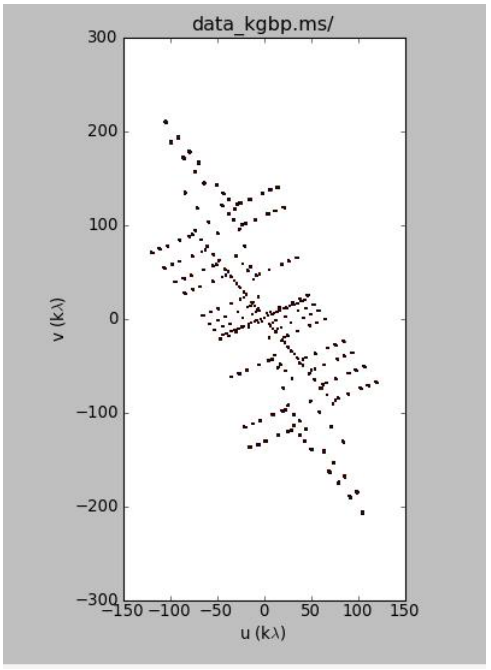
After calibration: The phases are flat and zero across many baselines. However, in some baselines phases deviates from zero. This means the gains solutions from one source to another source are transferable but not perfect. But still good enough to conduct the imaging.

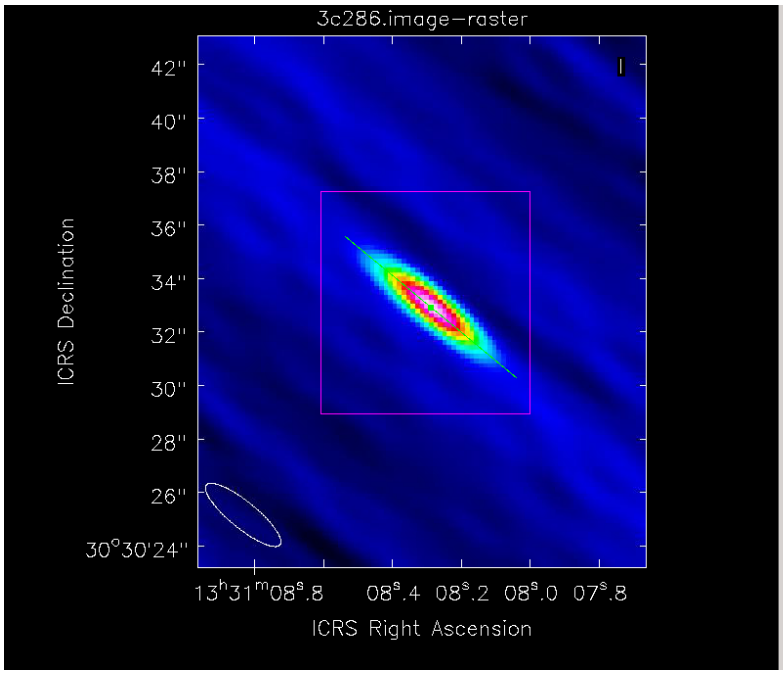


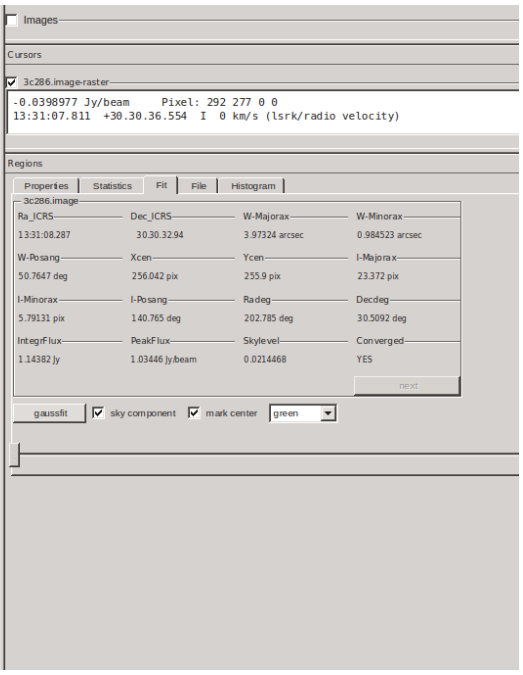
Test 4: The same datasets of 3C286 and J1310+3220 in UVH5 format were converted to measurement sets and calibrated in CASA to compare the results with the cosmic gain calibration code. Here, instead of a single step of gain calibration, a combination of delay, bandpass and gain calibration was conducted in CASA. The solutions derived from 3C 286 were applied to the same dataset and the J1310+3220 dataset. Then datasets were imaged using the CASA clean task. The plots below shows the images of 3C 286 and the corresponding the UV coverage.

Images of 3C 286:





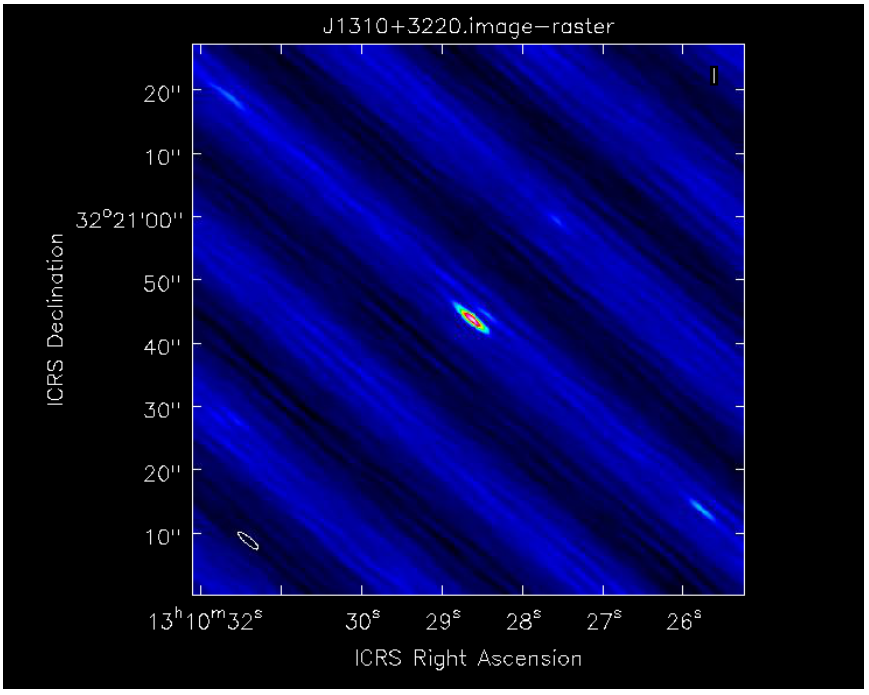


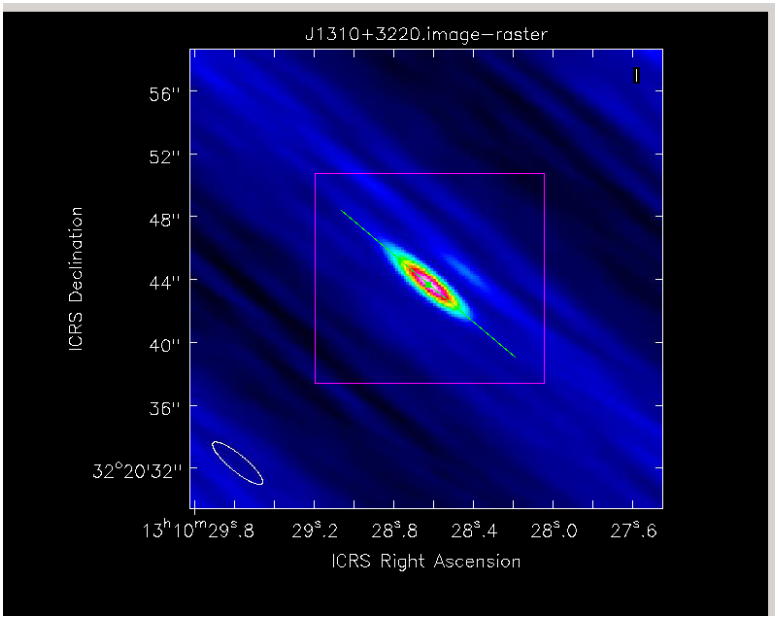


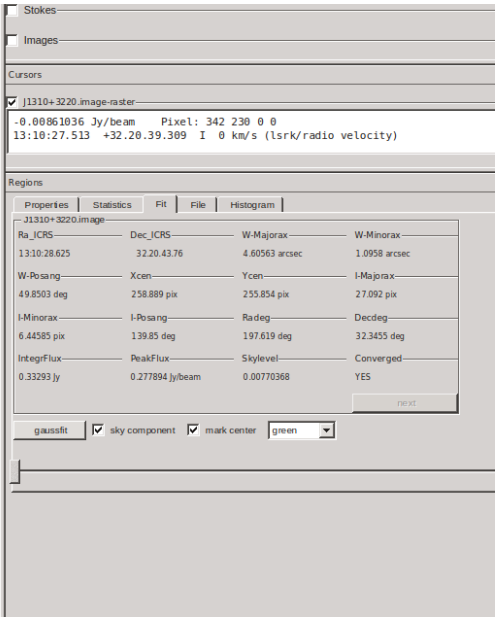
The data was collected at 6.5GHz for ~140 seconds. One bad antenna and a couple of bad baselines with delay tracking issues were flagged. This resulted in a total of 16 antennas after flagging with 32 MHz of bandwidth. The UV coverage is extended in one direction due to limited time, bandwidth and number of antennas. The clean task was used with a deconvolution of 500 iterations, a cell size of 0.17 arcsec (5 cells across the synthesized beam of 0.86 arcsec) and an image size of 512 pixels.

3C 286 looks like a pointed source, but slightly extended in one direction basically due to the poor UV coverage. The ripples seen across the image are also due to the bad UV coverage. The signal to noise ratio of the source is really good as evident from the image. After doing a 2D gaussian fit on the source, fitted coordinates (13h31m08.287, +30d30m32.94) match pretty well with the NED values (13h31m08.2879s, +30d30m32.958s)

Images of J1310+3220:





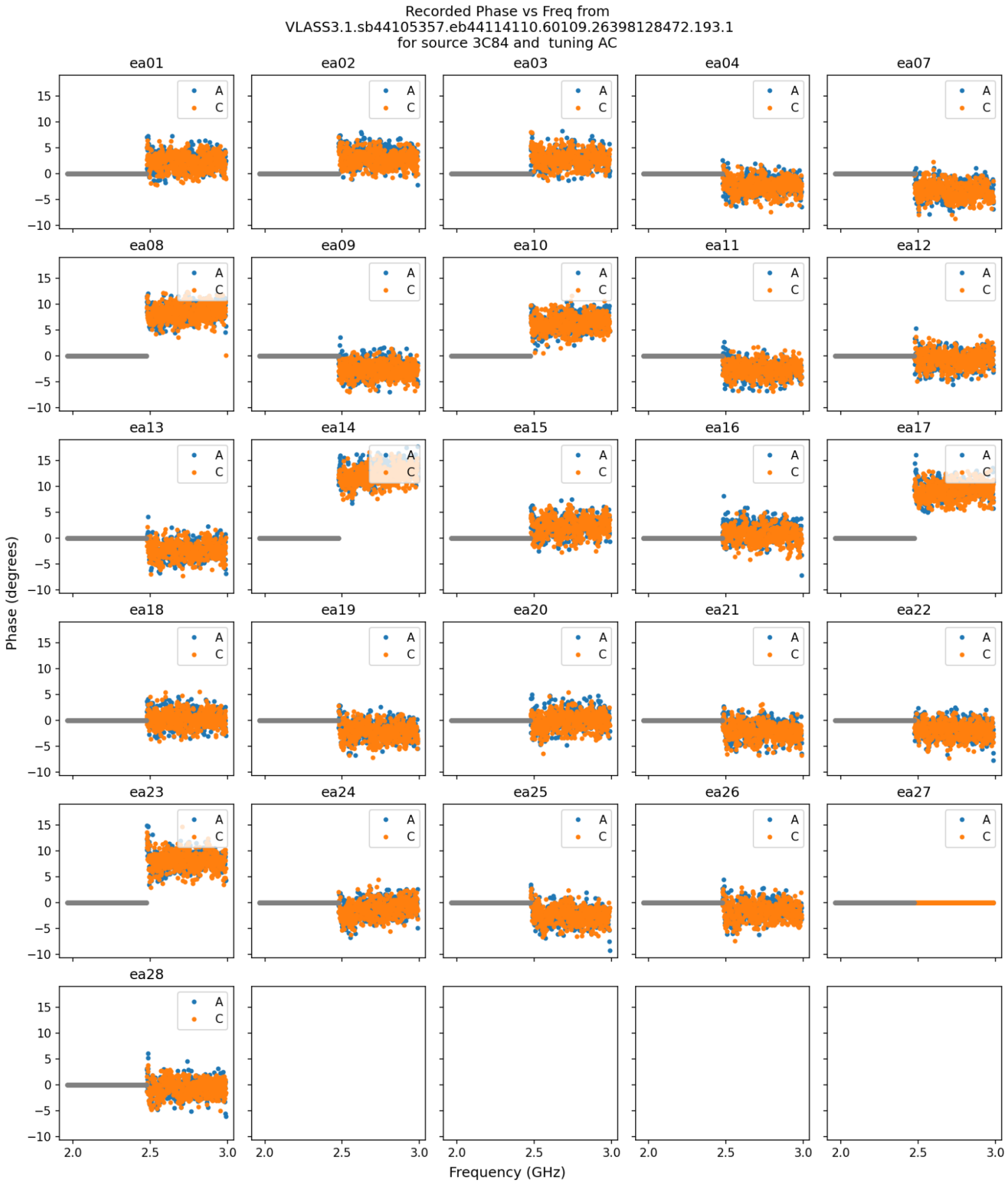


The calibrations derived from 3C286 are applied to the J1310+3220 dataset and it looks good with point source structure, but extended in one direction due to bad UV coverage. The 2D Gaussian fit gives an Ra, Dec [13:10:28.625, 32:20:43.76] and the actual values of radio position from NED are [13:10:28.6638, 32:20:43.783]. These values agree pretty well with each other (only 0.3" difference). This shows that the COSMIC data can be calibrated well with the cosmic gain calibration script and CASA. The astrometry of the calibrators from the back to back observations are pretty good and within 0.3 arcsecs.

7. Online or Real Time Calibration

Once the offline calibrations were successful, then back to back sources were observed and the calibrations were applied in real time. In order to conduct real-time calibration, the complex gains collected from each compute node are sent to the head node, where it is collated across frequencies in both tunings. A previous phase applied to the FPGAs is subtracted from the gain solutions. Then an IFFT of the gains will give the residual delay per antenna. Then, a phase factor corresponding to the residual delay is subtracted from the gains to get the phase solutions. After this stage, the derived residual delays and phases are applied to FPGAs in real time simultaneously. The residual delays and phases gets updated in the subsequent calibration runs

This part was mostly done by Talon, Jack and Paul. I tested the same algorithms on a couple of datasets and they all looked to work fine. However, a spread of phases was noted in some subsequent calibration runs due to unknown reasons. The phases are reset once the dataset ID changes during observations. This removes the bad phases propagated from bad observations with technical issues or the ones affected by RFIs. The calibration slack channel produces a phase vs freq for the gains received per antenna from a calibration dataset which gives an instantaneous view of the quality of calibration. The plot below shows such a plot from the slack channel showing the phases of the gains per antennas in the AC tuning. The following data correspond to a RFI free VLASS observation and it shows close to zero flat phases across frequency which is what we want. Here AC refers to a signal pair, where A is the left circular polarization and C is the right circular polarization. Similar results are observed for back to back observations of nearby calibrators as well.

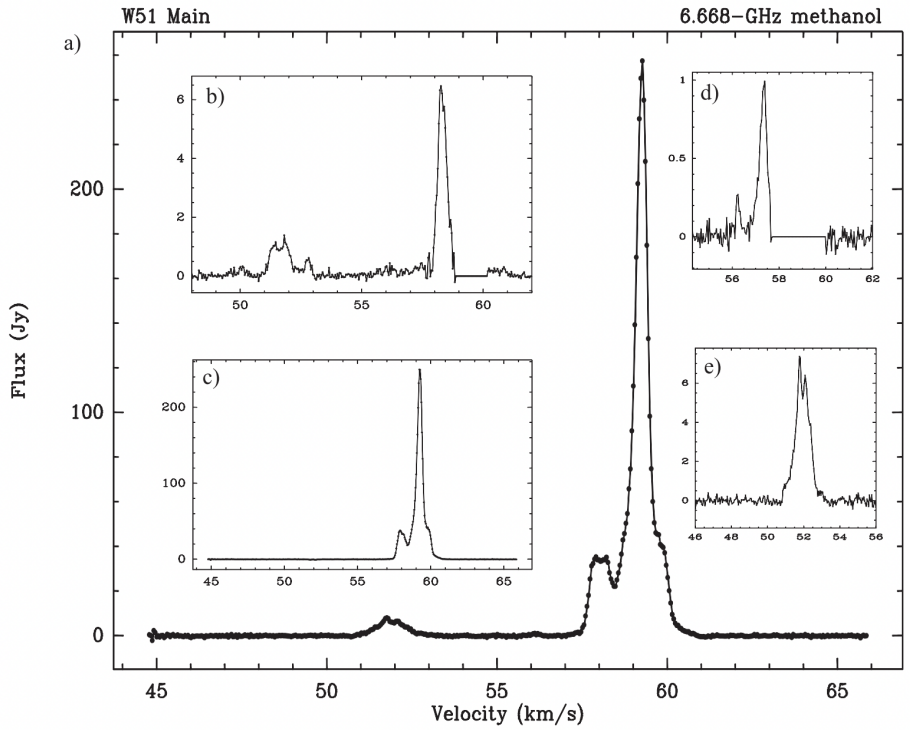

Recorded Phase vs Freq from
VLASS3.1.sb44105357.eb44114110.60109.26398128472.193.1
for source 3C84 and tuning AC
ea01
ea02
ea03
ea04
ea07
ea08
ea09
ea10
ea11
ea12
ea13
ea14
ea15
ea16
ea17
ea18
ea19
ea20
ea21
ea22
ea23
ea24
ea25
ea26
ea27
ea28
A
C
Phase (degrees)
Frequency (GHz)

8. Commissioning the BLADE beamformer code

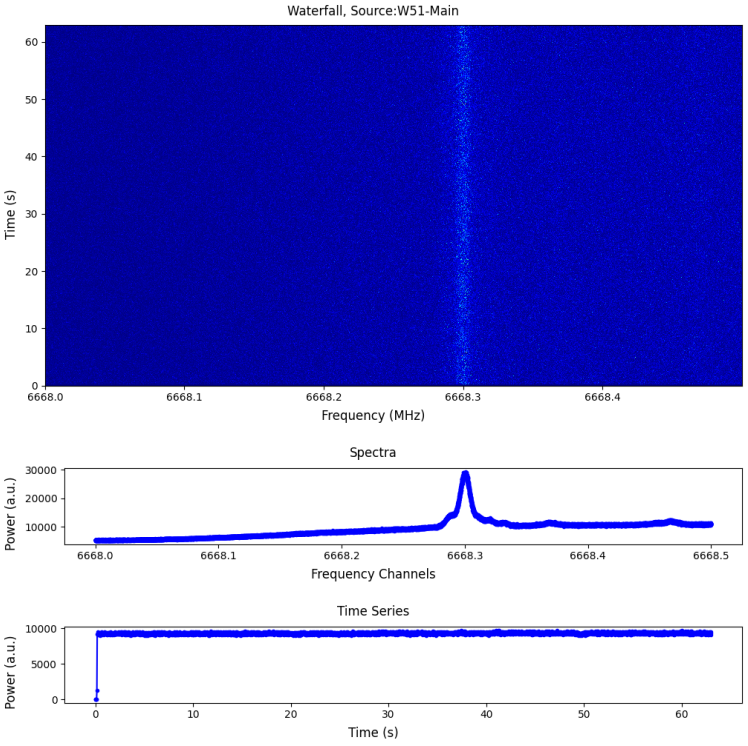
BLADE is the production ready real time beamformer code written in CUDA C++ for COSMIC. A good documentation of BLADE can be found here. Some of the commissioning tests of BLADE conducted by Chenoa Tremblay is listed in another document. In the initial days of testing, the BLADE code was not production ready and several fixes were needed in the software. This is the time I realized, there are many ways a beamformer code can go wrong.

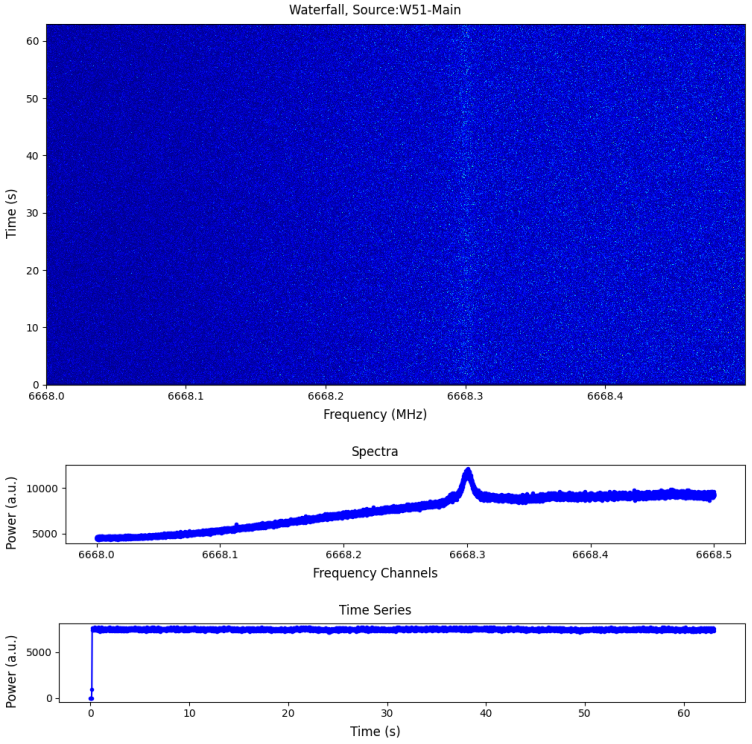
In order to test the findings of the BLADE, I wrote a prototype python based beamformer which is available here. The beamformer takes in the BFR5 file which contains the antenna locations, frequency information, beamforming coefficients, the coordinates to form the beams and the Guppi raw file stem. The results from the prototype code is compared with BLADE to measure the consistency between the 2 codebase. This beamfomer ingest one coarse channel of the data (in which you expect the signal), upchannelizes the data, conducts coherent and incoherent beamforming and produces per antenna spectra, integrated spectra, waterfall and raster scan plots.

Tests with Methanol Maser:
In order to test the accuracy of the beamformer, we observed a bright W51 methanol maser at 6 GHz, which has a flux density of ~ 274 Jy and ~0.5 MHz line width. The plot below shows the profile of the maser source from S. Etoka, M.D. Gray1 & G.A. Fuller 2012 and it utilized observations from the Merlin telescope. The plot also shows the spectra of resolved components along with larger spectra showing the main component.

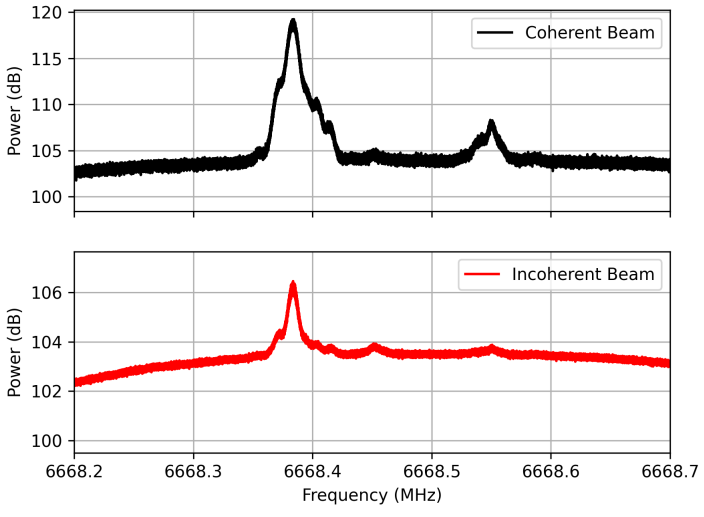


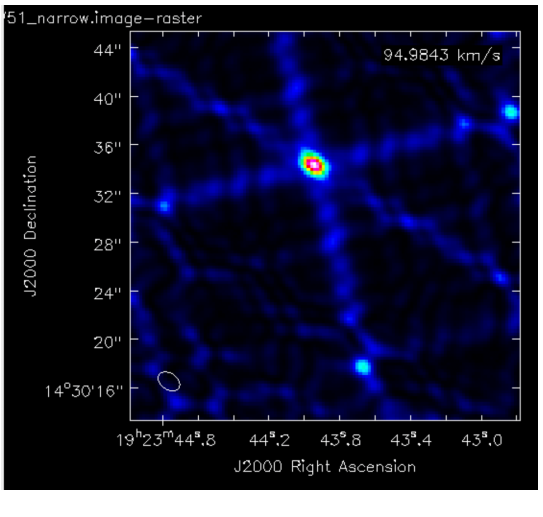
Before running the beamformer on the data, we needed to make sure that all the antennas had detected the maser signal. In order to do that, the associated GUPPI raw files were upchannelized to 30 kHz frequency channels and converted to Stokes I filterbank files for each antenna. The maser source is bright enough to show up in a single antenna data. The output filterbank files showed that the maser was detected in all the antennas. However, some antennas with low gain detected weak pulses compared to the rest of the antennas. The plots (waterfall, integrated spectra and time series) below shows a bright signal detected in ea08 and a weak signal detected in ea10. The initial dip in the time series is associated with some packet loss in the beginning of the recording.





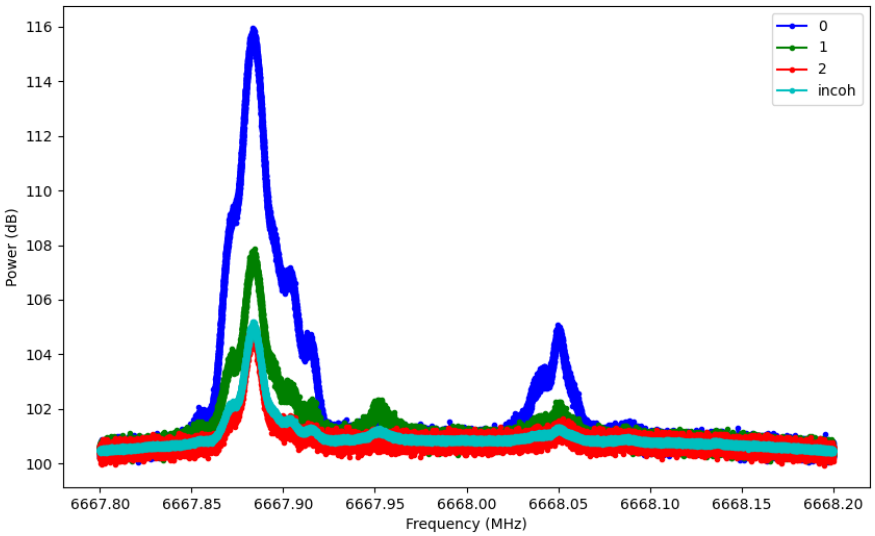
Once we verified that the maser is detected on all antennas, we moved on to the beamforming part. These are the results of BLADE on maser data taken from Cheona's document. The plot below on the left shows the power measured by the coherent beam on the maser position and the incoherent beam. The plot on the right is an image of the Maser made by Chenoa showing a really good bright point source.





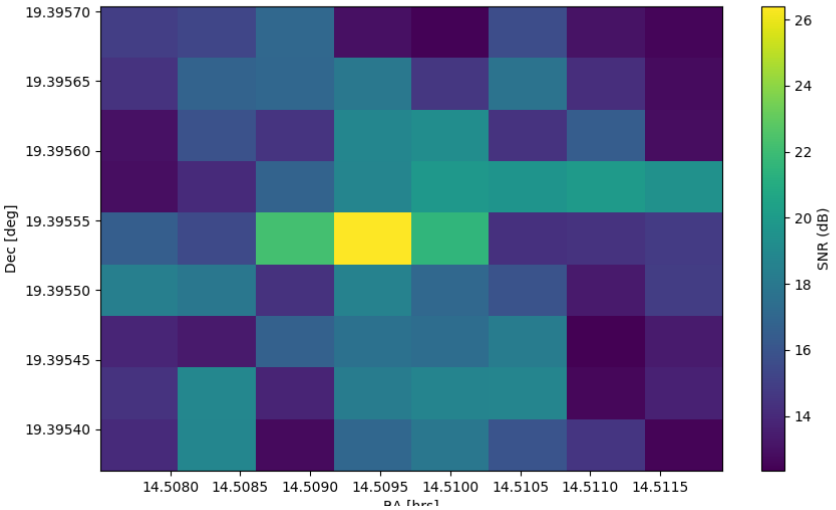
When the observation of the maser took place, the boresight of the telescope was pointed towards the maser coordinates from the literature RA, Dec: (19:23:43.873, +14:30:29.453 J2000). However, due to small calibration errors the peak point of the maser from the image was obtained at RA, Dec (19:23:43.94, +14:30:34.34) which is a couple of arcsec away. At C band and VLA B configuration, this is a couple of beams away and the beamforming coordinates were updated accordingly. The plot below shows the spectra from beams pointed at different locations from the python beamformer code.

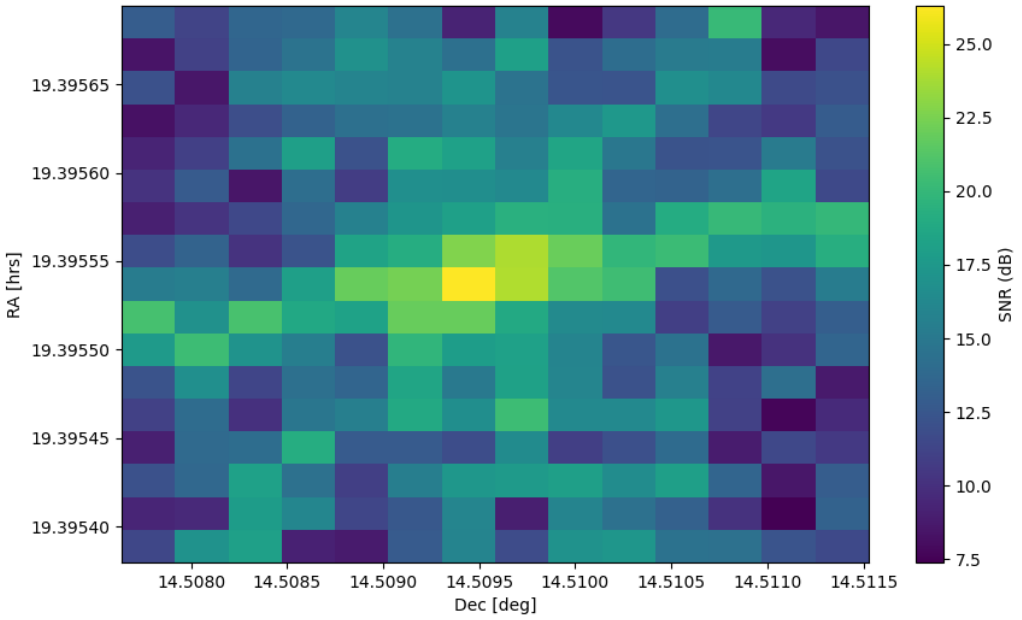
0 = Updated location of the maser source
1 = boresight where we thought maser would be
2 = point at edge of primary beam/FoV
Incoh = incoherent beam



We observe ~11.5 dB difference between the coherent and incoherent beam even though ideally we expect around 13 dB (10*log10(20 antennas)) difference. The incoherent beam could be picking up other point sources within the FoV. The power in the beam couple of arcsecs away is going down by 8 dB and power of the beam at the edge of primary FoV goes down by 12 dB. These results agree with the observations from BLADE.

Along with the multi point beamforming, a raster scan was conducted around the maser to map out the point source structure. The plots below show the power measured in 2 raster scans, first with a lower beam resolution and then a higher beam resolution.





In both the images, we can clearly see a point source and it matches with the image of the maser made from the visibilities. The dirty beam structures associated with the maser are clearly seen in the high resolution map. BLADE was having issues in retrieving the point source structure. After some bug fixes, raster scan from the python code matched with the BLADE version. Currently the BLADE has a pointing accuracy less than ⅓ arcsec.

Beamformer Test with setigen data:

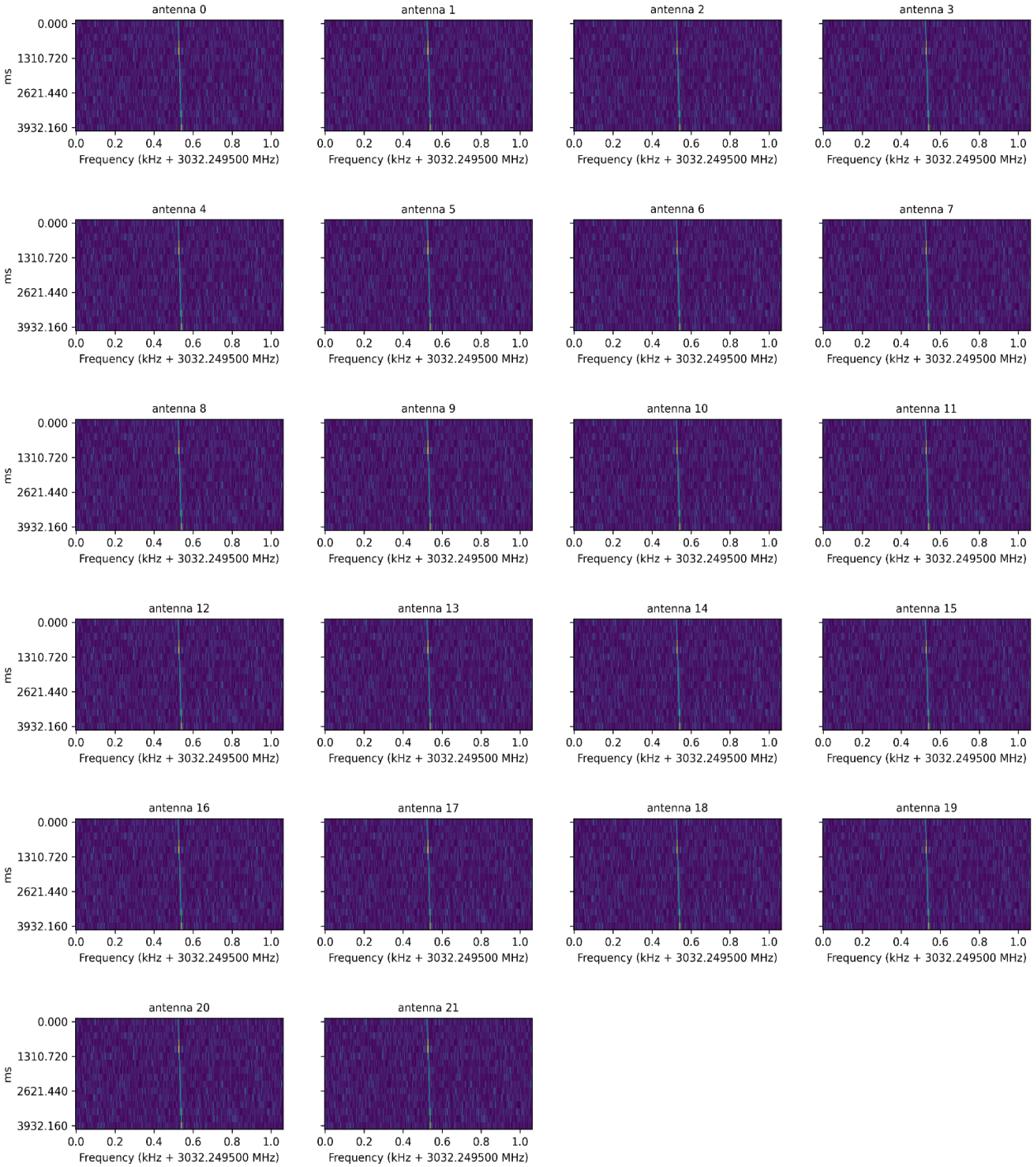
In order to test the efficiency of the beamformer, we conducted tests on simulated data generated by the setigen software which is discussed in the first session. Raw files similar to COSMIC were generated and narrowband drifting signals were injected into the raw files at specific frequencies. The goal was to study how the beamformed signals changes as the beam is pointed at different directions in the sky. During the study we realized that the raw data generated from setigen does not incorporate the effects of sky or the primary beam pattern of the telescopes. Due to this reason, the results from beamforming tests on simulated data were inconclusive.

12. Commissioning the BLADE beamformer search mode using setigen data

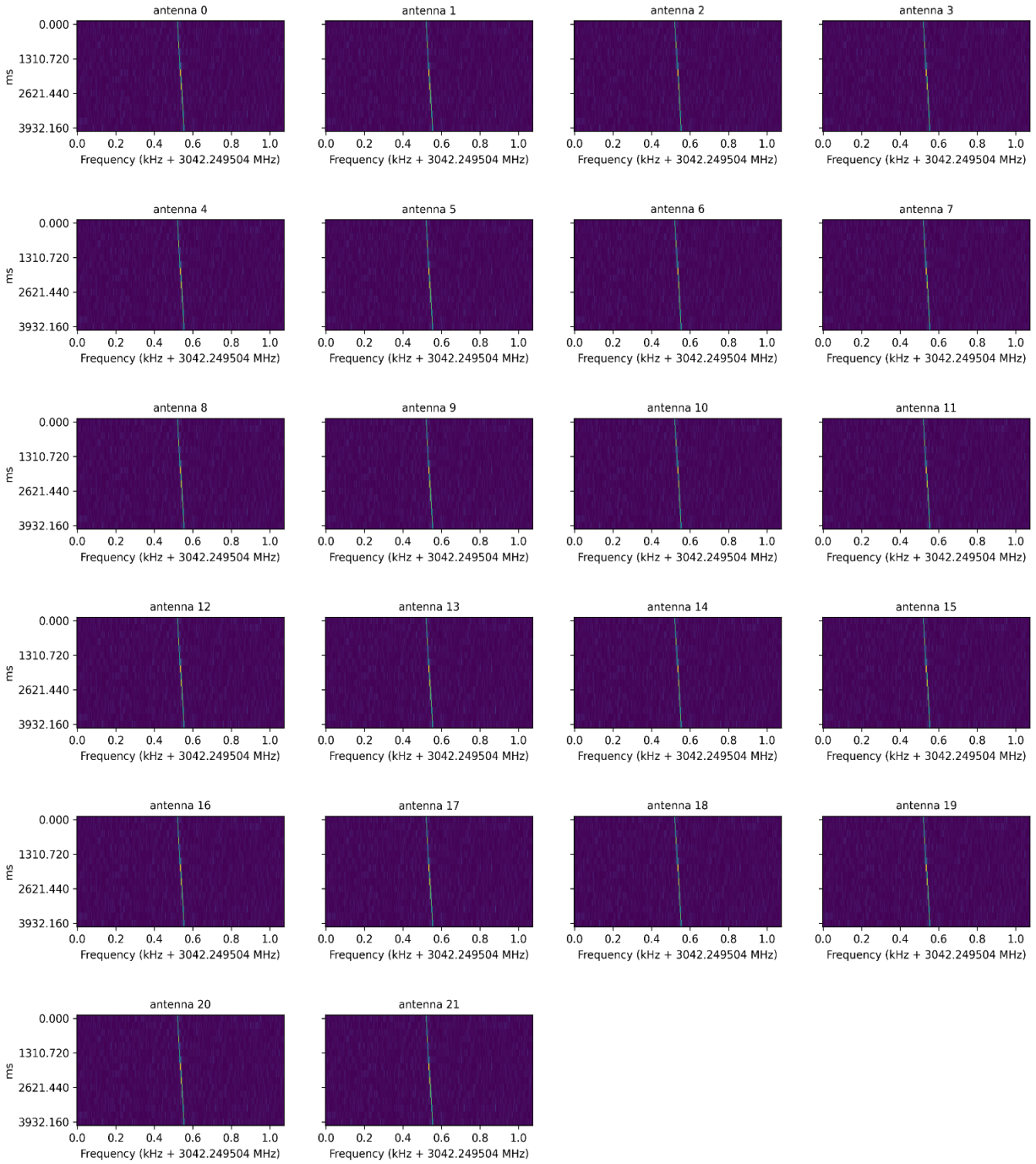
Once the beamformer was commissioned, the next step was to test the beamformer search mode. This mode basically conducts the upchannelization, beamforming, dedoppler search with seticore in the GPU and outputs the hits and associated stamps. In order to test the search mode, we utilized the simulated GUPPI raw files from setigen injected with a 4 Hz/s drifting signal at 3032.25 MHz and 8 Hz/s signal at 3042.25 MHz. The signal injected at 3042.25 had twice the signal level injected at 3032.25 MHz. In the blade commands, -BS was used for beamformed search mode with -SNR 10 and max drift rate = 10 Hz/s. If the search mode is working, we should see the injected signals as hits and their corresponding postage stamp which is a slice of 1 kHz of upchannelized raw data around the hit. If the hit is strong enough, the signals should be observed in all antennas.

The beamformer search mode was able to find all the injected signals as hits and output the corresponding postage stamps. A python viewer based on seticore was used to read the stamps into a numpy array and then conduct the postprocessing. The plots below show the waterfall plots of the injected signal at 3032.5 and 3042.5 MHz in all antennas from the output stamps.

The 4 Hz/s signal detected at 3032.5 MHz:



The 8 Hz/s signal detected at 3042.25 MHz:



10. Observations of Voyager1 to finalize the commissioning

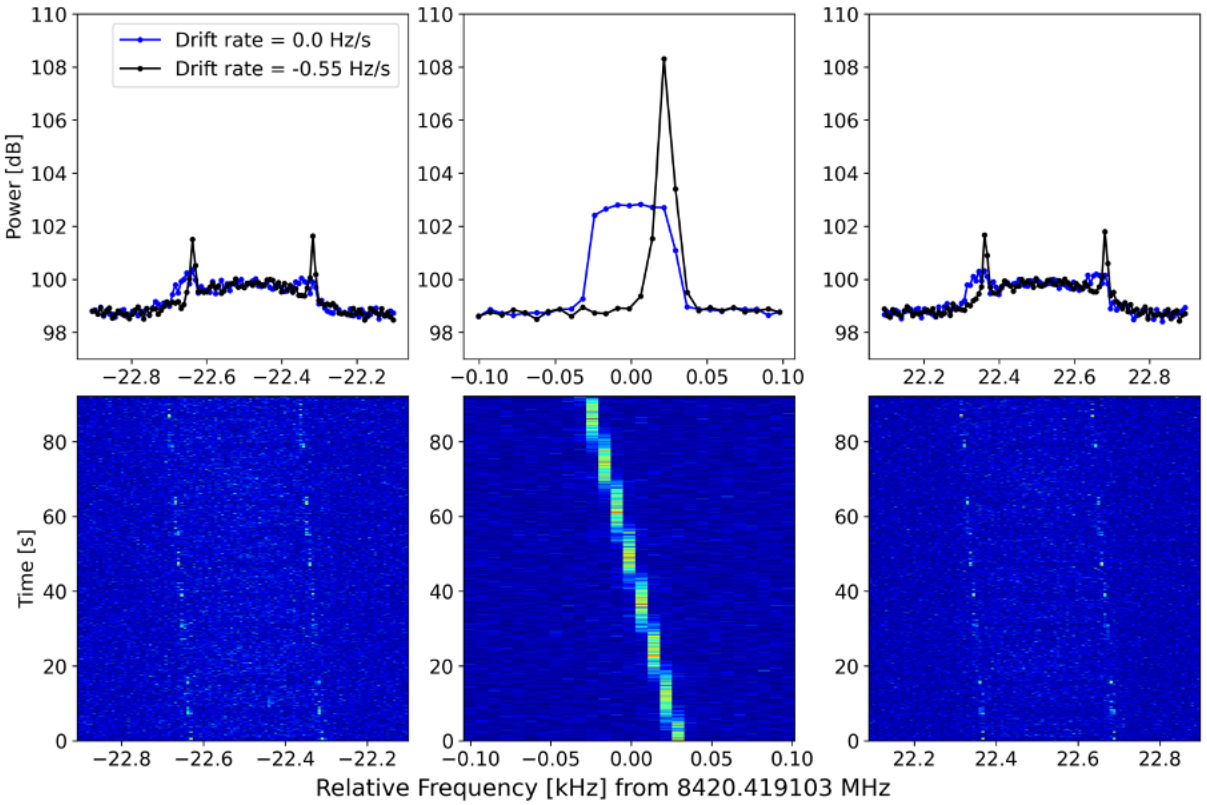
Even though the beamformer was verified using the methanol maser observations and the search mode was tested using the simulated data, the commissioning of the whole pipeline requires the detection of an expected technosignature from large distances. Here the expected technosignature refers to a signal which is highly polarized, narrow bandwidth in frequency and a drifting in time and frequency due to the relative doppler acceleration between receiver and transmitter. For this purpose, we need to observe known transmitters from satellites or space crafts. The best candidate for this task is the detection of the communication signals from the Voyager1 spacecraft launched in the 1970s by NASA. The Voyager1 spacecraft is currently at a distance of 169 AU which makes it the farthest known artificial object. The detection of Voyager 1 is a clear indication that the pipeline is ready to detect similar narrowband drifting technosignatures from other civilizations.

For this purpose, the coordinates of Voyager1 were retrieved from the NASA JPL horizons and a targeted stop and stare observations were conducted towards the spacecraft for a minute. The observations were conducted at X band (8 GHz) and some sideband concerns were required during the observations. The X band sideband is negative for VLA. What it means is that the frequencies around the middle of the bandwidth are flipped. So instead of having a usual increase in the frequencies from low to high, the frequencies will be high to low. This is an after effect of the down conversion. There still needs to be some clarification on the signs of the beamforming phasing coefficient due to this.

There are some uncertainties in the position of the voyager from the JPL Horizons as well, around a couple of arcseconds which are a couple of beams away for the VLA at X band. The beamformed search mode on the incoherent beam detected voyager signals as hits and saved the corresponding stamps. However, the signal was not found in the coherent beams due to uncertainties in the position of the source. The raw data was imaged to find the actual position of the voyager in the observations and beams were formed on those coordinates to detect the signal and understand the spectral structure.

The plot below shows the detection of the Voyager1 signal in the coherent beam. The upchannelization and beamforming were conducted at the same frequency and time resolution of the VLASS search mode (~ 8 Hz frequency and 0.13 s time resolution). The top row shows the integrated spectra without the doppler correction (blue) and with the doppler correction (black). The bottom row shows the corresponding waterfall plots. The left and right column shows the sideband signals and the middle column shows the carrier signal. The sideband signals are detected at +/- 22 kHz from the carrier signal. Both the sideband and carrier signals were found to drift at -0.55 Hz/s. This clearly shows the ability of the pipeline to detect

similar narrowband drifting signals from potential extraterrestrial civilizations. Check out the press release to find more about the voyager detection.



Similar narrowband drifting signals were also detected from the Mars rover satellites used for communication with Earth at X band. The signals from Mars are relatively brighter and positions are better known, and can be used to test the pointing accuracy of the beamformer.

11. RFI flagging to improve calibrations

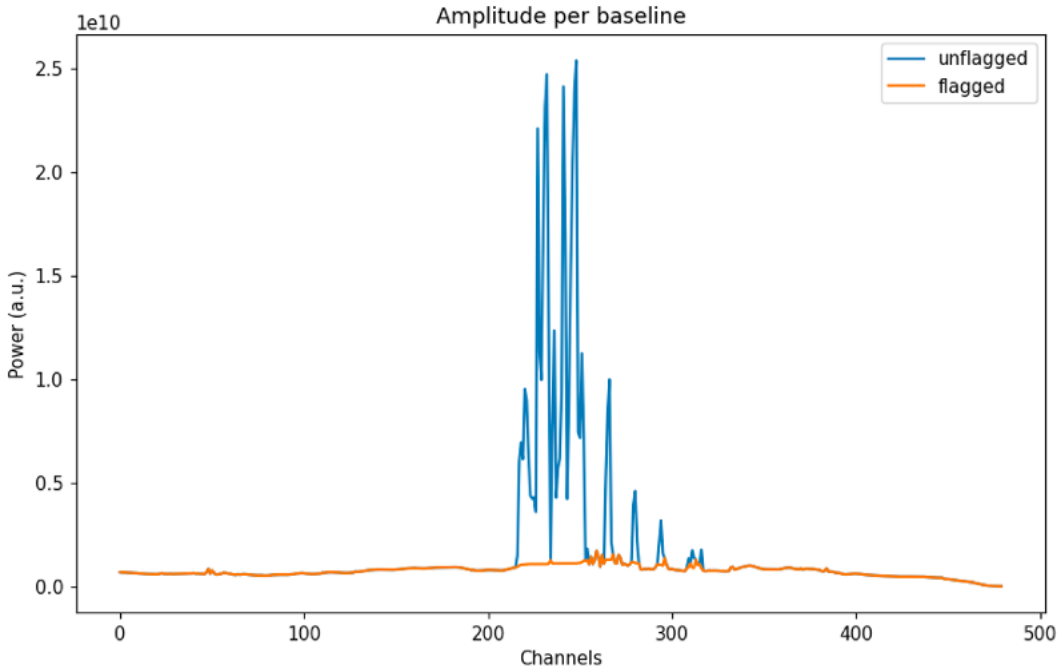
One of the important factors affecting the beamforming efficiency of any radio interferometer is the quality of the calibration. If the calibration solutions are bad, then the beamformer could be phasing the voltages in the wrong direction. The higher bands of the VLA are relatively clean compared to the L and S band. The L band is usually badly affected by the satellite transmissions whereas the S band has clean regions between 2.5 – 3.5 GHz. The calibration is conducted independently for each 32 MHz subband and if they are not completely affected by radio frequency interference (RFI), a decent RFI flagging will give relatively better solutions. Currently there exists a different number of algorithms to conduct the RFI flagging.

Here we are using 2 approaches to conduct a minimal RFI flagging on the correlated data in a real-time manner to improve the gain calibration solutions.

- Sliding median window
- Median based cutoff

Usually, the visibility data is arranged as (baseline, times, frequency, polarization). In both the methods, a time averaged visibility data is used to find the deviant points.

In the sliding median window method, data from each baseline is collected and a sliding median window is used across the frequency response in each polarization to create a smooth bandpass model for each polarization. This technique is really good for finding narrowband RFI signals and may not work for broadband RFI signals. Once the smooth bandpass model is created, this model is subtracted from the data to look for signals deviating by a threshold times the standard deviation. An example of how this works is demonstrated below. The plot below shows the amplitude measured as a function of frequency channels per baseline. The cyan color shows the actual data and the green shows the smooth bandpass model for the same data. This clearly shows how this technique can identify bright RFI points in the spectra per baseline.

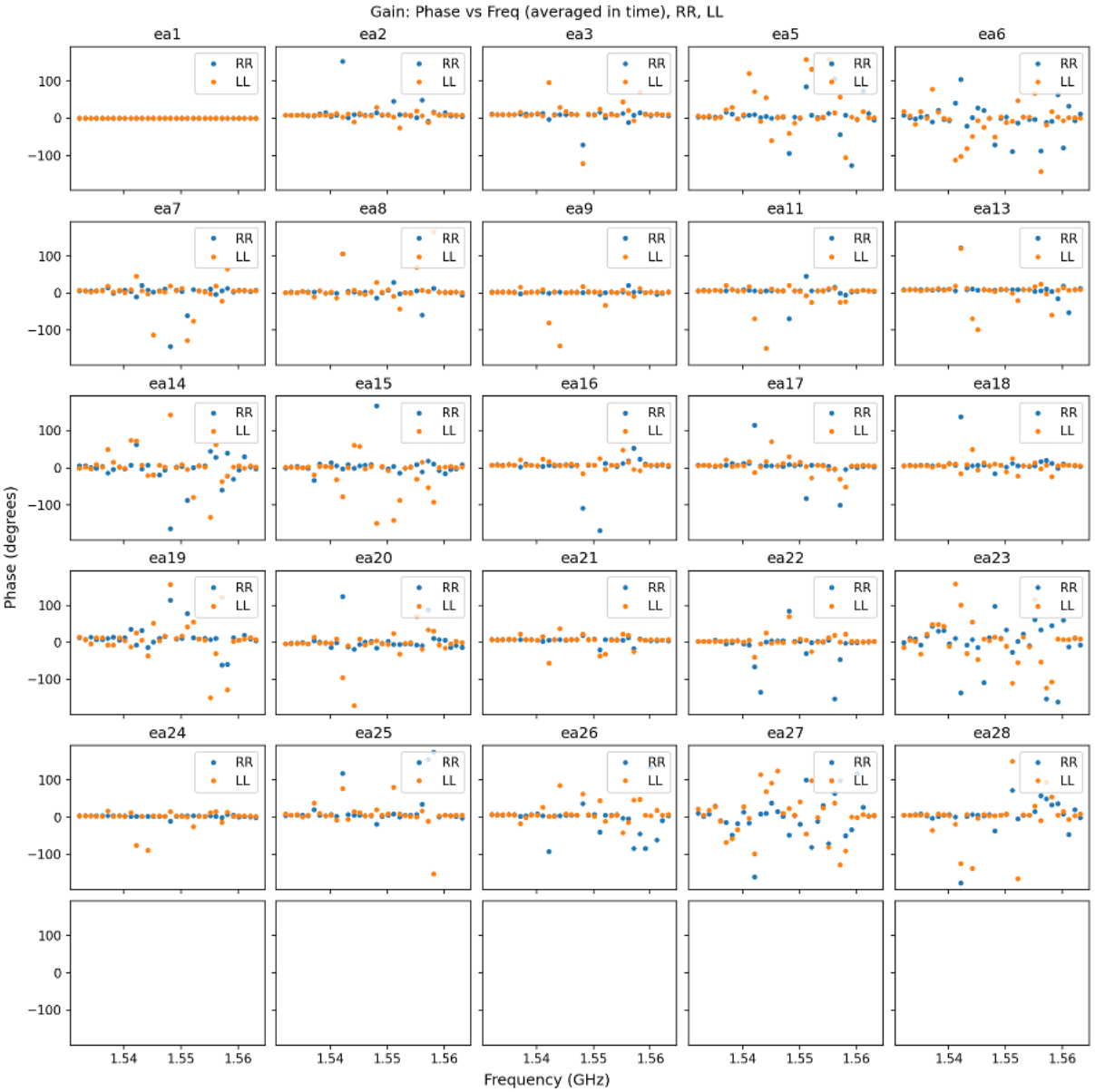


However, this technique seems to be inadequate when the whole band is affected by RFI, causing it to form a poor bandpass model.

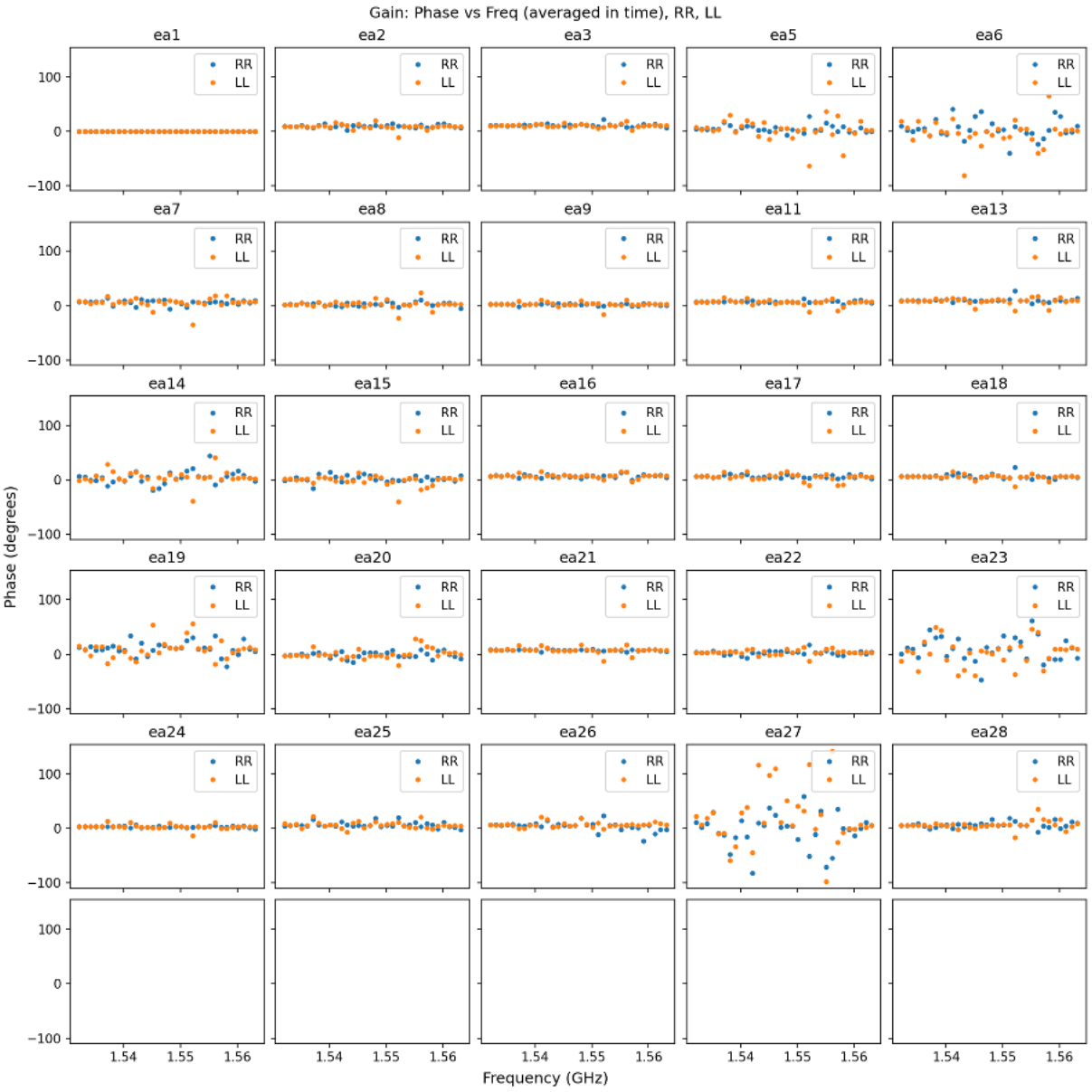
In the median based cutoff technique, for each baseline, a median is calculated across frequencies for each polarization. Data points deviating a certain threshold from the median are replaced with the calculated median values. This method works better if the spectral response is flat. In our case, the flagging is done for a subband of 32 MHz and the spectra from this region can be considered as relatively flat. The region towards the end of the tuning usually has a filter roll off. A higher threshold of 7 sigma is kept to avoid the unwanted flagging in these regions. After testing the datasets with both methods, the median cutoff method seems to be doing a better job when it comes to the RFI flagging of the data. The codes used for flagging the RFI can be found here.

The plot below shows the gains of antenna before and after RFI flagging of the data from a single sub band (32 MHz) with the median cutoff method.

Before Flagging:



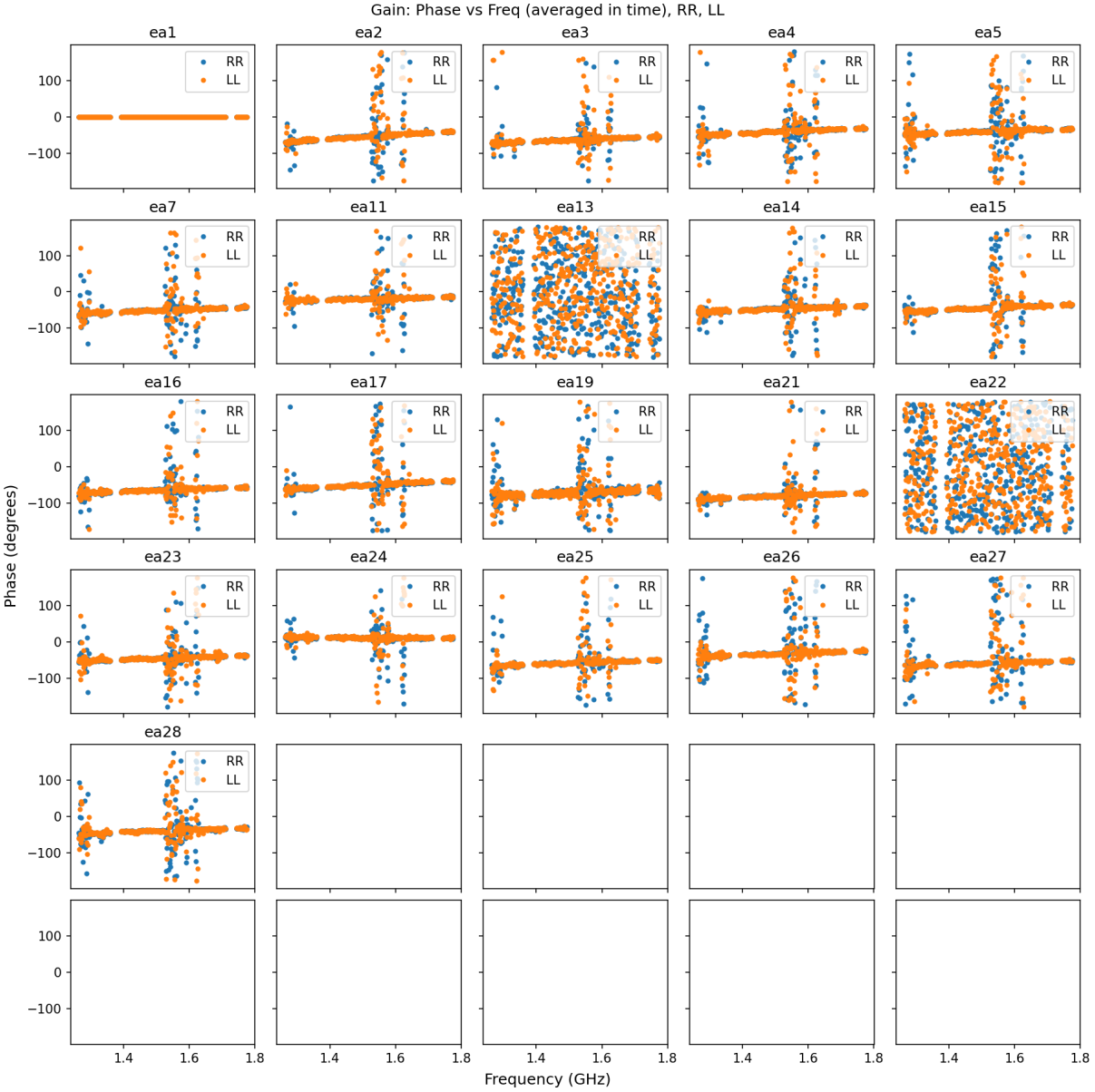
After flagging:



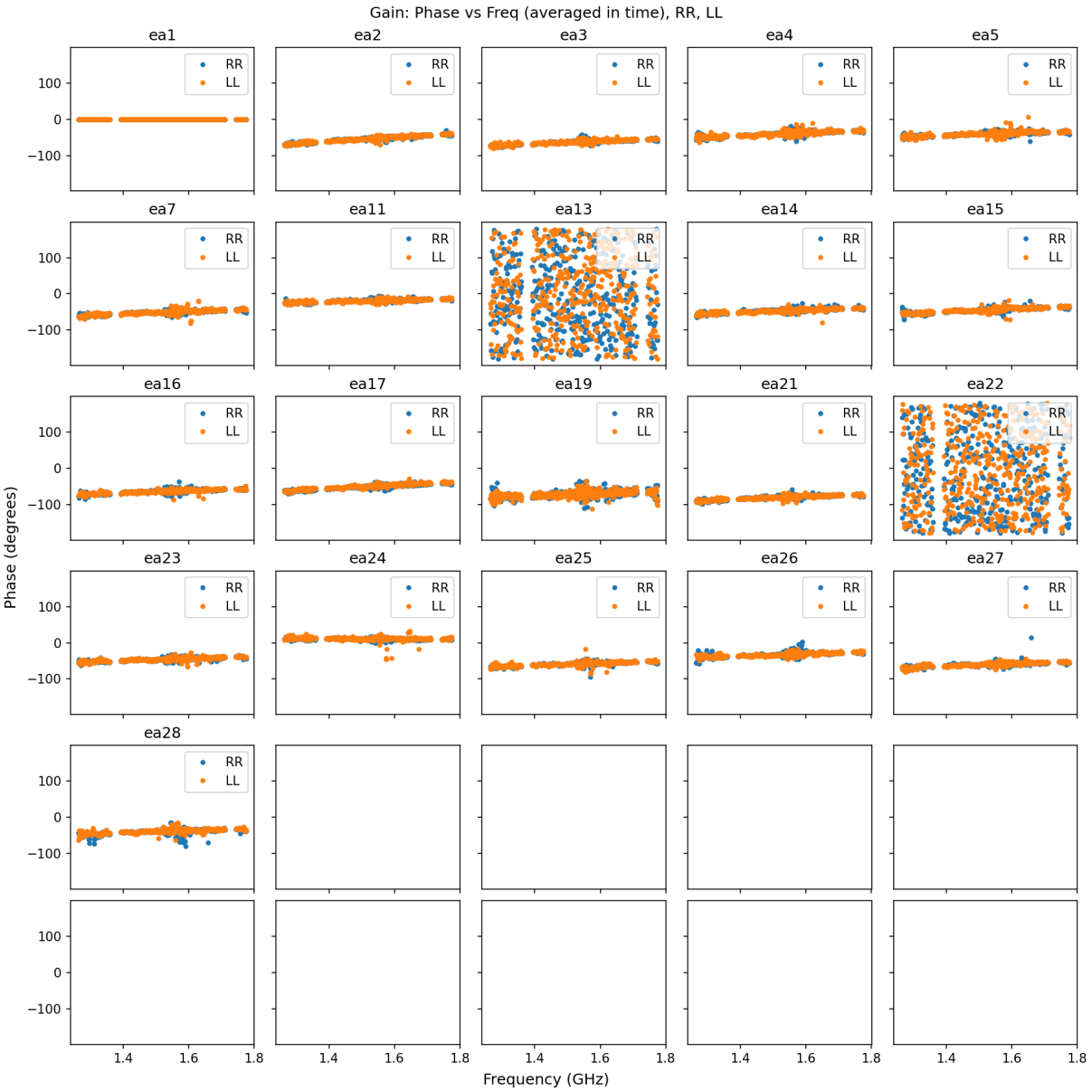
The solution after flagging looks better compared to the unflagged data. The threshold can be changed to conduct an aggressive flagging. However, detection of complicated RFI structures needs looking into other RFI algorithms and testing with the data.

Below is another example of median cufoff based flagging on a calibrator data with wider bandwidth (500 MHz) in L band.

Before flagging:



After flagging:



The gains solutions after flagging looks more or less flatter compared to the scattered solutions before flagging.