Lab #4 – Radio Telescopes and Pulsars

AST326 – February 8, 2019

Ayush Pandhi (ayush.pandhi@mail.utoronto.ca)

Group C: Ayush Pandhi and Hansen Jiang

1 Abstract

The study of pulsars is a large facet of radio astronomy and in this experiment radio data from the Algonquin Radio Observatory (ARO) for Cassiopeia A and B0329+54 are analyzed. The Cass A data is used to estimate parameters for the observatory's 46m dish, such as beamwidth, effective aperture, efficiency, noise temperature and gain. These properties are then used to study the pulsar by generating an average pulse profile through data manipulation (folding and de-dispersion) and using that to study the pulsar over a few periods. However, no clear conclusions can be drawn about the pulse amplitude variation from such a small sub-section of data. Overall, the goal of this experiment was to better understand radio observations regarding pulsars and specifically become familiar with the Algonquin Radio Observatory's instrumentation.

2 Introduction

Radio astronomy specifically deals with observations of celestial objects at low frequencies such as radio galaxies, quasars, pulsars and more. The origins of this young subfield in astronomy is rooted in the communications industry; beginning with Karl Jansky at Bell Telephone Laboratories who became the first person to detect radio waves from an astronomical source in 1932 when he observed radiation from the Milky Way. Over the past century, radio astronomy has gained popularity as it can abuse the radio atmospheric window to allow precise measurements from the Earth and as a result radio antennae across the globe have been conglomerated into large arrays to optimize the quality of observations. However signals at long wavelengths, particularly in the radio, cannot use the photoelectric effect for detection as individual photons do not have sufficient energy to photo-ionize materials. Therefore in radio astronomy, sufficiently strong signals are measured coherently by directly observing the oscillating electric fields within incident light. As such, coherent detectors are useful for analyzing electric fields, power and spectra of these observations at low frequencies.

One specific application of radio astronomy that has emerged is the study of pulsars. A pulsar is a type of rotating neutron star with an extremely strong magnetic field which generates jets of charged particles that emit synchrotron radiation. Importantly, these objects have precise rotational periods, making them great tools for studying the universe: providing insight on the behaviour of fundamental particles in dense environments, they have been used to indirectly detect gravitational waves, probe the interstellar medium through ionization and more.

The following experiment uses data gathered from the Algonquin Radio Observatory 46m dish to calibrate parameters such as beamwidth, effective aperture, efficiency, noise temperature and gain. These are then applied in a study of the pulsar, B0329+54, where an average pulse profile is generated and the pulse variability is analyzed as a convolved timestream. This is done through manipulating the Stokes-I data for the pulsar; folding the data over the period of the pulsar (to get a bright, average pulse), correcting for the dispersion (to account for the dispersive delay over the frequency sub-band) and finally creating a timestream by summing over the frequency for each timestep.

3 Observation and Data

The data sets used for this experiment were taken from the Algonquin Radio Observatory's 46m dish. This telescope is located at (45°57′20″ N, 78°04′23″ W) in terms of latitude and longitude and is at an altitude of 260.4 m. The dish itself has a diameter of 45.8 m, a total collecting area of 1640 m^2 and focal length of 18.3 m. Additionally, the full surface focal ratio is 0.4 and the inner solid surface focal ratio is 0.5 [6]. Both data sets have been pre-processed using a Fourier Transformation to organize the data into 1024 spectral bins over the given frequency range. Then the variance of each spectral bin is computed every 256 samples for both linear polarizations and summed to give a Stokes-I measurement. The systematic errors within the data sets are characterized by the aperture efficiency, noise temperature and gain which are described in detail in Section 5.

The first data set is a binary file of archival data of Cassiopeia A from August 8, 2017 (13h 43m 28.5103s - 14h 01m 23.302s), taken in the 400-800 MHz band (organized in 1024 frequency bins, each covering roughly 390 kHz) over 49 time slices (each 1027 s) [4]. The scanning area in the sky is 337.75° – 339.90° (azimuth) and 16.49° – 24.40° (elevation). A rough sketch of the telescope orientation is provided in the Appendix (Figure 13). The data itself provides the Stokes-I samples for each frequency bin at each time slice. The second data set is also a binary file of archival data of the pulsar, B0349+54 from August 3, 2016, also taken in the 400-800 MHz band (organized in 1024 frequency bins) over approximately 90 seconds [4]. This data set also provides Stokes-I measurements for each frequency bin over the scanning time.

4 Data Reduction and Methods

The first stage of this experiment involves computing various properties of the ARO 46m telescope. These parameters stem from applying a Gaussian curve fit to the power as a function of time for all sub-bands within the data set. The model function for a Gaussian function is given by:

$$f(x) = ae^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{1}$$

Where, in this case, x is the power from the Stokes-I measurements, a is the amplitude, μ is the mean value of x and σ is the standard deviation of x. The mean and standard deviation themselves is described as:

$$\mu_x = \frac{\sum_i x_i}{N} \quad and \quad \sigma_x = \sqrt{\frac{\sum_i (x_i - \mu_x)^2}{N}} \tag{2}$$

Applying this fit provides estimates for the amplitude and beamwidth as a function of frequency. Furthermore, the experimental full-width half maximum in the Gaussian case is then computed directly from the beamwidth as [4]:

$$FWHM_e = 2.355(beamwidth) \tag{3}$$

The theoretical full-width half maximum as a function of frequency (or wavelength) stems from the simple diffraction limit for point spread functions [2]:

$$\Delta\theta \approx \frac{1.22\lambda}{d} \quad \Longrightarrow \quad FWHM \approx \frac{1.028\lambda}{d} \tag{4}$$

Where λ is the wavelength and d is the diameter of the observing dish. The full-width half maximum helps parameterize the Airy pattern (roughly Gaussian) profile of the beams. This is then further developed to provide the effective aperture (and immediately aperture efficiency as well), which is given as [4]:

$$\mathrm{A_{eff}} = \frac{\lambda^2}{SA} \quad and \quad A_{phys} = \pi r^2 \quad \longrightarrow \quad Efficiency = A_{eff}/A_{phys} \tag{5}$$

Here, SA is the solid-angle for the system and A_{phys} is the physical aperture area which depends on radius r. The flux of Cassiopeia A was referenced from *Baars et all, 1977* [1] and was halved to account for the 50% dimming effect over 40 years. Using a linear fit for fluxes provided in the 400-800 MHz band, a relationship for flux as a function of frequency was developed. Using this, the temperature of the system was estimated as [2]:

$$T = \frac{P}{k\Delta v} = \frac{FA_{eff}}{k\Delta v} \tag{6}$$

Where P is the power, F is the flux, v is frequency and k is the Boltzmann constant. This was then used to calibrate the noise temperature and gain of the system, again with the use of the curve fit module.

When working with the B0329+54 data set, folding the data set provides a more clear, average, profile of the pulsar's Stokes-I measurements over the frequency range. This is done by cutting the data set at a constant interval correlating to the period of the pulsar and folding each cut onto one another over the time axis. The flux is then computed for the folded data set using the previously computed gain.

The folded data set shows a clear dispersion with higher frequencies arriving before the lower frequencies. To correct for this, the data is de-dispersed using the dispersion measure (DM) for this specific pulsar found on the ATNF Pulsar Database [3] and the de-dispersion follows [4]:

$$DD = 4.15x10^{-3}\frac{DM}{v^2} \tag{7}$$

Furthermore, to analyze the average pulse profile of the pulsar, the de-dispersed data set is summed over the frequency axis for each timestep. This average profile is then used to convolve the original timestream to help pick out the pulses more clearly.

Throughout the experiment, various data reduction methods were applied to make the data set manageable or cut out noise. The Stokes-I data for Cassiopeia A was initially reduced by cutting out RFI that had abnormally high off-peak intensity using a simple intensity threshold condition when applying a Gaussian fit. The B0329+54 data set was very large and a small (10,000 point) portion was cut from the full data set and worked with instead. Throughout the analysis and modeling, waterfall plots for this pulsar are also divided out by their median to make the pulsar's signal pop out compared to the background noise. Additionally, to reduce noise in the timestream when summing over frequency, the data is cut at roughly 730 MHz as to not include the LTE sub-band.

The largest source of error throughout the analysis and modeling section arises from statistical uncertainty when fitting the data to some specific model function. This uncertainty dominates the fluctuations described by the radiometer equation and other minor conditions. These statistical uncertainties are drawn from the estimated covariance matrix by the curve fit module and then are propagated to further parameters which derive from the original fitted parameter.

5 Data Analysis and Modeling

The detailed python code for generating all figures and data analysis in this section can be found in the Appendix. To start, the elevation and azimuthal offset was computed and plotted against the total time of the scan (Figure 1 in Appendix). From this it is clear that elevation offset dominates the total offset of the dish and as such further analysis was done with only the elevation offset in mind. The power was then plotted as a function of offset for each frequency sub-band and fitted to a Gaussian model function (Equation 1). As mentioned earlier, some RFI was removed by doing a threshold cut for off-peak power (Figure 2).

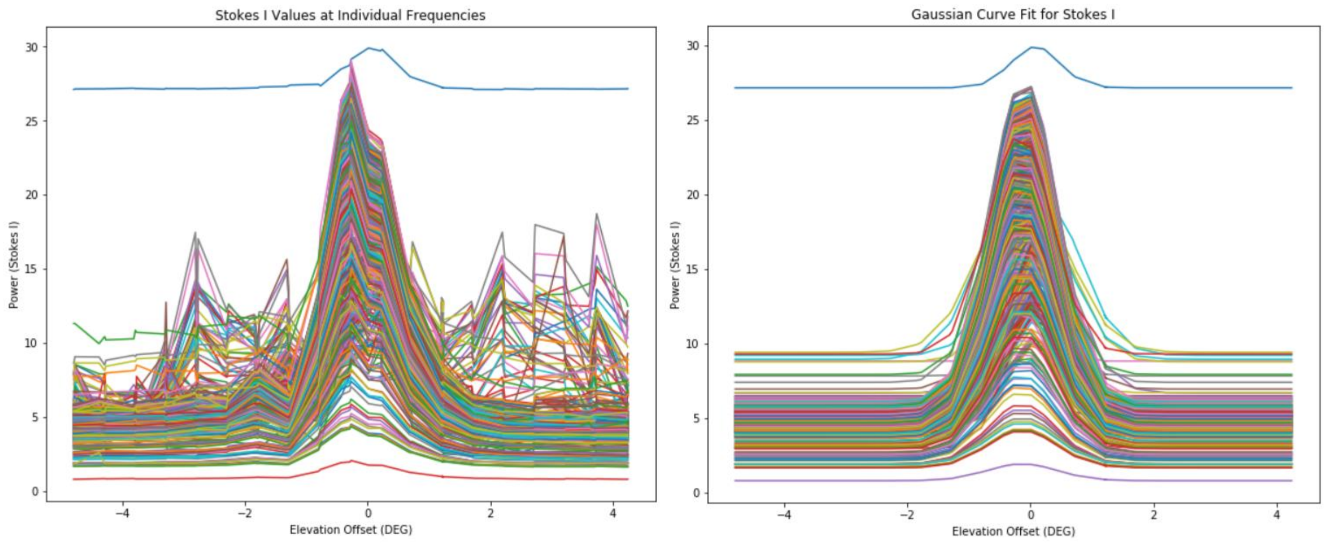


Figure 2: *a) (Left) Power plotted over elevation offset for each frequency sub-band, b) (Right) Power curve fitted to a Gaussian and with RFI removed to clearly see the peak power from B0329+54.*

From this curve fit, the amplitude and FWHM beamwidth were computed based on Equation 3 and are plotted against the theoretical beamwidth given in Equation 4 (Figure 3). As mentioned earlier, the uncertainty in these figures arises from the estimated covariance parameters of the curve fit.

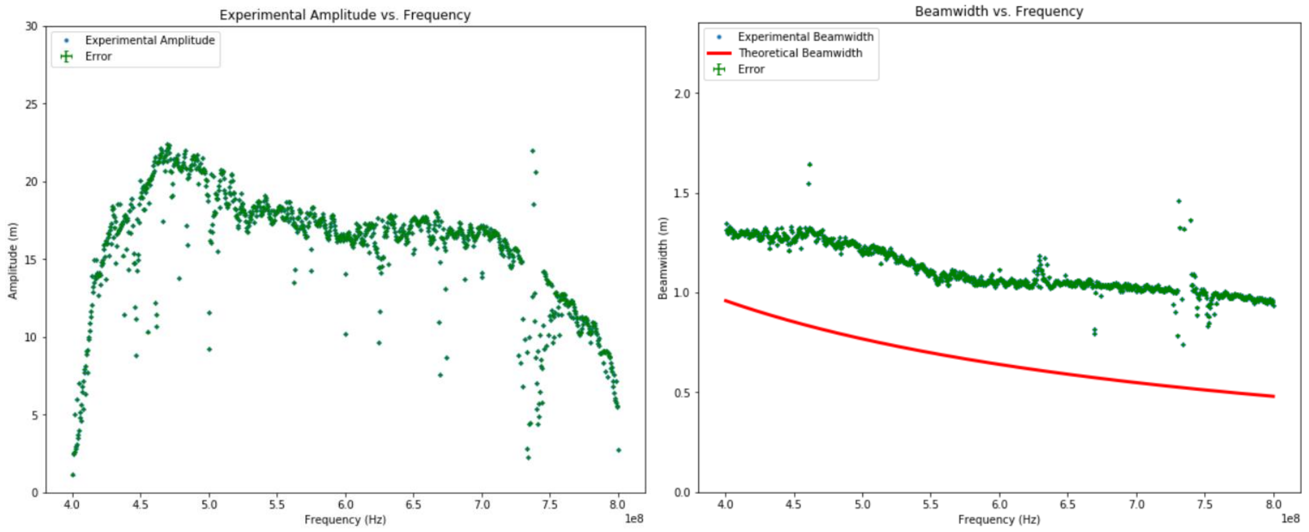


Figure 3: *a) (Left) Amplitude plotted as a function of frequency with corresponding curve fit errors, b) (Right) FWHM beamwidth (experimental and theoretical) plotted as a function of frequency also with curve fit errors.*

Furthermore, the aperture efficiency was computed in accordance with Equation 5 and the errors from the FWHM were propagated to plot the efficiency as a function of frequency as well (Figure 4).

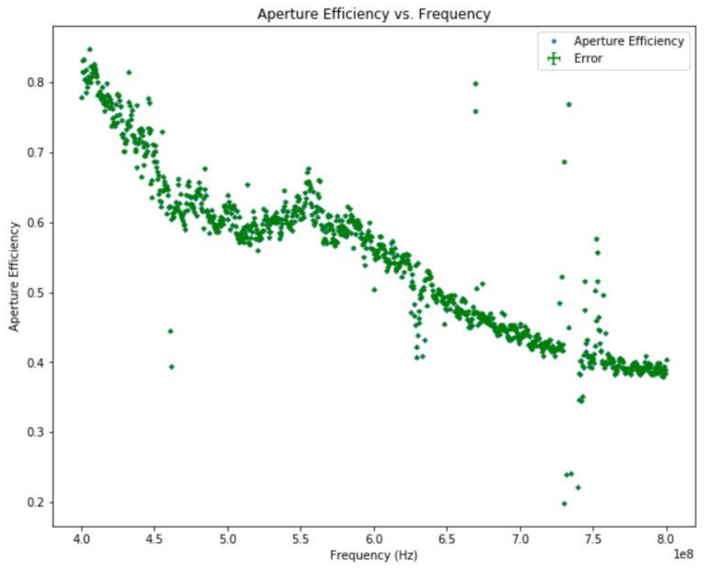


Figure 4: *Aperture Efficiency plotted as a function of frequency with propagated curve fit errors.*

Calibrating a flux relationship with frequency using *Baars et al, 1977* [1] as reference, the temperature can be obtained from Equation 6 and is plotted against frequency (Figure 5). As expected the shape of the distribution is similar to that of the aperture efficiency and the temperature between 400-800 MHz is roughly between 900 – 3000K.

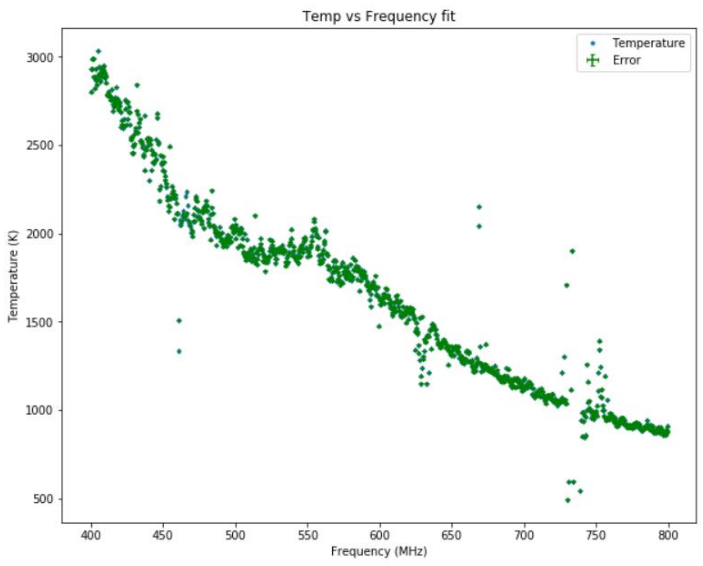


Figure 5: *Temperature plotted as a function of frequency with propagated curve fit errors.*

Assuming the sky around Cassiopeia A is uniformly 10K, the noise and gain can be computed by fitting a linear relationship at each frequency for the temperature and Stokes-I measurements (Figure 6).

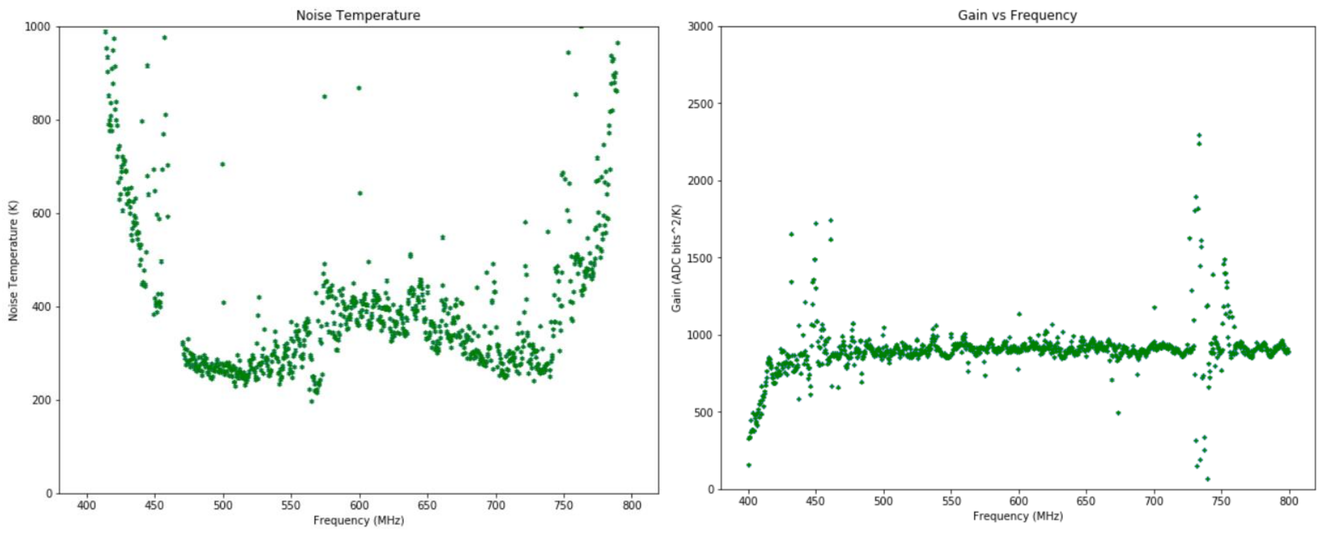


Figure 6: *a) (Left) Noise Temperature plotted vs Frequency with curve fit covariance uncertainties, b) (Right) Forward Gain plotted against Frequency also with curve fit uncertainties.*

Analyzing the B0329+54 pulsar's data, first the raw waterfall plot of the Stokes-I data is plotted for a 10,000 sub-section of the full data set. To make the pulsar's signal more visible, the median is divided out from the data and replotted (Figure 7).

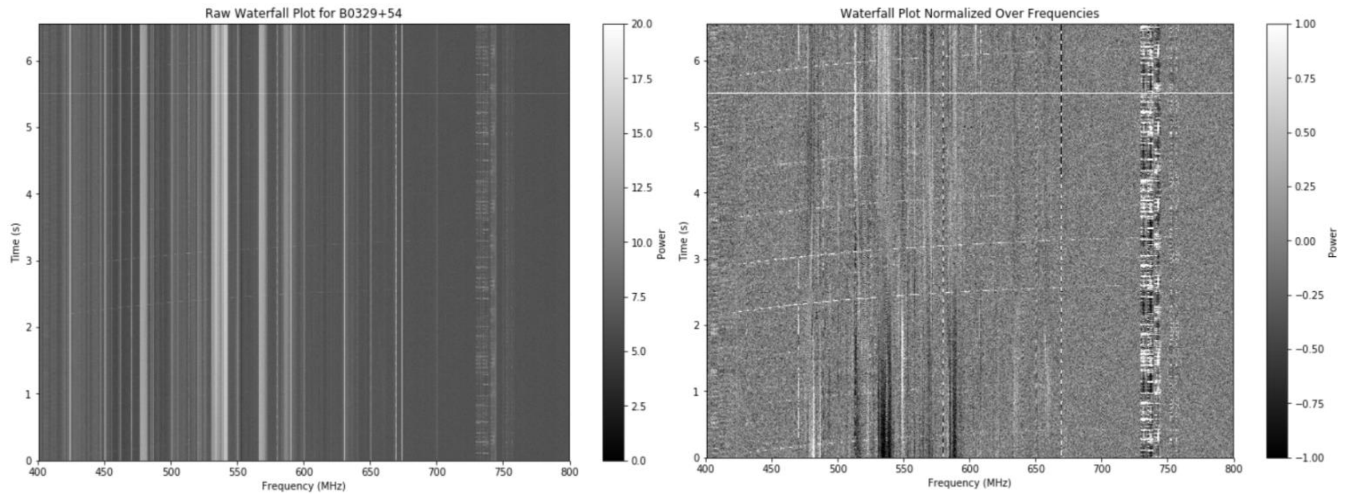


Figure 7: *a) (Left) Raw waterfall plot of B0329+54 data, b) (Right) Same plot but with the median divided out.*

To get a single distinctive average pulse, the dataset is folded and a waterfall plot for the power in terms of frequency and pulse phase (Figure 8). Additionally, the flux of B0329+54 across the frequency band is computed by using the gain that was estimated previously. The pulsar's typical flux in this band is approximated 1.6 Jy.

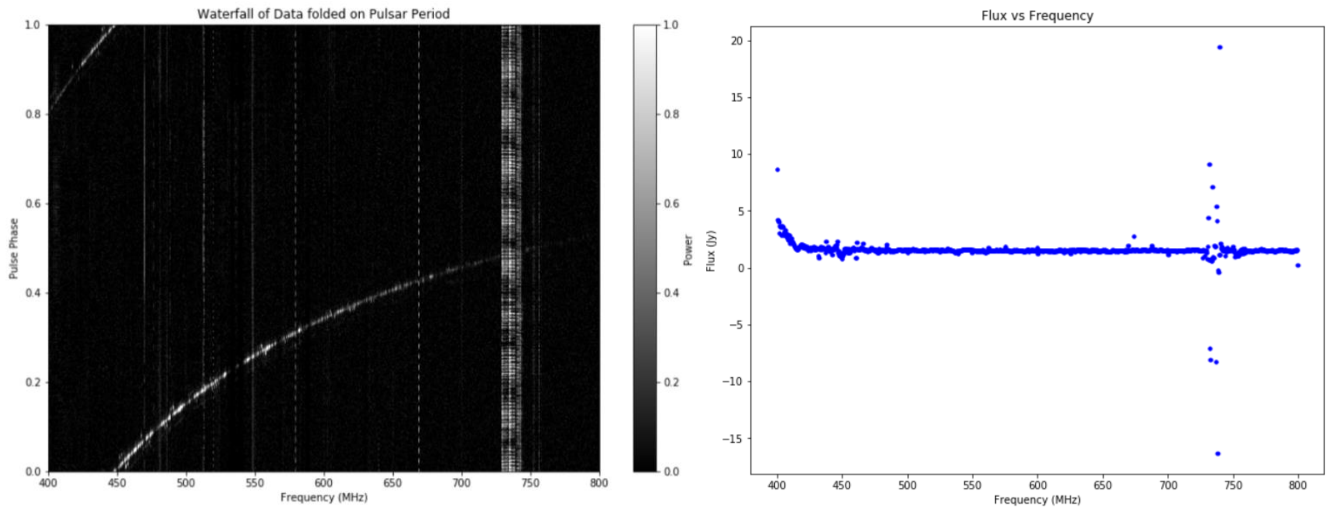


Figure 8: *a) (Left) Folded waterfall plot of B0329+54 data with respect to frequency and pulse phase, b) (Right) Flux of B0329+54 plotted against frequency.*

To correct for the dispersion effect that causes higher frequencies to arrive gradually faster than lower frequency, Equation 7 is applied to the folded dataset. The resulting waterfall plot is a representation of what the pulse would look like when it was emitted (Figure 9).

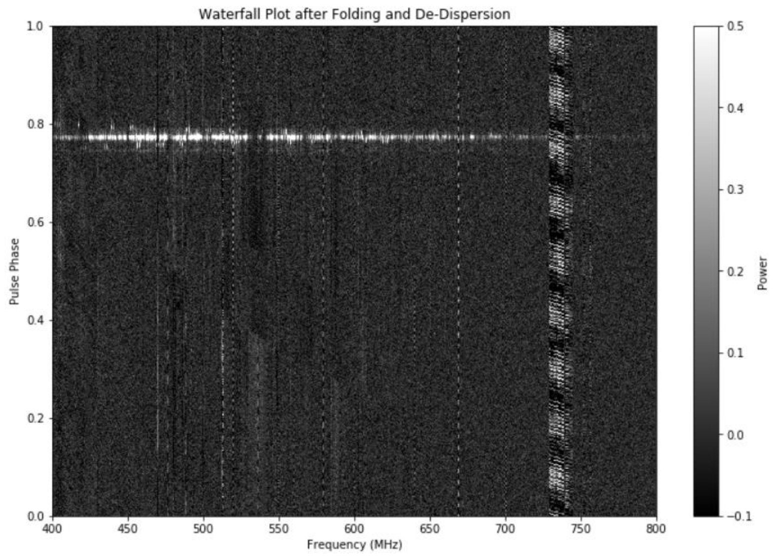


Figure 9: *Waterfall plot of the folded data set after de-dispersion with respect to frequency and pulse phase.*

By summing over the frequency axis of the folded and de-dispersed data, an average pulse profile is generated. It is paramount that this step is after de-dispersion because otherwise it would result in a noisy timestream with no clear pulse (Figure 10). The plot has axis of pulse phase and flux relative to mean, where the mean is roughly 1.6 Jy, as computed earlier.

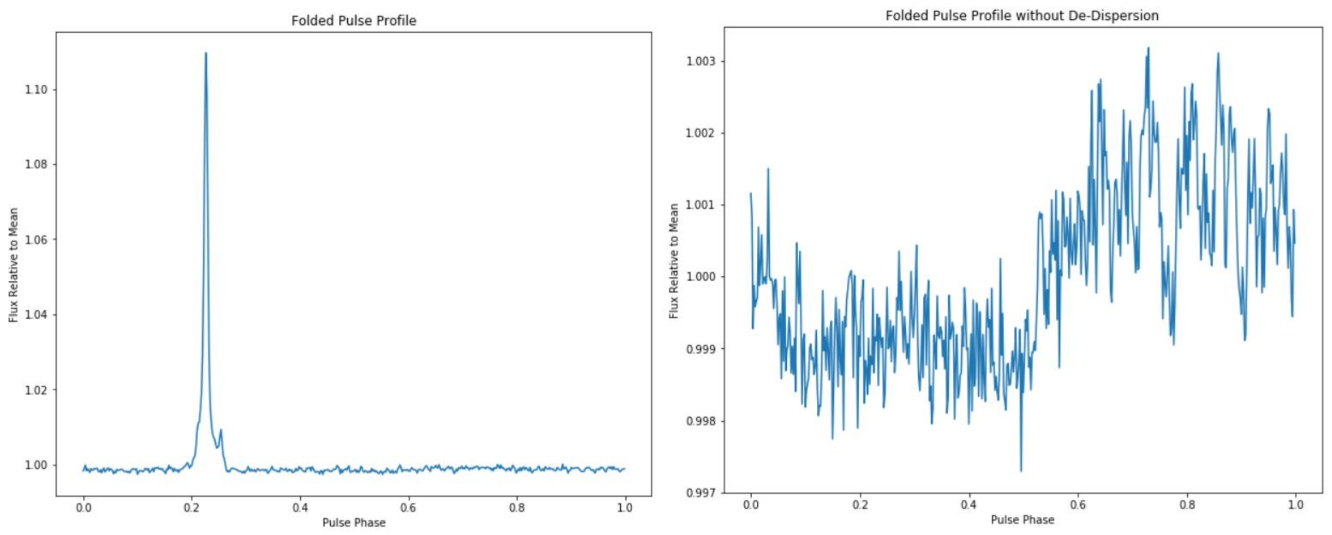
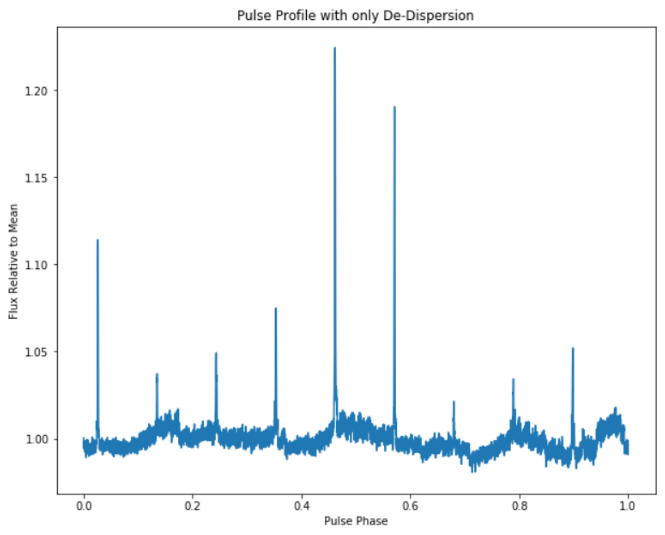


Figure 10: *a) (Left) An average pulse profile of B0329+54 taken by summing over frequency of a folded and de-dispersed dataset, b) (Right) Done similarly but applied on a dataset that has not been de-dispersed.*

To analyze pulse variability, the un-folded data set is taken, de-dispersed and summed over frequency so that multiple peaks can be seen across the timestream. A histogram depicting the number of measurements at certain flux levels is also created (Figure 11).



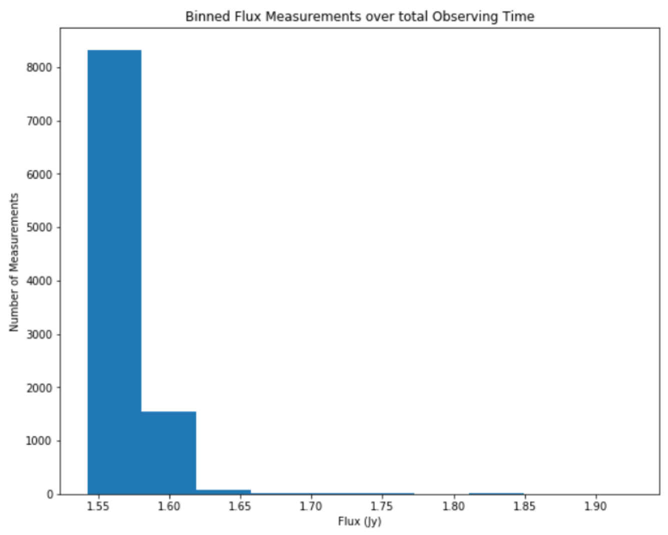


Figure 11: *a) (Left) A timestream generated from the unfolded dataset to show individual pulses, b) (Right) Histogram of number of measurements for specific flux (Jy).*

Finally, the average pulse profile (Figure 10 a)) is used to convolve the un-folded time stream to help pick out pulses better. The resulting convolved time stream is also given in terms of flux relative to the mean and pulse phase (Figure 12).

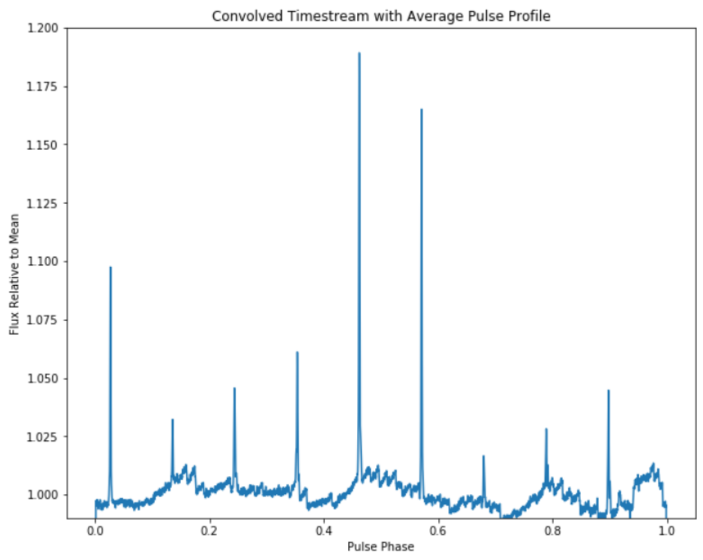


Figure 12: *Similar to Figure 11 a) except convolved with the average pulse profile of the pulsar.*

6 Discussion

Firstly, discussing the Cassiopeia A data and its use in calibrating the ARO parameters. The object is a point source and as such fits rather well into a Gaussian model as seen in Figure 2. The central peak clearly depicts the increase in power when scanning across the object while a lot of the RFI that is high-power away from the central peak is disregarded as noise and not a physical component of the system being observed.

The experimental beamwidth in Figure 3 is visibly larger than the theoretical, diffraction-limited, beamwidth even with uncertainties being taken into consideration. This matches the expected outcome as the telescope is not ideal and does have diffraction affects that cause a wider beam which can be parameterized as a point spread function.

Examining the aperture efficiency, it is expected that the telescope is not 100% efficient as it is not ideal. However in Figure 4, it can be seen that the aperture efficiency is greater at lower frequencies and falls off at higher frequencies. The observed range in efficiency is roughly 80% to 40% over the 400-800 MHz band. Additionally, as expected from Equation 6, the temperature (Figure 5) follows a similar pattern to the aperture efficiency with higher temperatures at lower frequencies with an overall range of approximately 3000K to 900K.

From Figure 6, it is observed that the noise temperature remains fairly stable at 200-400K between approximately 450-750 MHz and becomes extremely high at either end of the band. Somewhat similarly, the gain is very stable for the majority of the frequency band with a fall off at lower frequencies near 400 MHz. Again, the RFI (especially near 730-750 MHz) appears as an outlier on the data set but is ignored as it does not pertain to a physical feature of the object in question.

Studying B0329+54, in Figure 7 it is difficult to pick out the pulses unless the median is divided out from the data and even then it is faint. In Figure 8, the dataset is folded to stack the pulses onto one another and show a much brighter summed signal. Again, the LTE RFI is disregarded for this analysis. The flux remains fairly constant throughout the band and the typical flux is roughly 1.6 Jy. The small increase in flux at lower wavelengths corresponds to the lower gain in that region.

The clear dispersion pattern seen in Figure 8 is de-dispersed in Figure 9 and the pulse, as it would appear at the source, can be seen. The importance of de-dispersion can be seen in Figure 10 as the de-dispersed dataset produces a clear average pulse profile while the one without de-dispersion is incoherent and appears as noise across the band. This is because averaging (horizontally) over each frequency serves no purpose if the pulse is curved due dispersion.

In Figure 11, it can be observed that the pulse amplitudes vary over time. From this limited data, a small symmetry can be observed for pulses 2-4 and 5-7 where the amplitude increases slightly after each pulse. However, this analysis is done for a very small sub-section (10,000 points) of the full data and it cannot be easily concluded whether there is a clear overall pattern in the pulse amplitude variation.

Figure 12 is similar to Figure 11 however the convolved set makes it slightly easier to pick out the peaks from the dataset. The pulse seems to always appear in the same place within its period, which was obtained from the ATNF Pulsar Catalogue to be roughly 0.71 seconds.

7 References

[1] Baars et al, Max Planck Institute for Radio-astronomy, Bonn, Germany. "The absolute spectrum of CAS A - an accurate flux density scale and a set of secondary calibrators", Astronomy and Astrophysics, vol. 61, October 1977.

[2] Carroll B. W., Ostlie D. A., Weber State University, Ogden, Utah. "An Introduction to Modern Astrophysics", 1996.

[3] CSIRO Australia Telescope National Facility, Epping, Australia. "ATNF Pulsar Catalogue", May 2018. Retrieved from https://www.atnf.csiro.au/research/pulsar/psrcat/.

[4] Department of Astronomy and Astrophysics, University of Toronto, Toronto, ON. "Astronomy 326 Lab 4: Radio Telescopes and Pulsars", February 2019. Retrieved from http://www.astro.utoronto.ca/~astrolab/.

[5] Ransom S., National Radio Astronomy Observatory, Charlottesville, Virginia. "Radio Telescopes and Radiometers", October 2018. Retrieved from https://www.cv.nrao.edu/~sransom/web/Ch3.html.

[6] Thoth Technology, Pembroke, Ontario. "ARO Technical Specifications", January 2019. Retrieved from http://thothx.com/services/aro-technical-specifications/.

8 Appendix

8.1 Azimuthal and Elevation Offsets for ARO's Cassiopeia A data

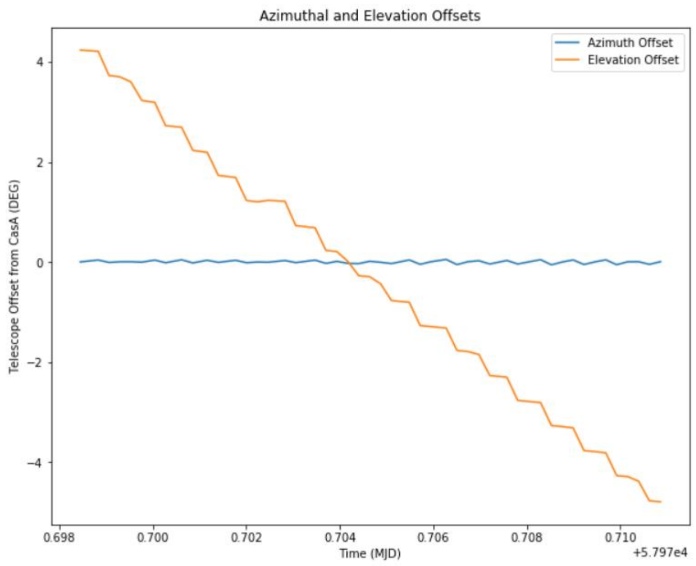


Figure 1: *Azimuthal and Elevation Offsets of the ARO 46m dish plotted as a function of scan time.*

8.2 Rough Sketch of ARO's 46m Dish Orientation while observing Cassiopeia A

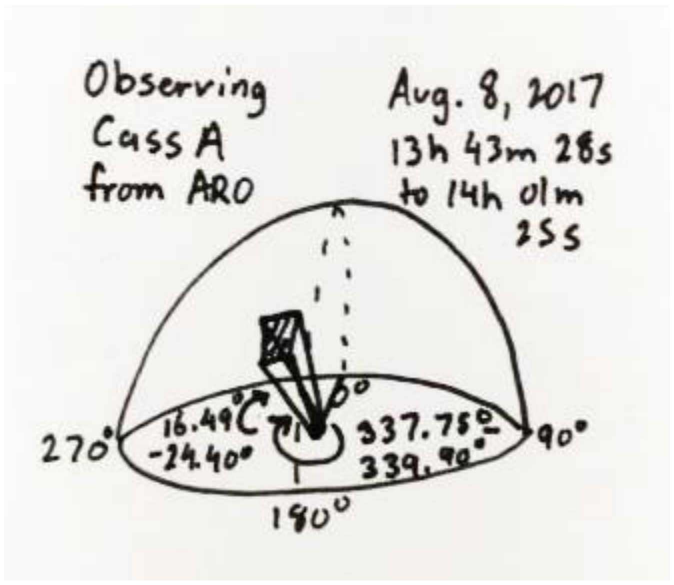


Figure 13: *ARO 46m dish scanning a portion of the sky over Cassiopeia A. Illustrated by the author.*

8.3 Python Code for Calibrating ARO with Cassieopia A Data (Figures 1-6)

```
#AST326 Lab 4 – Radio Telescopes and Pulsars
#Ayush Pandhi (1003227457)
#February 8, 2019

#Importing required modules
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as md
import astropy as ast
from astropy import coordinates
from astropy.time import Time
from scipy.optimize import curve_fit
import scipy.stats as st
import datetime
import struct

#Gaussian model function
def gauss_function(x, amp, mu, sigma, C):
    return amp*np.exp(-(x-mu)**2/(2*sigma**2)) + C

#Linear model function
def y(x, m, b):
    return ((m*x)+b)

#Defining a function to read data (for 4.2)
def read_data(rawdata):
    data_header = rawdata.read(ARO_header)
    idx, azimuth, elevation = struct.unpack('Iff', data_header)
    data = np.fromstring(rawdata.read(ARO_length - ARO_header), dtype=np.float32)
    return idx, azimuth, elevation, data

#Loading Cass A data from the .dat file
casA_full = np.fromfile('AST326_ARO_CasA_calibration.dat', dtype=float)

#Re-organizing the data into required parameters
nf = casA_full[0]                        #number of frequency channels
nt = casA_full[1]                        #number of time slices
ns = casA_full[2]                        #size of each time slice
lf = casA_full[3:3+1024]                 #list of frequencies
wl = (3*(10**8))/lf                      #list of wavelengths
casA_samples = casA_full[1027:].reshape(49,1027)
timestamp = np.empty(49,)
elevation = np.empty(49,)
azimuth = np.empty(49,)
stokes = np.empty([49,1024])
for i in range(49):
    timestamp[i] = casA_samples[i,0]        #timestamp on observations
    elevation[i] = casA_samples[i,1]        #altitude of the telescope
    azimuth[i] = casA_samples[i,2]          #azimuth of the telescope
    for j in range(1024):
        stokes[i,j] = casA_samples[i,3+j]       #stokes samples for each frequency

#Basic properties of the ARO
height = 260.4           #meters
diameter = 45.8          #meters
focal_length = 18.3      #meters
focal_ratio_fs = 0.4     #full surface
focal_ratio_iss = 0.5    #inner solid surface
latitude = 45.95550000     #45 57 19.8
longitude = 281.92694444   #281 55 37.0

#Using astropy's time and coordinates to get orientation of telescope
t = Time(timestamp, format='mjd')
lat = coordinates.Angle('45.9557d')
```

```
long = coordinates.Angle('-78.0720d')
alt = 260.4
loc = coordinates.EarthLocation(lat=lat, lon=long, height=alt)
ra = coordinates.Angle(u'23h 23m 26s')
dec = coordinates.Angle(u'58d 48')
c = coordinates.SkyCoord(ra=ra, dec=dec)
aa = c.transform_to(ast.coordinates.AltAz(location=loc, obstime=t))
el = aa.alt.deg
az = aa.az.deg

#Computing offsets for elevation and azimuth
az_offset = az - azimuth
el_offset = el - elevation

#Plotting offsets over time
plt.figure(figsize=(10,8))
plt.plot(t.mjd, az_offset, label='Azimuth Offset')
plt.plot(t.mjd, el_offset, label='Elevation Offset')
plt.title('Azimuthal and Elevation Offsets')
plt.xlabel('Time (MJD)')
plt.ylabel('Telescope Offset from CasA (DEG)')
plt.legend()
plt.show()

#Plotting the power in each sub-band
plt.figure(figsize=(10,8))
plt.plot(el_offset, stokes)
plt.title('Stokes I Values at Individual Frequencies')
plt.xlabel('Elevation Offset (DEG)')
plt.ylabel('Power (Stokes I)')
plt.show()

#Fitting with curve fit
amp = np.zeros(1024)
bw = np.zeros(1024)
pt = np.zeros(1024)
base = np.zeros(1024)
amp2 = np.zeros(1024)
bw2 = np.zeros(1024)
pt2 = np.zeros(1024)
base2 = np.zeros(1024)
fwhm = np.zeros(1024)
```

```
var = np.zeros(1024)
reduc = np.zeros(1024)

#Setting up plot parameters
plt.figure(figsize=(10,8))
plt.title('Gaussian Curve Fit for Stokes I')
plt.xlabel('Elevation Offset (DEG)')
plt.ylabel('Power (Stokes I)')

#Loop to fit Stokes I values for each freq to a gaussian
for i in range(1024):
    coeff, var_matrix = curve_fit(gauss_function, el_offset, stokes[:,i], p0=[20, 0, 1, 5])
    amp[i] = coeff[0]
    pt[i] = coeff[1]
    bw[i] = coeff[2]
    base[i] = coeff[3]
    gauss = gauss_function(el_offset, coeff[0], coeff[1], coeff[2], coeff[3])
    gauss2 = gauss - np.min(gauss)

    #Removing some RFI under a selected threshold
    if gauss2[36] <= 0.1:
        plt.plot(el_offset, gauss, '-')
        amp2[i] = amp[i]
        pt2[i] = pt[i]
        bw2[i] = bw[i]
        base2[i] = base[i]

#Getting experimental/theoretical FWHM and plotting
fwhm_e = 2.355*bw2
fwhm_t = 1.028*wl*(1/46)*(180/np.pi)
plt.show()

#Defining errors based on covariance matrix
amp_error = 7.71125738e-04
bw_error = 3.57170002e-04
fwhm_error = 2.355*bw_error

#Plotting Experimental Amplitude vs. frequency
plt.figure(figsize=(10,8))
plt.plot(lf, amp, '.', label='Experimental Amplitude')
plt.errorbar(lf, amp2, xerr=0, yerr=amp_error, linestyle='none', color='g', label='Error', capsize=2)
plt.title('Experimental Amplitude vs. Frequency')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (m)')
plt.ylim(0,30)
plt.legend()
plt.show()

#Plotting Experimental Bandwidth vs. frequency
plt.figure(figsize=(10,8))
plt.plot(lf, fwhm_e, '.', label='Experimental Beamwidth')
plt.plot(lf, fwhm_t, '-', linewidth=3, color='r', label='Theoretical Beamwidth')
plt.errorbar(lf, fwhm_e, xerr=0, yerr=fwhm_error, linestyle='none', color='g', label='Error', capsize=2)
plt.title('Beamwidth vs. Frequency')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Beamwidth (m)')
plt.ylim(0,2.355)
plt.legend()
plt.show()

#Aperture efficiency based on solid angle
solidangle = (np.pi/4)*((fwhm_e*(np.pi/180))**2)
A_eff = wl**2/solidangle
A_phys = np.pi*(23**2)
efficiency = A_eff/A_phys

#Defining error in solid angle and aperture efficieny
solidangle_error = (2*((fwhm_error*(np.pi/180))/(fwhm_e*(np.pi/180)))*solidangle)
A_eff_error = A_eff*(solidangle_error/solidangle)
efficiency_error = A_eff_error/A_phys

#Plotting Aperture Efficiency vs. Frequency
plt.figure(figsize=(10,8))
plt.plot(lf, efficiency, '.', label='Aperture Efficiency')
plt.errorbar(lf, efficiency, xerr=0, yerr=efficiency_error, linestyle='none', color='g', label='Error', capsize=2)
plt.title('Aperture Efficiency vs. Frequency')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Aperture Efficiency')
plt.legend()
plt.show()

#Calibrating reference flux and computing power/temp
kb = 1.380*(10**(-23))            #Boltzmann constant
flux_550 = (5170/2)*(10**(-26))    #SI units
```

```
power = flux_550*A_eff[640]
temp = power/kb

#Printing temperature of CasA at 550 MHz
print('CasA power and temperature on the telescope at 550 MHz:', power, 'and', temp, 'K')

#Temp of sky around CasA
temp_b = np.full((1024,), 10)

#Defining multiple flux reference points and lists to fit
flux_625 = (4670/2)*(10**(-26))
flux_710 = (4240/2)*(10**(-26))
flux_780 = (3870/2)*(10**(-26))
flux = np.array([flux_550, flux_625, flux_710, flux_780])
fluxfreq = np.array([550, 625, 710, 780])
freqlist = np.arange(400, 800, 400/1024)

#Defining errors to each reference point
flux_550_error = flux_550*0.032
flux_625_error = flux_625*0.033
flux_710_error = flux_710*0.035
flux_780_error = flux_780*0.035
fluxerrorlist = np.array([flux_550_error, flux_625_error, flux_710_error, flux_780_error])

#Linear fit to flux vs frequency relationship
coeff2, var_matrix2 = curve_fit(y, fluxfreq, flux)
fluxfit = y(freqlist, coeff2[0], coeff2[1])

#Plotting fit
plt.figure(figsize=(10,8))
plt.plot(fluxfreq, flux, '.', color='k', label='Reference points')
plt.plot(freqlist, fluxfit, color='r', label='Linear fit')
plt.errorbar(fluxfreq, flux, xerr=0, yerr=fluxerrorlist, linestyle='none', color='g', label='Error', capsize=2)
plt.title('Linear Fit of Flux vs Frequency')
plt.xlabel('Frequency (MHz)')
plt.ylabel('Flux (SI units)')
plt.legend()
plt.show()

#Getting power and temperature as a function of freq
powerfit = fluxfit*A_eff[::-1]
tempfit = powerfit/kb
```

```
#Defining error for power and temp
power_error = fluxfit*A_eff_error
temp_error = power_error/kb

#Plotting power fit vs frequency
plt.figure(figsize=(10,8))
plt.plot(freqlist, powerfit, '.', label='Power')
plt.errorbar(freqlist, powerfit, xerr=0, yerr=power_error, linestyle='none', color='g', label='Error', capsize=2)
plt.title('Power vs Frequency fit')
plt.xlabel('Frequency (MHz)')
plt.ylabel('Power')
plt.legend()
plt.show()

#Plotting temp fit vs frequency
plt.figure(figsize=(10,8))
plt.plot(freqlist, tempfit, '.', label='Temperature')
plt.errorbar(freqlist, tempfit, xerr=0, yerr=temp_error, linestyle='none', color='g', label='Error', capsize=2)
plt.title('Temp vs Frequency fit')
plt.xlabel('Frequency (MHz)')
plt.ylabel('Temperature (K)')
plt.legend()
plt.show()

#Finding the power at each freq bin
powersource = stokes[22] - stokes[0]

#Removing inf points from data set
for i in range(len(tempfit)):
    if tempfit[i] == np.inf:
        tempfit[i] = tempfit[i-1]
    else:
        tempfit[i] = tempfit[i]

#So now fit a line for every frequency
coeff3 = np.zeros((2, 1024))
var_matrix3 = np.zeros((2, 2))
for i in np.arange(1024):
    coeff_temp, var_matrix_temp = curve_fit(y, [10, tempfit[::-1][i]], [stokes[::][0,i], powersource[::][i]]))
    coeff3[0,i] = coeff_temp[0]
    coeff3[1,i] = coeff_temp[1]
```

```
#Noise vs Frequency plot
plt.figure(figsize=(10,8))
plt.plot(freqlist, (coeff3[1]/coeff3[0]), ls = ‘none’, marker = ‘.’)
plt.errorbar(freqlist, (coeff3[1]/coeff3[0]), xerr=0, yerr=noise_error, linestyle='none', color='g', label='Error', capsize=2)
plt.title('Noise Temperature')
plt.xlabel('Frequency (MHz)')
plt.ylabel('Noise Temperature (K)')
plt.ylim(0,1000)
plt.show()

#Converting volts to ADC bits
Gain_ADC = (1023/5)*coeff3[1]

#Gain (ADC bits^2/K) vs Frequency plot
plt.figure(figsize=(10,8))
plt.plot(freqlist[::-1], Gain_ADC, ls = ‘none’, marker = ‘.’)
plt.errorbar(freqlist[::-1], Gain_ADC, xerr=0, yerr=gain_error, linestyle='none', color='g', label='Error', capsize=2)
plt.title('Gain vs Frequency')
plt.xlabel('Frequency (MHz)')
plt.ylabel('Gain (ADC bits^2/K)')
plt.ylim(0,3000)
plt.show()
```

8.4 Python Code for B0329+54 Analysis (Figures 7-12):

```
#loading data from .dat file
rawdata = open('/Users/ayush/Desktop/School/University/Third Year/AST326/Lab 4/AST326_ARO_B0329+54.dat', 'rb')

#Reading and loading header data into variables
header_data = struct.unpack('=iiiidiiild', rawdata.read(48))
ARO_length, ARO_header, ARO_samples, ARO_dtype, ARO_cadence, ARO_frequency, ARO_elements, ARO_summed, ARO_idx0, ARO_utc0 = header_data

#Additional header data
info_header = rawdata.read(ARO_frequency*4*2)
freqlist = np.fromstring(info_header[:ARO_frequency*4*2], dtype=np.float32)
freqlist = freqlist.reshape(-1,2)
freqs = np.mean(freqlist, axis=1)
dt = ARO_cadence*ARO_summed
elemlist = np.fromstring(info_header[ARO_frequency*4*2:], dtype=np.uint8)
```

```
#Reading and appending data
data = []
for i in np.arange(10000):
    idx, azimuth, elevation, dataset = read_data(rawdata)
    data.append(dataset)
data=np.array(data)
t = dt*np.arange(data.shape[0])

#Raw Waterfall Plot from data
plt.figure(figsize=(12,8))
plt.imshow(10*np.log10(data), extent=[np.amin(freqs), np.amax(freqs), np.amin(t), np.amax(t)], aspect='auto',
vmax=20, vmin=0, cmap='gray')
plt.xlabel('Frequency (MHz)')
plt.ylabel('Time (s)')
plt.title('Raw Waterfall Plot for B0329+54')
plt.colorbar().set_label('Power', rotation=90)

#Renormalizing each band
data_median = np.nanmedian(data, axis=0)
data_min = np.nanmin(data, axis=0)
data_normalize = data/data_median[np.newaxis,:]

#Normalized (over frequency) Waterfall Plot
plt.figure(figsize=(12,8))
plt.imshow(10*np.log10(data_normalize), extent=[np.amin(freqs), np.amax(freqs), np.amin(t), np.amax(t)],
aspect='auto', vmin=-1, vmax=1, cmap='gray')
plt.xlabel('Frequency (MHz)')
plt.ylabel('Time (s)')
plt.title('Waterfall Plot Normalized Over Frequencies')
plt.colorbar().set_label('Power', rotation=90)

#Reducing some RFI in data
plt.figure(figsize=(10,8))
plt.semilogy(t, np.sum(data, axis=1), '.')
plt.xlabel('Time (s)')
plt.ylabel('Power')
plt.title('Average Power in Detector vs Time')
plt.show()

#Removing RFI and setting folding variables
period = 0.714519699726
```

```
highfreq = np.where(data.sum(axis=1) > 8000)
nfreq = 1024
pbins = 500
wfold = np.zeros([pbins, nfreq])
cfold = np.zeros([pbins, nfreq])

highpower2 = highfreq[0].tolist()
highpower2.append([0])
for i in np.arange(data.shape[0]):
   if i == highpower2[0]:
      highpower2.pop(0)
      continue
   phase = np.int(pbins*np.modf(i*dt/period)[0])
   wfold[phase, :] += data[i, :]
   cfold[phase, :] += 1

#Checking the folded data
data_fold = wfold/cfold
data_median = np.nanmedian(data, axis=0)
data_divmedian = data_fold/data_median[np.newaxis, :]
plt.figure(figsize=(12,8))
plt.imshow(10*np.log10(data_divmedian), extent=[np.amin(freqs), np.amax(freqs), 0, 1], aspect='auto', vmax=1,
vmin=0, cmap='gray')
plt.xlabel('Frequency (MHz)')
plt.ylabel('Pulse Phase')
plt.title('Waterfall of Data folded on Pulsar Period')
plt.colorbar().set_label('Power', rotation=90)

#Computing flux (Jy) using gain from earlier
ARO_flux = kb*(data_divmedian[1]/Gain_ADC)*(10**(26))

#Flux vs Frequency plot
plt.figure(figsize=(10,8))
plt.plot(freqlist, ARO_flux, '.', color='b')
plt.title('Flux vs Frequency')
plt.xlabel('Frequency (MHz)')
plt.ylabel('Flux (Jy)')
plt.show()

#Computing typical flux across the band
print('Typical flux across 400-800Mhz is: ', np.mean(ARO_flux), 'Jy')
```

```
#Defining constants for de-dispersion
k = 4.15*10**(-3)
dm = 26.7641

#Defining dispersion equation to correct for curve
dispersion = k*dm*((freqs/1000)**(-2))

#Setting up empty arrays for loop
data_divmedian2 = data_divmedian*0
data_disperse = data_fold*0

#Looping to de-disperse data
for i in np.arange(nfreq):
   data_divmedian2[:, i] = np.roll(data_divmedian[:,i], - int(dispersion[i]/period*pbins))
   data_disperse[:,i] = np.roll(data_fold[:,i], - int(dispersion[i]/period*pbins))

#Removing LTE noise by cutting at 740MHz
data_divmedian_cut = data_divmedian[:,0:830]
data_divmedian2_cut = data_divmedian2[:,0:830]

#Plotting folded and de-dispersed data
plt.figure(figsize=(12,8))
plt.imshow(10*np.log10(data_divmedian2/np.median(data_divmedian2,axis=0)[np.newaxis,:]),
extent=[np.amin(freqs), np.amax(freqs), 0, 1], aspect='auto', vmax=0.5, vmin=-0.1, cmap='gray')
plt.xlabel('Frequency (MHz)')
plt.ylabel('Pulse Phase')
plt.title('Waterfall Plot after Folding and De-Dispersion')
plt.colorbar().set_label('Power', rotation=90)

#Summing over frequency bins to get flux per pulse phase
pulse = np.zeros(500)
for i in range(500):
   pulse[i] = np.sum((data_divmedian2_cut/np.median(data_divmedian2_cut, axis=0))[i,:])
pulse = pulse/np.mean(pulse)

#Plotting Folded Pulse Profile
plt.figure(figsize=(10,8))
plt.plot(np.arange(0, 1, 1/500), pulse)
plt.title('Folded Pulse Profile')
plt.xlabel('Pulse Phase')
plt.ylabel('Flux Relative to Mean')
plt.show()
```

```
#Summing over frequency bins to get flux per pulse phase without de-dispersion
pulse2 = np.zeros(500)
for i in range(500):
   pulse2[i] = np.sum(data_divmedian_cut[i,:])
pulse2 = pulse2/np.mean(pulse2)

#Plotting Folded Pulse Profile without De-dispersion
plt.figure(figsize=(10,8))
plt.plot(np.arange(0, 1, 1/500), pulse2)
plt.title('Folded Pulse Profile without De-Dispersion')
plt.xlabel('Pulse Phase')
plt.ylabel('Flux Relative to Mean')
plt.show()

#Redoing dispersion with non-folded dataset
data_dd_nofold = data_normalize*0
ddatad = data*0
for i in np.arange(nfreq):
   data_dd_nofold[:, i] = np.roll(data_normalize[:,i], - int(dispersion[i]/dt*period))
   ddatad[:,i] = np.roll(data[:,i], - int(dispersion[i]/dt*period))

#Removing LTE noise by cutting at 740MHz
data_dd_nofold_cut = data_dd_nofold[:,0:830]

#Plotting folded and de-dispersed data
plt.figure(figsize=(12,8))
plt.imshow(10*np.log10(data_dd_nofold_cut), extent=[np.amin(freqs), np.amax(freqs), 1, 0], vmax=1.0, vmin=-
0.5, aspect='auto', cmap='gray')
plt.xlabel('Frequency (MHz)')
plt.ylabel('Phase within Pulse Period (s)')
plt.title('Waterfall Plot after De-Dispersion')
plt.colorbar().set_label('Power', rotation=90)

#Summing over frequency bins to get flux per pulse phase with de-dispersion
pulse3 = np.zeros(10000)
for i in range(10000):
   pulse3[i] = np.sum(data_dd_nofold_cut[i,:])
pulse3 = pulse3/np.mean(pulse3)

#Plotting Folded Pulse Profile without De-dispersion
plt.figure(figsize=(10,8))
plt.plot(np.arange(0, 1, 0.0001), pulse3)
plt.title('Pulse Profile with only De-Dispersion')
plt.xlabel('Pulse Phase')
plt.ylabel('Flux Relative to Mean')
plt.show()

#Plotting histogram of flux
plt.figure(figsize=(10,8))
plt.hist(pulse3*1.572816790231749, bins = 10)
plt.title('Binned Flux Measurements over total Observing Time')
plt.xlabel('Flux (Jy)')
plt.ylabel('Number of Measurements')
plt.show()

#Making a histogram for only the pulse peaks
data_peaks = []
for i in range(len(pulse3)):
    if pulse3[i] > 1.02 and pulse3[i] > pulse3[i-1] and pulse3[i] > pulse3[i+1]:
        data_peaks.append(pulse3[i])

#Plotting threshold histogram
plt.figure(figsize=(10,8))
plt.hist(data_peaks, bins = 10)
plt.title('Binned Flux wrt Background (> 1.02)')
plt.xlabel('Flux Relative to Background')
plt.ylabel('Number of Measurements')
plt.show()

#Convolving data using average pulse profile
data_convolve = np.convolve(pulse3, pulse[108:119])
data_convolve = data_convolve/np.mean(data_convolve)

#Plotting convolved data set
plt.figure(figsize=(10,8))
plt.plot(np.arange(0, 1, 1/10010.00), data_convolve)
plt.title('Convolved Timestream with Average Pulse Profile')
plt.xlabel('Pulse Phase')
plt.ylabel('Flux Relative to Mean')
plt.ylim(0.99, 1.20)
plt.show()
```