Radio Imaging and Spectroscopy

Cosmic Messengers Lab (PHY/AST-3880)

By: **Esha Sajjanhar**

Prof. Dipankar Bhattacharya

Submission Date: December 1, 2024

Abstract

Radio interferometry lies at the heart of modern radio astronomy. In this report, we explore the fundamentals of imaging and spectroscopy using interferometric data from the Very Large Array (VLA). We perform various flagging, calibration, plotting and imaging tasks in order to obtain an image of a supernova remnant and a spectral image cube of a late stage stage. All analysis is performed using NRAO's Common Astronomy Software Applications (CASA) package.

1 Theoretical Background

A single radio dish focuses the incident electromagnetic signal onto the focus of the dish where a detector is present. The incoming signal is channeled through a superheterodyne receiver which amplifies and integrates the incoming signal. It may also shift the entire signal into another, standard frequency band known as the baseband. For incident signal with a wavelength λ and an antenna with diameter D, the resolution of the telescope is diffraction-limited to λ/D. Due to larger wavelengths in the radio band, achieving subarcsecond resolution requires impractically larger antenna diameter. Single dishes are hard to engineer, maintain, and point. Thus, to achieve high resolution while retaining our ability to maneuver and point the dishes, we use radio interferometry.

1.1 Radio Interferometry

The goal of radio interferometry is to map the intensity distribution on the sky by using multiple identical dishes separated by some distance. We assume that the sky can be decomposed into an array of point sources which make up the total intensity distribution. The intensity distribution along the direction $\hat{s}$ is $I_\nu(\hat{s})$ where ν is the operating frequency of the telescope.
Let's consider two dishes separated by a vector $\vec{b}$ pointing at a source which is located along direction $\hat{s}$. Any wavefront arriving at the second dish will have a path delay (known as geometric delay) as compared to the first, indicated by τ_g in Figure 1. Each antenna is connected to a detector which produces a voltage due to the incident electric field. Any line perpendicular to $\hat{s}$ will identify an equiphase locus, i.e. a wavefront.

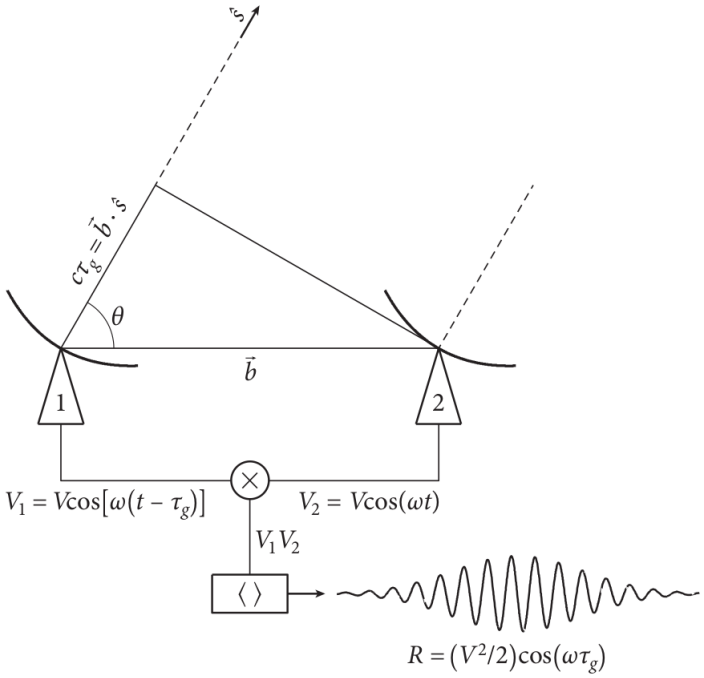


Figure 1: Schematic diagram of a simple two antenna interferometer (from Condon, 2016).

Each antenna will produce a voltage proportional to the incident electric field. Let d be the distance from antenna 2 to the source. Then, the distance from antenna 1 to the source is $d+\Delta$ where $\Delta = c\tau$. Let the the electric field arriving at each antenna at time t be E_1, E_2 respectively. Then,

$$E_1 = E_0\, e^{i(\omega t - kd - k\Delta)}, \tag{1}$$

$$E_2 = E_0\, e^{i(\omega t - kd)}, \tag{2}$$

where E_0 is the amplitude of the incident electric field, k is the wavenumber, and ω is the angular frequency. Due to these incident fields, the antennae will produce,

$$V_1 = G_1 E_0 \cos(\omega t - kd - k\Delta), \tag{3}$$

$$V_2 = G_2 E_0 \cos(\omega t - kd), \tag{4}$$

where G_1, G_2 are the complex gains of each of the antenna systems. These voltages are then brought together in a multiplier. The multiplier output in this case gives us,

$$\begin{aligned} V_1V_2 &= G_1G_2E_0^2 \cos(\omega t - kd)\cos(\omega t - kd - k\Delta) \\ &= \frac{1}{2}G_1G_2I_\nu(\hat{s})\left(\cos 2\omega t - 2kd - k\Delta + \cos k\Delta\right). \end{aligned} \tag{5}$$

Then, taking a time average, we get,

$$\langle V_1V_2\rangle = \frac{1}{2}G_1G_2I_\nu(\hat{s})\cos k\Delta \equiv R_1. \tag{6}$$

R_1 contains information about the phase difference between the two antennas. This is the real part of the correlation function which is a complex quantity which describes the relationship between the electric field received by each antenna. To characterise the imaginary part of the correlation, we

perform a second multiplication after delaying the signal in one arm by a 90° phase. With this phase delay, we get,

$$V_1V_2 = G_1G_2E_0^2\cos(\omega t - kd - k\Delta)\sin(\omega t - kd)$$
$$= \frac{1}{2}G_2G_2I_\nu(\hat{s})\left(\sin(2\omega t - 2kd - k\Delta) + \sin k\Delta\right).$$

Then, on taking a time average, we get,

$$\langle V_1V_2\rangle = \frac{1}{2}G_1G_2I_\nu(\hat{s})\sin k\Delta \equiv R_2, \tag{7}$$

where R_2 gives the imaginary part of the correlation. Using both R_1, R_2 we can define a visibility $\mathcal{V} \equiv R_1 - iR_2$ which has amplitude $|\mathcal{V}| = \sqrt{R_1^2 + R_2^2}$ and phase $\phi = \tan^{-1}(\frac{R_2}{R_1}) = k\Delta$. Then, $\mathcal{V} = |\mathcal{V}|e^{-ik\Delta}$.

Now, instead of the diameter of the dish, the resolution depends on the baseline. We define the spatial frequency as the baseline in units of λ as $\vec{u} \equiv \frac{\vec{b}}{\lambda}$. However, this is just a 1-dimensional picture. Extrapolating the above analysis to two dimensions, we can define two spatial frequencies u, v. These are direction cosines and are thus dimensionless. We can similarly define the plane of the sky in terms of direction cosines l, m. Then, the intensity distribution on the sky is $I_\nu(\hat{s}) = I_\nu(l, m)$.

Now, $\vec{b} = (u\lambda, v\lambda)$ such that $\vec{b}\cdot\hat{s} = (ul + vm)\lambda$ and $\phi = k\Delta = \frac{2\pi}{\lambda}(\vec{b}\cdot\hat{s}) = 2\pi(ul + vm)$. Now, we know that any pair of antennas measure the visibility function $\mathcal{V}(u, v)$ and not the intensity distribution on the sky $I_\nu(l, m)$. We know that $\mathcal{V} = |\mathcal{V}|e^{-ik\Delta}$. Using this,

$$\mathcal{V}(u, v) = \frac{1}{2}G_1G_2I_\nu(l, m)e^{i\phi}$$
$$= \frac{1}{2}G_1G_2I_\nu(l, m)e^{-2\pi i(ul+vm)}. \tag{8}$$

This gives us a relationship between the visibility function in one direction ($\hat{s}$) in the sky and the intensity distribution in that direction. We integrate this quantity over the entire sky to get,

$$\mathcal{V}(u, v) = \frac{1}{2}G_1G_2\iint I_\nu(l, m)e^{-2\pi i(ul+vm)}\,\mathrm{d}\Omega$$
$$= \frac{1}{2}G_1G_2\iint I_\nu(l, m)\frac{e^{-2\pi i(ul+vm)}}{\sqrt{1 - l^2 - m^2}}\,\mathrm{d}l\,\mathrm{d}m. \tag{9}$$

We can see that the visibility function is a Fourier transform of the intensity distribution in the sky. This is known as the **van Cittert-Zernike theorem** and is the basis of radio interferometry. Each baseline (i.e. each u, v pair) gives us one Fourier component of the intensity distribution in the sky. Radio interferometers try to sample as many points in the uv plane as possible in order to reconstruct $I_\nu(l, m)$ faithfully. This is accomplished through *aperture synthesis*. As we have seen, the important quantity in this process is $\vec{b}\cdot\hat{s}$, i.e. the baseline 'seen' by the source or the 'projected' baseline. This means that in addition to having many antennas with varying distances between them and considering all possible pairs to get multiple baselines, interferometers can use the rotation of the Earth to sample a larger number of projected baselines. These techniques improve uv coverage and help reconstruct the source intensity distribution.

In practice, we never have perfect uv coverage. Due to finite sampling, there are gaps in the uv coverage. Let $S(u, v)$ be the sampling function in the uv plane. Then, what we measure in practice is $\mathcal{V}(u, v)S(u, v)$. Using van Cittert-Zernike theorem, the intensity distribution is then given by,

$$I_\nu^D(l, m) = \iint \mathcal{V}(u, v)S(u, v)e^{2\pi i(ul+vm)}\mathrm{d}u\mathrm{d}v. \tag{10}$$

We can define the *synthesized beam* of the telescope as $B(l,m) = \iint S(u,v)e^{2\pi i(ul+vm)}dudv$. Then, $I_\nu^D(l,m)$ gives the 'dirty' intensities which are the true intensities ($I_\nu(l,m)$) convolved with the synthesized beam. Finally, due to finite size of the antenna dishes, we obtain 'side-lobes' in the antenna response. That is to say, the antenna response contains contributions other than the ones we are interested in. The response of a single dish to a point source is known as the 'primary beam' or the antenna reception pattern of a telescope. This is multiplied with the observed intensities. That is, for antenna reception pattern $A_\nu(l,m)$, we have,

$$I_\nu^D(l,m) = [A_\nu(l,m)I_\nu(l,m)] * B(l,m). \tag{11}$$

In practice, the measured visibilities are the Fourier transform of these dirty intensities.

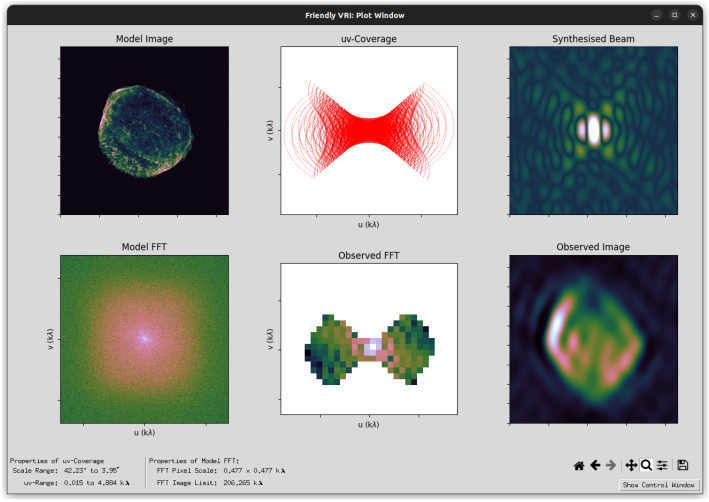


Figure 2: An example of the relationship between *uv* coverage, clean image, dirty image and the synthesised beam (from FriendlyVRI).

1.2 Radio Imaging

Radio imaging involves using visibilities in the *uv* plane to find the intensities in the *lm* (sky) plane. However, before we can begin imaging, we must first identify any bad data and obvious noise present in the observation. This is accomplished through a process known as 'flagging'. We then need to find the value of the complex antenna gain G for each antenna. Further, we need to express our measured voltages in standardised units. These are accomplished through various kinds of calibration.

Flagging involves identifying outliers in the measured visibilities which correspond to noise, known as 'radio frequency interference' (RFI). Mobile communication signals, wifi, radio, as well as higher harmonics of certain electrical appliances can all contribute to RFI. These are typically much stronger than the astronomical sources being observed and occur as very sharp features in a time series. There exist some algorithms which automatically find RFI in observations, but radio observations typically require some amount of manual flagging. Flagging can also be used to omit bad data segments (such as a non-functioning antenna) from the data used for analysis. Flagging is typically performed by adding a 'flag' to the bad data in the table of visibilities and thus does not involve deleting or modifying any part of the recorded data. This means that it is easy to remove flags as well. Information about flags is generally stored in a 'flag table'.

Once flagging is complete, we move on to calibration. In order to find the true intensity of the target, we need to observe some calibration sources. First, we have the flux or amplitude calibrator. This is an unresolved (point) source of known intensity from a set of identified standard sources. We can use this source to identify our antenna gains. We also use this to identify frequency-dependent changes in antenna gains through bandpass calibration. So far, we have used a bright source for calibration which allows us to correct for the broad features. However, our target sources sees a different patch of the ionosphere than the flux calibration source. In order to correct for this, we use the phase calibrator which is a calibration source chosen to be close enough to the source that it sees the same ionosphere. Using the phase calibrator observations to find the phase corrections for the target source uses the assumption that between two observations of phase calibrator, the ionospheric variations are smooth and can be interpolated to the target observations.
Now, we begin imaging. We start by using an inverse Fourier transform to obtain the dirty image $I_\nu^D(l,m)$ from the measured visibilities. We then need to deconvolve the synthesised beam in order to obtain $A_\nu(l,m)I_\nu(l,m)$. We do so using the CLEAN algorithm which relies on finding the highest intensity regions in the image. This algorithm identifies the brightest sources in the image, adds these to a table and iterates this process until the residual flux in the image can be considered to be noise-level. The table then gives us all the point-sources in the sky. In general, the algorithm does not remove the entire flux from the detected peak flux region in a single iteration. This allows better identification of real features and helps identify them separately from the noise.

1.3 Image Cubes

In order to perform spectral analysis with radio data, we create 'image cubes' which are 3-D data structures which contain intensity information against two dimensions of coordinates as well frequency. A spectral frequency is defined by the centre of the bandwidth of observation of the telescope.

2 Data and Software Used

All analysis was conducted using NRAO's CASA (Common Astronomy Software Applications) package. The data used here comes from NRAO's CASA tutorials.
The data used for continuum imaging is a 2010 VLA D-array observation of the supernova remnant 3C 391 at 4.6 and 7.5 GHz each with a bandwidth of 128 MHz.
The data used for spectral analysis is a 2010 VLA D-array observation of the late stage star IRC+10216 with 64 channels in two windows- each centered on one spectral line. The two spectral lines in our data are HC3N and SiS.

3 Analysis

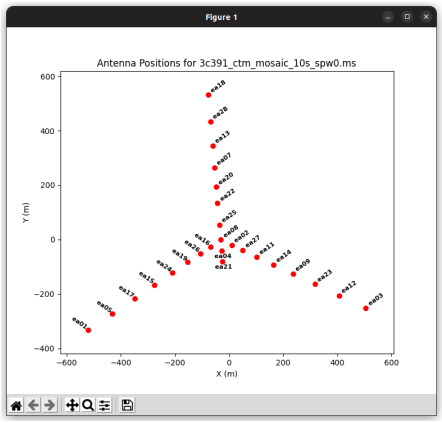
3.1 Continuum Imaging

To begin with, we go through the observation log and note down any anomalies recorded there such as non-functioning antennae, mechanical errors, data loss errors, etc. These are important because they help us understand what parts of the data would be contaminated and would thus need to be flagged. We open CASA in our data directory with the command `casa`. This launches the python-based

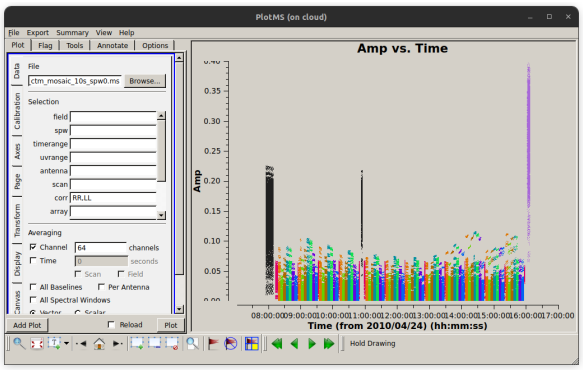
prompt within which we can perform all the necessary data analysis tasks. All tasks discussed further are performed within the CASA prompt unless otherwise specified. We used `obs_dict` to define an observation dictionary containing important information about the observation metadata which can be accessed using `listobs`. The observation data (visibilities) is stored within a 'measurement set' or a .ms directory alongside the observation metadata.

This observation contains scans of multiple different fields, each numbered so that scans of the same field can be easily identified. The target source, 3C391, is imaged in 7 separate fields. The image of the full source can be obtained as a mosaic of these 7 fields. In addition to the target source, the observation contains scans of an amplitude calibrator and a phase calibrator. The phase calibrator is observed before each set of target scans. Here, the phase calibrator observed is J1822-0938, and the amplitude (or flux) calibrator is J1331+3030. This observation contains only one spectral window with 63 channels each with bandwidth 128 MHz. We use only RR and LL correlations for intensity measurements.

Now, we can use `plotants` to see the antennae that were used in this observation and their configuration. We find that the manual log has likely misreported the unavailable antenna (log reported antenna 13 dysfunctional, while the plot shows antenna 10 missing). We take note of this before we begin flagging.



(a) Antenna configuration for the observation (using `plotant`).



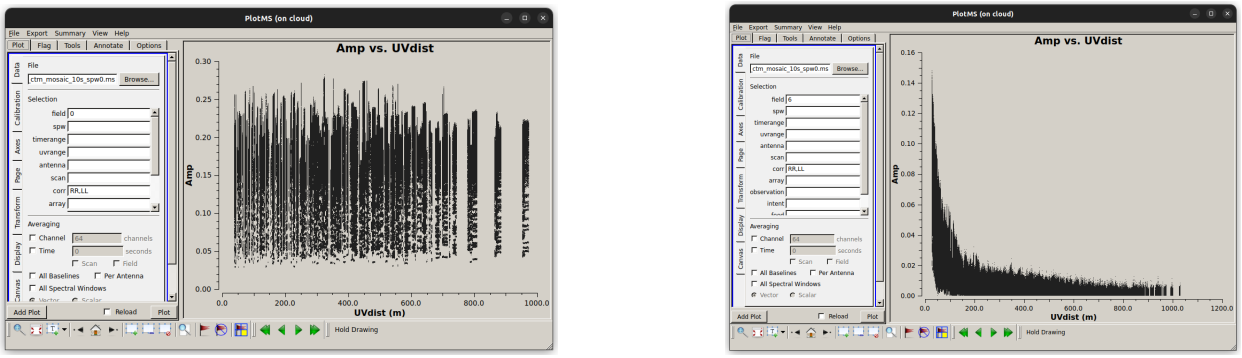
(b) Lightcurve of the observation showing the various antennae.

Figure 3: Plots showing the antenna configuration used and the lightcurve of the observation.

In general, the first scan is a 'setup' scan and thus doesn't contain useable data. We begin by flagging this scan for all antennae using the task `flagdata`. Next, we use the same command to flag all correlations involving antennas 13 and 15 because these had data loss issues according to the observer's log. The antennas must move to a new pointing direction before every scan, which results in some residual oscillations in the dish due to the mechanical movement. For this reason, it is customary to flag out the first 10 seconds of each scan in VLA observations. This type of flagging where something is applied to all scans is known as 'quack' flagging. Hence, we use the `flag` command with `mode='quack'` and `quackmode='beg'` (i.e. 'beginning') such that the first 10 seconds of each scan are flagged.

Once this is done, we want to plot the data segments and manually verify that there aren't any remaining bad data segments or RFI effects. We use the `plotms` command to plot the data segments. The plot we obtain is the equivalent of a light curve. There is some statistical fluctuation in the

visibility amplitude due to averaging which gives the width of each of the lines.
We can use the plotting window to adjust plot parameters and thus modify the plot. For instance, we can look at visibility amplitude as a function of baseline distance. For very narrow sources, we expect the visibilities to be flat across baselines. If they are not flat, the observation has been able to resolve the source and we should be able to see structures. We should see variation in visibility amplitude with baselines. In order to test whether our flux calibrator is truly a point source, we can set the field to 0, remove channel averaging, and set our x-axis to uv distance.



(a) uv amplitude plot for the flux calibrator.

(b) uv amplitude plot for source observation.

Figure 4: *uv* amplitude plots.

We see that the uv amplitude is largely constant which means that structure is not resolved even at the largest baseline used. Now, for our target, we expect higher amplitude at lower baselines and lower at higher baselines because it is known to be a large structure.
Now, in order to search for RFI or other bad data segments, we we want to look at both the time distribution of data and the frequency dist of data for each baseline. We first remove channel averaging in the plotting window, then set antenna to ea01, set x-axis to frequency and y-axis to phase, and finally choose iteration by baseline.

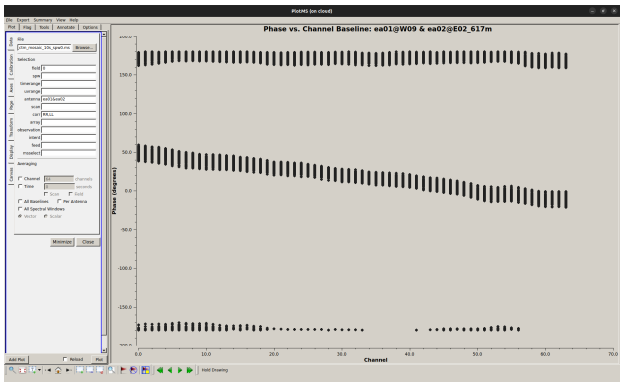


Figure 5: Phase and channel plot showing the need for bandpass calibration.

On plotting phase versus channel, we find a gradient of phase with frequency. This is due to changes in antenna phase and the gradient should disappear once we apply bandpass calibration. We see two lines in the plot corresponding to the two correlations (RR,LL) that we are using. Phase data contains information about structure of the source, while amplitude data contains information about flux of the source. Phase is thus the most important quantity to check when flagging data.
In order to aid us in flagging, the data, we can use the function `flagdata` with mode='tfcrop' which

can automatically search for outliers and thus flag bad data segments and RFI according to specified thresholds. Finally, we want to apply antenna position corrections using the command `gencal`. Now we can move on to calibration.

We start by applying flux calibration. We first set the flux of the calibrator source using the command `setjy`. Now, we perform each of bandpass and phase calibration in two stages- a coarse stage and a finer stage. We use the command `gaincal` to perform initial phase calibration. For this coarse stage, we use only the middle channels as being representative of the entire dataset and use an antenna close to the centre of the array as our reference antenna. We now perform the finer stage of phase calibration using `gaincal` again, this time with all the spectral channels. Now we can perform bandpass calibration. At the coarse stage, we correct for the phase delays corresponding to the geometric delay to a given source, despite the compensation at the centre of the field. This is performed using corrtype='k' which refers to relative delays between antennas, with respect to a reference antenna. The task `gaincal` is again used for this purpose. We then correct for frequency-dependent delays in the finer stage. This is done using the task `bandpass`. We then use the task `gaincal` to perform phase calibration.

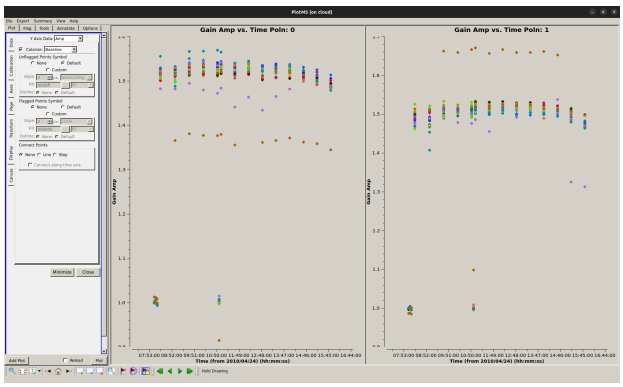


Figure 6: Phase and amplitude plot after performing bandpass and phase calibration.

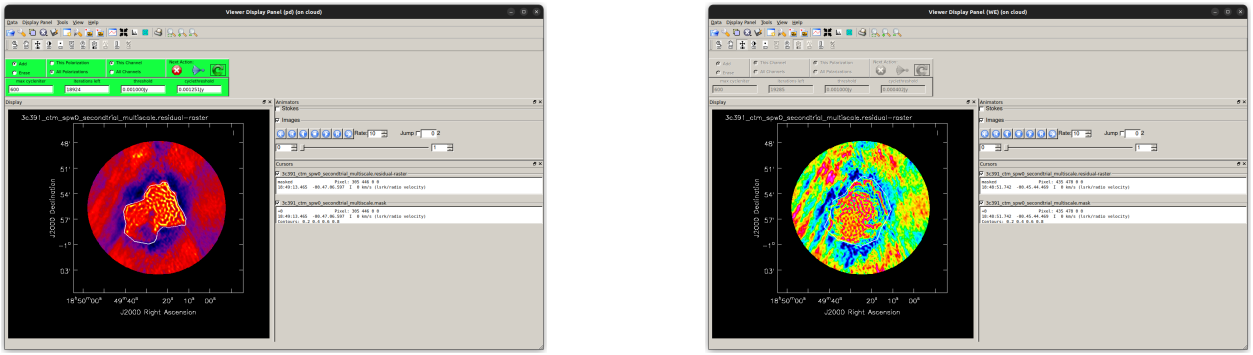
Now, we finally just need to make sure that all the sources observed have the right fluxes. We can use our flux calibrator to set the flux scale for our target observations using the command `myscale`. Once we apply the flux calibration to all the target fields, we will be able to image the target in units of Janskys. We use `applycal` to apply the calibration to the required fields - phase calibrator and all source fields.

We can now begin imaging our source- this involves performing both fourier trasforms and deconvolutions to obtain an intensity distribution from our set of visibilities. These operations are performed by `TCLEAN`. We first split off the part of the measurement set that is relevant to imaging (the target fields) into a new measurement set using the command `split`. Since we are only interested in total intensity imaging, we can retain only the RR and LL polarizations in our new (split) dataset. We run the `TCLEAN` command on our split dataset interactively. This allows us to define regions within which the algorithm will search for CLEAN components. The CASA viewer window allows us to define an arbitrary polygon region within which this can be done.

Once we choose a polygon region, we can run CLEAN for a specified number of iterations after which it images the remaining intensities and returns to the interactive window. We can now modify the polygon region if needed and start another such cycle. Such iterations must be repeated until the overall noise level in the image (reported on the CASA logger window at the end of each cycle) falls

to a low enough value.

After deconvolving and imaging, we need to perform primary beam correction using `pbcor`. The last step in image processing is self-calibration, but that was not performed for the data used here. We can view the final image using CARTA which also allows us to obtain statistics for the image or regions within it.



(a) The CASA viewer interface showing a polygon region for TCLEAN.

(b) Residuals after performing several iterations of TCLEAN.

Figure 7: TCLEAN window showing various stages of cleaning.

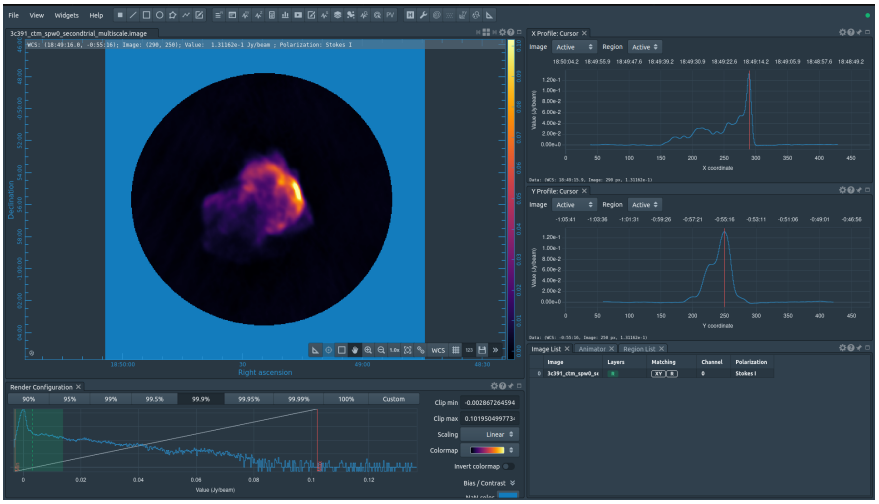


Figure 8: Final continuum image displayed on CARTA showing the intensity profiles for the central bright regions using X and Y slices.

3.2 Spectroscopy

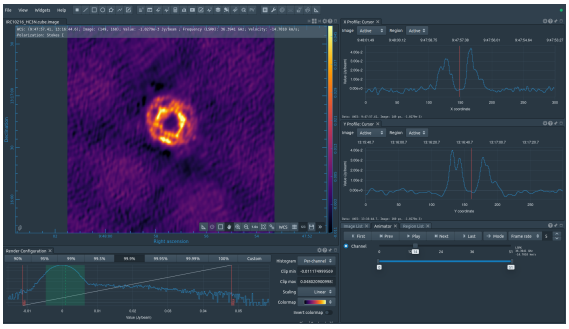
Here, we perform data reduction for two spectral lines (H3CN and SiS) for the star IRC+10216 observed using the VLA. Once again, we first read through the observation log and identify data segments that need to be flagged. We can use the task `listobs` to find details of the observation, and then plot the participating antennae using `plotants`. We can then use `flagdata` to flag the bad antennas. We perform antenna position corrections using `gencal`. This command can also be used to find how each antenna behaves as a function of source elevation. We also need to identify the atmospheric conditions at the time of the observation since this affects the opacity of the observation. This is done using the command `plotweather` to characterise the weather conditions and `gencal` to

generate a calibration table for these. We then set up a model for the flux calibrator using `setjy`. Once this has been done, we can proceed to performing calibration.

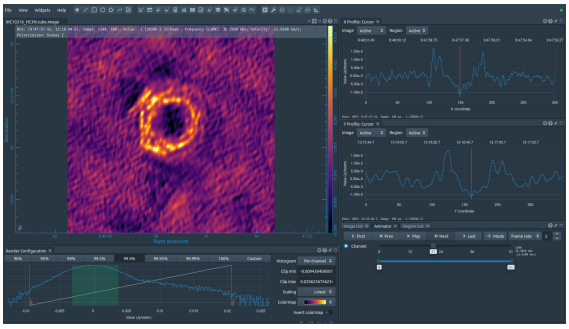
We use the same steps as with the previous section to perform bandpass and complex gain calibration. However, since this observation is at a higher frequency, the primary beam of each antenna is much smaller. As the antenna tracks the source from rise to set, this results in some distortions in the primary beam due to different gravitational loading on the antenna. We use calibration to correct for these effects. This wasn't necessary in the previous section (continuum imaging) because we were working with lower frequency data which had a larger primary beam. Further, we need to correct for the atmospheric optical depth as a function of elevation. Such a correction is important for this data because the atmosphere along the line of sight at any moment has some absorption which may affect the spectral lines we are observing.

Now, we want to split our data to only retain the target since we don't want to image the calibrators. We retain the RR, LL correlations and plot them as a function of spectral channel. This creates a 'data cube'. However, this cube contains continuum contributions in addition to the spectral line we are interested in. In order to remove the continuum contribution, we create a fit to all channels except the one which contains the line and subtract this fit from the all channels. We do this in the uv domain using the task `uvcontsub` for both the spectral channels we are interested in. When creating a continuum baseline fit, we avoid using the first and last few channels since they show some abrupt drop-off.

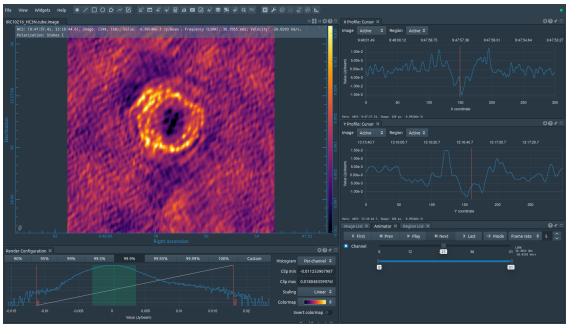
Finally, we run `TCLEAN` for both spectral line images separately. We want to clean the image until the residuals are of the order of 0.5 mJy. We can export the .image directory as a FITS file for greater flexibility in image analysis.



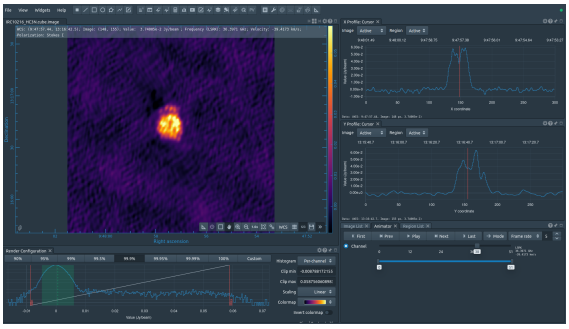
(a) Channel 14



(b) Channel 21



(c) Channel 25



(d) Channel 38

Figure 9: Various channels of the image cube showing the H3CN image for different channels.

4 References

1. NRAO CASA guides *VLA Continuum Tutorial 3C391-CASA6.4.1*
2. NRAO CASA guides *VLA high frequency Spectral Line tutorial - IRC+10216-CASA6.2.0*
3. *Essential Radio Astronomy* (2016) by Condon and Ransom
4. *Synthesis and Imaging in Radio Astronomy* (1998) by Taylor, Carilli, Perley.

Appendix: CASA Commands

The CASA commands used to perform the analysis described above have been listed here. Many of the commands listed here were used multiple times in the analysis but have been listed only once for brevity.

Continuum Imaging

```
obs_dict = listobs(vis='3c391_ctm_mosaic_10s_spw0.ms')
```

```
plotants(vis='3c391_ctm_mosaic_10s_spw0.ms/')
```

```
flagdata(vis='3c391_ctm_mosaic_10s_spw0.ms', flagbackup=True, mode='manual', scan='1')
```

```
plotms(vis='3c391_ctm_mosaic_10s_spw0.ms', selectdata=True, correlation='RR,LL',
averagedata=True, avgchannel='64', coloraxis='field')
```

```
gencal(vis='3c391_ctm_mosaic_10s_spw0.ms', caltable='3c391_ctm_mosaic_10s_spw0.antpos',
caltype='antpos')
```

```
setjy(vis='3c391_ctm_mosaic_10s_spw0.ms', field='J1331+3030', standard='Perley-Butler 2017',
model='3C286_C.im', usescratch=True, scalebychan=True, spw='')
```

```
gaincal(vis='3c391_ctm_mosaic_10s_spw0.ms', caltable='3c391_ctm_mosaic_10s_spw0.G0all',
field='0,1,9', refant='ea21', spw='0:27~36', gaintype='G', calmode='p', solint='int', minsnr=5,
gaintable=['3c391_ctm_mosaic_10s_spw0.antpos'])
```

```
bandpass(vis='3c391_ctm_mosaic_10s_spw0.ms', caltable='3c391_ctm_mosaic_10s_spw0.B0',
field='J1331+3030', spw='', refant='ea21', combine='scan', solint='inf', bandtype='B',
gaintable=['3c391_ctm_mosaic_10s_spw0.antpos', '3c391_ctm_mosaic_10s_spw0.G0',
'3c391_ctm_mosaic_10s_spw0.K0'])
```

```
myscale = fluxscale(vis='3c391_ctm_mosaic_10s_spw0.ms', caltable='3c391_ctm_mosaic_10s_spw0.G1',
fluxtable='3c391_ctm_mosaic_10s_spw0.fluxscale1', reference='J1331+3030',
transfer=['J1822-0938'], incremental=False)
```

```
applycal(vis='3c391_ctm_mosaic_10s_spw0.ms', field='J1331+3030', gaintable=['3c391_ctm_mosaic_10s_
spw0.antpos', '3c391_ctm_mosaic_10s_spw0.fluxscale1', '3c391_ctm_mosaic_10s_spw0.K0',
'3c391_ctm_mosaic_10s_spw0.B0'], gainfield=['', 'J1331+3030', '', ''], interp=['', 'nearest',
'', ''], calwt=False)
```

Spectroscopy

```
flagdata(vis='day2_TDEM0003_10s_norx', mode='list', inpfile=["field='2,3' antenna='ea12'
timerange='03:41:00~04:10:00'", "field='2,3' antenna='ea07,ea08' timerange='03:21:40~04:10:00'
spw='1'"])
```

```
gencal(vis='day2_TDEM0003_10s_norx', caltable='gaincurve.cal', caltype='gceff')
```

```
myTau = plotweather(vis='day2_TDEM0003_10s_norx', doPlot=True)
```

```
setjy(vis='day2_TDEM0003_10s_norx', field='7', spw='0~1', scalebychan=True, model='3C286_A.im',
usescratch=True)
```

```
gaincal(vis='day2_TDEM0003_10s_norx', caltable='delays.cal', field='5', refant='ea02',
gaintype='K', gaintable=['antpos.cal','gaincurve.cal','opacity.cal'])
```

```
bandpass(vis='day2_TDEM0003_10s_norx', caltable='bandpass.bcal', field='5', refant='ea02',
solint='inf', solnorm=False,
gaintable=['antpos.cal','gaincurve.cal','opacity.cal','delays.cal','bpphase.gcal'])
```

```
applycal(vis='day2_TDEM0003_10s_norx', field='5',
gaintable=['antpos.cal','gaincurve.cal','opacity.cal','delays.cal','bandpass.bcal'],
gainfield=['','','','5','5'], calwt=False)
```

```
fluxscale(vis='day2_TDEM0003_10s_norx', caltable='amp.gcal', fluxtable='flux.cal',
reference='7', incremental=True)
```

```
split(vis='day2_TDEM0003_10s_norx', outputvis='IRC10216', field='3', correlation='RR,LL')
```

```
uvcontsub(vis='IRC10216', outputvis='IRC10216.contsub', field='', spw='', scan='', intent='',
array='', observation='', datacolumn='data', fitspec='0~1:4~14;51~59', fitmethod='gsl',
fitorder=0, writemodel=True)
```

```
statwt(vis='IRC10216.contsub', fitspw='0~1:15~50', excludechans=True, datacolumn='data')
```

```
tclean(vis='IRC10216.contsub', spw='0:7~58', imagename='IRC10216_HC3N.cube', imsize=300,
cell=['0.4arcsec'], specmode='cube', outframe='LSRK', restfreq='36.39232GHz',
perchanweightdensity=True, pblimit=-0.0001, weighting='briggs', robust=0, niter=100000,
threshold='3.0mJy', interactive=True, savemodel='modelcolumn')
```