
Outside CASA:

GMRT data are recorded in the LTA format. Initially, convert the data into the FITS format, using the tasks "listscan" and "gvfits" (outside casa):

listscan file.lta

gvfits file.log
mv TEST.FITS 0311_raw.fits

If you like, you can edit the listscan output file "file.log" to select channel ranges, sources, scans, etc, before running gvfits.

Inside CASA:

First, import the FITS file into the Measurement Set ("MS") format:

1. Run importuvfits (or importgmrt) to load the FITS file.

```
inp importuvfits              # This shows the inputs of the task.
fitsfile='0311_raw.fits'
vis='0311_raw.ms'
go
```

---

2. Next, check the contents of the MS with the task "listobs":

```
inp listobs
vis='0311_raw.ms'
go
```

Look closely at the CASA log file. Work out which sources are the flux calibrator, the phase

calibrator, the target, etc.

Check the frequency settings, and the number of channels. How much on-source time is there on the target?

Next, check the VLA calibrator manual to see if the phase calibrator is bright enough to be a bandpass calibrator. Does the phase calibrator have any UV range constraints?

---

3. Use the task "split" to extract a single channel for gain calibration. Used channel 100 here.

```
inp split
vis='0311_raw.ms'            # The input multi-channel MS.
outputvis='0311_raw.ch0'     # The output single-channel MS.
spw='0:100'                  # Use channel 100 from spectral window 0.
datacolumn='data'
go
```

---

4. To check on bad data, bad antennas, etc, plot the data using the task "plotms":

```
inp plotms
vis='0311_raw.ch0'
go
```

Click on "Data" on the extreme left, to select different data
(x-axis would typically be time or uvwave for a single-channel data set.
y-axis would typically be amplitude or phase, either "data" or "corrected".)

Click on "Display" to increase the symbol size, colourize correlations, etc.
Click on "Page" to iterate over baselines, antennas, etc.
Click on "Options" to increase the number of grid rows or columns.

(1) Identify dead antennas on the flux and phase calibrators.
(2) Identify bad times on the calibrators.
(3) Do NOT flag data in plotms!

---

5. The task "flagdata" is the workhorse to flag data in CASA. Initially, use manual inspection to excise dead antennas.

For example, to flag antenna W02 (26):

```
inp flagdata
vis='0311_raw.ch0'
mode='manual'
antenna='26'
go
```

To flag some bad timerange for all antennas, zoom in to the timerange in plotms to identify the start and end times of the bad time range. After this,

```
inp flagdata
vis='0311_raw.ch0'
mode='manual'
antenna=''
timerange='22:34:00~22:38:00'
go
```

Note: Flag only the calibrators at this stage.

---

6. Next, one sets the flux density scale using the task "setjy", and with one or more of the primary calibrators (3C286, 3C48, 3C147, 3C138):

```
inp setjy
vis='0311_raw.ch0'
field='0'              # Usually, the flux calibrator is the first source.
usescratch=True        # To create and use the MODEL_DATA column.
go
```

---

7. For gain calibration, we'll use the task "gaincalR" (written by Aditya Chowdhury). Note that this is NOT "gaincal", the regular CASA calibration task. "gaincalR" is better than gaincal!

Note that no separate phase calibrator was observed during the observations of 0311+430. We will treat 0311+430 as a phase calibrator in the analysis, and then use self-calibration to make a good image of the field.

```
inp gaincalR
vis='0311_raw.ch0'
caltable='0311.cal'
field='0,1'     # Solve for both the flux and phase calibrators (3C48, 0311+430).
uvrange='> 1.5km'    # This is to exclude the short central-square baselines.
solint='1 min'       # This averages the data for 1 minute, and then solves for the gains.
                     #  The solution interval should be chosen based on how bright the
                     #  calibrators are.
refant=1             # This uses C00 as the reference antenna for calibration.
go
```

Check the log file for the output of gaincalR. See how many solutions failed, and how many failed _during_ gaincalR, as opposed to failures because some antennas were flagged. Note that the GMRT measurement sets have 32 antennas, so 4 solutions will always fail.

7a. If things look funny with the calibration outputs, can run the task "clearcal" to remove the calibration, and start again.

```
inp clearcal
vis='0311_raw.ch0'
addmodel=True
go
```

---

8. Use "fluxscale" to get the flux density of the phase calibrator, from the flux calibrator.

```
inp fluxscale
```

```
vis='0311_raw.ch0'
caltable='0311.cal'
fluxtable='0311.flux'
reference=['0']        # The flux calibrator.
transfer=['1']         # The phase calibrator or phase calibrators.
go
```

---

9. Use "applycal" to apply the calibration to the target source and to all calibrators:

```
inp applycal
vis='0311_raw.ch0'
gaintable=['0311.flux']   # The output table of fluxscale has to be used here.
go
```

---

10. Inspect the calibrated data for the flux and phase calibrators, using plotms.

Choose "corrected" for the y-axis. Check if the amplitude is approximately constant versus "uvwave", and the phase is approximately zero (both with a spread). If there are outliers, check which baselines or antennas show the outliers (can mark them in plotms), and flag these baselines/times/antennas.

---

11. Use flagdata to flag any bad data identified with plotms. The approach below clips any outliers outside the range [35,42] for the flux calibrator:

```
tget flagdata
mode='clip'
field='0'
clipminmax=[35,42]
datacolumn='corrected'
go
```

---

12. Once flagging has been completed, and the data look clean, go back and repeat the calibration. Check the calibrated data again with plotms. If the data look fine, your basic gain calibration is done.

---

14. Run the task "repeatflag", also by Aditya Chowdhury, to apply the flags of the single-channel file to the multi-channel data set.

```
inp repeatflag
visfrom='0311_raw.ch0'
visto='0311_raw.ms'
go
```

---

15. Run applycal to apply the calibration to the sources of the multi-channel data set:

```
inp applycal
vis='0311_raw.ms'
gaintable=['0311.flux']   # The output table of fluxscale has to be used here.
go
```

---

16. Split out the calibrated data on the bandpass calibrators, to measure the system bandpass.

```
inp split
vis='0311_raw.ms'
outputvis='0311_bpass.ms'
field='0'                 # This assumes that only the flux calibrator will be used for the bandpass.
datacolumn='corrected'
go
```

---

17. Add the model data column to the bandpass dataset:

```
tget setjy
vis='0311_bpass.ms'
field='0'
go
```

---

18. To measure the antenna-based bandpass shapes, use the task "bandpassR" (from Aditya Chowdhury): Note: Use "bandpassR", and not the standard CASA task "bandpass".

```
inp bandpassR
vis='0311_bpass.ms'
caltable='0311.bpass'
field='0'                    # Use the flux calibrator to measure the bandpass.
uvrange='> 1.5 km'           # Exclude the central-square baselines while solving.
solint='inf'                 # This will solve for the bandpass in each scan.
refant=1
solnorm=True                  # Normalize the bandpass, setting amplitude=1 and phase=0 at
                              #  some channel (normchanrange below).
normchanrange='100'          # Normalize at the channel used for gain calibration.
dividebychanzero=False
go
```

---

19. Inspect the bandpass solutions with plotms, for each antennas. Check whether the solutions are smooth and whether they look noisy.

```
tget plotms
vis='0311.bpass'
go
```

Could also apply the bandpass solutions to the file from which the bandpass table was produced, to check whether the corrected data look clean. Then can clip out any outliers, and produce the bandpass table again. This is often an efficient way of editing out RFI.

```
tget applycal
```

```
vis='0311_bpass.ms'
gaintable='0311.bpass'
go
```

And then run plotms on the corrected data to look for outliers. After flagging (using mode='clip') with flagdata, re-run bandpassR, and proceed iteratively until things look clean.

Can also use "aoflagger" to flag data in the file "0311_bpass.ms".
"rfigui" can be used to inspect the data, in the time-frequency plane for each baseline. And can be used to test out flagging strategies. The strategies can then be applied with aoflagger.

---

20. Once the bandpass solutions look clean, apply the gain and bandpass calibration to the main MS, using applycal:

```
tget applycal
vis='0311_raw.ms'
gaintable=['0311.flux','0311.bpass']
source='1'                      # The target source.
interp=['linear','nearest'] # Usually better to use 'nearest' for the bandpass.
applymode='calflagstrict'   # I would advise this for "normal" calibration).
go
```

---

21. Next, split out the calibrated data on the target source into a new file:

```
tget split
vis='0311_raw.ms'
outputvis='day1.ms'
field='1'
datacolumn='corrected'
go
```

If you have multiple observing runs, carry out the above analysis separately for each observing run, to produce separate calibrated measurement sets "day1.ms", "day2.ms", "day3.ms", etc.

22. If there are multiple observing runs, combine the target source datasets with concat:

```
inp concat
vis=['day1.ms','day2.ms','day3.ms']  # Order the measurement sets by time.
concatvis='0311_full.ms'
dirtol='1 arcsec'                To make sure that all runs have the same source ID.
go
```

If you have a single observing run, ignore this step.

---

23. Inspect the new visibility data set with plotms. Try to identify bad time ranges and bad antennas, and remove them with flagdata, with mode='manual'. Can also use mode='clip' to remove egregiously bad data on individual baselines (usually bad due to RFI). In the example below, we're clipping the full data set at 10 Jy.

```
tget clip
vis='0311_full.ms'
mode='clip'
clipminmax=[0,10]
go
```

Can also run aoflagger at this stage to flag data in the multi-channel data set. Do not flag too deep, as there is still source structure in there.

---

24. One should average channels together to reduce the data volume before doing imaging and self-calibration. Before averaging, one should flag the data at the highest channel resolution so that narrow-band RFI doesn't affect the averaged channels. Easiest way is to run a clip on the data, using a threshold. The threshold could be either theoretical or from the

actual data. It's a good idea to try to compute what the theoretical threshold should be. Also should work out how many channels one can average (i.e. the coarsest acceptable frequency resolution, for the region that will be imaged). Typical acceptable channel widths are 0.25 MHz (Band-3), 0.5 MHz (Band-4), and 1 MHz (Band-5).

We'll choose to average 32 channels together, to reach a resolution of ~130 kHz.

```
tget split
vis='0311_full.ms'
outputvis='0311_full.ch0'
field='0'
datacolumn='data'
width=32                    # Averages 32 channels together.
go
```

---

25. Plot the new data to check the range for clipping bad data. The following would clip at 10 Jy.

```
tget flagdata
vis='0311_full.ch0'
mode='clip'
clipminmax=[0,10.0]
go
```

Can also run aoflagger on the channel-averaged data set, but do not flag deep at this stage.

---

26. Use plotms to plot the data versus uvwave, to check for holes and decide on the UV taper.

---

28. Now to the imaging, with tclean. Detailed information is provided next to each parameter.

```
inp tclean
vis='0311_full.ch0'
imagename='0311.im.1'
datacolumn='data'           # This is for the first imaging; change to 'corrected' after self-cal.
```

```
cell='1.5 arcsec'          # Choose such that there are ~4 cells across the beam FWHM.
imsize=[3600]              # Image a region of 3600 cells; image out to at least the primary
                               beam null.
spw='0:1~14'               # Exclude edge channels.
uvtaper=['35 klambda']     # Taper at 35 klambda, to smoothen the edges of the UV
                               distribution.
uvrange ='> 1.5 km'        # Exclude short baselines.
specmode='mfs'             # Multi-frequency synthesis.
gridder='wproject'         # 3-D imaging with w-projection.
wprojplanes=-1             # Allow tclean to decide on the number of w-planes.
pblimit=-0.1               # Image the entire region (do NOT put a primary beam cut).
weighting='briggs'         # Use robust weighting.
robust=-0.5                # Weighting slightly towards uniform; will give a more pointy
                               beam.
niter=100000000            # Set to a large number.
threshold='1 mJy'          # Set this to a highish value, ~ 1 mJy, >~ 10 sigma, in the first
                               round, where sigma is your theoretical noise; this would be
                               lowered after each round of self-calibration.
interactive=True           # Interactively clean.
cycleniter=500             # This should be set for interactive Cleaning.
savemodel='modelcolumn' # Save the Clean component model in the model column.
mask='0311.mask'           # Save the boxes that you'll be Cleaning on, in the mask file.
go
```

---

29. Inspect the image using the casa viewer:

    viewer()

    Use "Statistics" to measure the RMS noise from off-source regions and the peak flux density in the field.

---

30. The Clean model will contain both positive and negative components. Only the positive ones are "real" sky components. In order to filter out the negative ones, use the task "imthreshold" (by Aditya Chowdhury).

```
inp imthreshold
imname='0311.im.1'
threshold=0.0
boxsize=50
go
```

This will create a model image called 0311.im.1_noneg.model, which contains no negatives.

Outside CASA:

```
cp -R 0311.im.1.mask 0311.mask
```

---

31. Next, add the model without negatives back into the model column. For this, use tclean again:

```
tget tclean
savemodel='modelcolumn'
startmodel='0311.im.1_noneg.model'
niter=0
imagename='tst.im.1'
go
```

---

32. Next is the first round of self-calibration, with gaincalR. One would do 3-4 rounds of phase-only self-calibration and imaging, followed by one round of amplitude and phase self-calibration. This would be done with "gaincalR" not "gaincal".

```
tget gaincalR
vis='0311_full.ch0'
calmode='p'
caltable='0311.pcal.1'
refant=1
solint='1 min'                  # 1-2 minutes would be fine here.
```

```
uvrange='> 1.5 km'
spw='0:1~14'                # Exclude edge channels and line channels; same as tclean)
go
```

---

31. Next, run applycal to apply the calibration to the data set:

```
tget applycal
vis='0311_full.ch0'
gaintable='0311.pcal.1'
interp=['linear']
spw=''
go
```

---

33. Back to imaging with tclean, but now of the calibrated data: You would have to keep adding boxes at each tclean step, to cover new sources that are now visible in the improved image.

```
tget tclean
data='corrected'
startmodel=''
imagename='0201.im.2'
niter=10000000
threshold='1.0 mJy'         # Could lower this to around 5-sigma, at this stage.
go
```

---

34. Use the viewer to view the new image:

```
viewer()
```

Measure the RMS noise from off-source regions and the peak flux density in the field.

The RMS noise should be lower, and the peak flux density higher, than in the first image.

---

35. Repeat the above phase-only self-calibration and imaging, until phase self-calibration converges. Lower the threshold value to ~3-sigma, as measured on the image.

Once phase-only self-calibration has converged (no significant change in the RMS noise, the peak flux density in the field, or the total cleaned flux density), then move to amplitude-and-phase self-calibration. After the last round of imthreshold and tclean, to put the model back into the MS:

```
tget gaincalR
vis='0311_full.ch0'
calmode='ap'                # This does amplitude-and-phase self-calibration.
caltable='0311.apcal.1'
refant=1
solint='1 min'              # 1-2 minutes would be fine here.
uvrange='> 1.5 km'
spw='0:1~14'                # Exclude edge channels and line channels; same as tclean)
solnorm=True                # Normalize the gains. This is important!
go
```

---

36. Apply the gains with applycal, and then run tclean again. Normally, the image improves significantly with amplitude-and-phase self-calibration.

---

37. Again run imthreshold to remove negatives, and run tclean to put the model without negatives back into the model column. Next is to subtract the continuum model from the calibrated visibilities. This is done with "uvsub"

```
inp uvsub
vis='0311_full.ch0'
go
```

The calibrated data column now contains the residual visibilities, after continuum subtraction.

---

38. Check how the residual visibilities look in plotms; you will probably see a bunch of outliers, and bad data. Use aoflagger and/or flagdata to flag all bad data.

---

39. After flagging, run gaincalR again, on the flagged data set, to produce new estimates of the gains. Apply these gains and make a new image (with additional boxing, if needed). Again run imthreshold, tclean to put back the model without negatives, and uvsub to subtract the continuum. Finally, check the residuals with plotms. If the residuals look fine, your continuum imaging and self-calibration is done. If there's still muck in the residuals, run flagdata and/or aoflagger again to clean things further.

---

40. Now run applycal on the multi-channel data set to apply the final gains to the spectral-line data:

```
tget applycal
vis='0311_full.ms'
gaintable='0311.apcal.final'          # The final gain table from amplitude-and-phase
                                              self-calibration.
go
```

---

41. Run the task "repeatflag" (by Aditya Chowdhury) to transfer flags from the .ch0 file to the .ms file:

```
inp repeatflag
visfrom='0311_full.ch0'
visto='0311_full.ms'
go
```

---

42. Next, run tclean on the multi-channel data set to add the model to the model column.

```
tget tclean
vis='0311_full.ms'
spw=''
savemodel='modelcolumn'
startmodel='0311.im.final.model'   # The final continuum model. Note that this is WITH the
                                     negatives.
niter=0
imagename='tst.im.final'
go
```

---

43. Next, run uvsub to subtract the continuum from the multi-channel data set:

```
inp uvsub
vis='0311_full.ms'
go
```

---

44. Next, run aoflagger and/or flagdata to do the final flagging on the multi-channel data set.

---

45. Finally, make the spectral cube. Here, one can choose to specify a channel width or just let CASA do it for you (as done below):

```
tget tclean
vis='0311_full.ms'
imagename='0311.cube.1'
datacolumn='corrected'
cell='1.5 arcsec'          # Choose such that there are ~4 cells across the beam FWHM.
imsize=[128]               # Imaging a smaller region, since we're looking for
                             absorption against the central source.
spw=' '
```

```
uvtaper=['35 klambda']      # Taper at 35 klambda, to smoothen the edges of the UV
                                distribution.
uvrange ='> 1.5 km'         # Exclude short baselines.
specmode='cube'
outframe='bary'             # Apply the doppler correction to the barycentric frame.
restfreq='FILL MHz'         # This is the expected line rest frequency, after applying
                                any redshift corrections. For 0311+430, z=2.289.
                                restfreq='431.86553663727576 MHz'

gridder='wproject'
wprojplanes=-1
pblimit=-0.1
weighting='briggs'
robust=1                    # This would give better sensitivity than robust=-0.5,
                                but still downweight the shorter baselines.
niter=0                     # No cleaning, as there's no continuum and the line is
                                against a point source.
go
```

You now have a spectral cube at some velocity resolution.

---

46. Inspect the cube with the viewer:

Use "Spectral Tools" to obtain a spectrum at the location of the target source (centre of the field). To get the exact location at which the spectrum should be taken, check the location of peak flux density of the source in the continuum image.

Write the spectrum to an ASCII file (using the Spectral Tools button at the top left).

Analyse the ASCII spectrum for your science. Enjoy.