ALMA Data Processing, calibration and imaging

E. Villard
ALMA/JAO

Outline

- Short introduction to CASA
- Main steps of data reduction
- Script generator (Cycle 0)
- Pipeline (future Cycles)

Short introduction to CASA

Python and tasks

- CASA is built on Python environment
  - Scripts can be run with execfile()
  - System commands can be run with a !
- The task is the high-level interface to CASA
  - A task is a Python script, with inputs and outputs
  - *tasklist*, to obtain a list of all tasks
  - *help taskname*, to obtain help on a task
  - *inp taskname*, to list the input parameters of a task
  - *tget taskname*, to load the input parameters of a task at the previous successful execution
  - *go* to run active task

The three data columns

- DATA, CORRECTED, MODEL
  - Same dimensions
  - DATA contains the raw visibilities
  - CORRECTED contains the calibrated visibilities
  - MODEL contains the model visibilities

applycal

- CASA is never modifying the raw visibilities (the DATA column)
- applycal takes the DATA column, the calibration table, and puts the result in the CORRECTED column.
- This strategy means that one cannot apply calibration tables subsequently. All calibration tables must be applied in one go.

gaincal and setjy

- gaincal is used to calibrate out the time variations of phase and amplitude.
- In case of a resolved calibrator (e.g. Titan for flux calibration), we don't want to calibrate out phase/ amplitude variations coming from the extension of the object.
- The strategy is to have model visibilities in the MODEL column, and gaincal divides DATA by MODEL. (In other words: gaincal assumes a 1Jy point source.)
- setjy is used to fill in the MODEL column.
  - It has a model for most planets and moons.
  - It can take an image.

Main steps of data reduction

Main steps of data reduction

- Conversion ASDM to MS
- Tsys and WVR corrections, applycal
- Split of channelized spectral windows (not required)
- A priori flagging (autocorr, shadow, pointing and atm cal scans)
- setjy, if using any resolved calibrators
- Initial phase solution, bandpass
- Complete phase solution
- Complete amplitude solution
- Flux scale
- Clean (imaging)

Script generator (Cycle 0)

Motivations

- Human pipeline for Cycle 0 for ALMA delivery.
- Filling most parameters of the tasks is unambiguous and saves time.
- Started thinking about high-level code doing deeper analysis (heuristics) above that needed for delivery to the PI.
- Script generator: all intelligence in the code for selection and combination of parameters. Only CASA commands in final script.
- Final script can be tweaked by data reducer, and delivered to PI.

Main steps

- Automatic selection of reference antenna
- Fixplanets (adding in ss object position to ms)
- A priori flagging
- WVR cal table + smooth to sampling time
- Tsys cal table + interpolation if FDM mode
- Antenna position correction cal table
- Applycal + split to obtain science spws
- Clearing the pointing table
- Additional flagging as needed
- Setjy for ss object or quasar
- Bandpass cal table + smoothing in frequency if FDM mode
- Gain cal tables
- Applycal
- Flux bootstrapping if many datasets
- Imaging

Automatic flagging

- The usual: autocorr, shadow
- Calibration scans: pointing, atmcal, sideband ratio
- Online flags (Flag.xml)
- Edge channels: TDM mode, percentage of channels on both sides
- CO lines for Neptune and Titan

Antenna position correction

- Script producing a complete gencal command, using two files: “database” of baseline solutions, list of antenna moves
- Current limitation: Antenna table in ms contains only pad vector, no antenna vector. Solution is to transfer Antenna and Station tables from SDM.
- Per antenna: searches for latest baseline run after move to current pad. If found, compares solution and current positions, and uses solution only if difference is more than 3σ.
- Algorithm is always looking for latest solution. Possibility to use post-obs solution by giving date in the future.

Bandpass calibration

- Usual: ‘ap’ int + bandpass inf
- Automatic detection of bandpass calibrator. Use of phase calibrator if no bandpass calibrator.
- Automatic smoothing in time if FDM mode (no automatic switch to BPOLY if still noisy after smoothing, but plans to do it easily)

Gain calibration

- Usual: ‘p’ int + ‘a’ inf for amp cal and ‘p’ inf for phase cal
- Automatic detection of phase cal for each science target
- Main problem: when SSO is highly resolved. Automatic detection. Workaround:
  - Gaincal on subset of close antennas -> flux of phase cal
  - Setjy on phase cal with above flux
  - Gaincal on all antennas

Flux bootstrapping

- Two difficulties: some datasets have no SSO + necessity to equalize flux across datasets.
- Solution: keep flux calibration for the very end. If no SSO, then no setjy.
- At the very end: generate text file with average flux values, using all flux values available. Run setjy on each dataset, gaincal and applycal.
- Text file can be checked/edited by data reducer before running setjy.

Imaging

- Still relies on data reducer. But developed two tools to help:
  - Clean command generator, with standard parameters already filled (e.g. cell, imsize, phasecenter, field, imagename)
  - Line finder: average of amp at spectral resolution from proposal and over uv range using LAS from proposal, then histogram and detection using criteria of N adjacent channels (default = 3). Plots are very useful. Detection algorithm is moderately successful (yet very simple).
- Both tools interact. If specified, clean command generator will run line finder, and use channel ranges for spw parameter.

Pipeline (future Cycles)

There are 3 pipelines

- Telescope calibration pipeline
  - online, to provide calibration results
- Quicklook display pipeline
  - online, to provide quick overview to AoD and operators
- Science pipeline
  - offline

The science pipeline

- ALMA must be available to all of the astronomical community, i.e. also interferometry non-experts
- Hence an automatic pipeline is required to provide science quality data products
- The automatic processing will be performed for all standard reduction modes
- The pipeline scripts will be made available for further offline processing by the PIs and the ARCs

Infrastructure

- The ALMA Pipeline is based on CASA using data in ASDM / MS format
- The processing is (meta-) data driven
- CASA team provides engines as tools and tasks
- Pipeline team uses these engines to create heuristics tasks for Quicklook and Science Pipelines
- Pipeline will run on HPC cluster environment
- Parallelization via parallel tasks and Open MP / Open MPI

Expert knowledge

- Began with data reduction scripts from OFFLINE (i.e. CASA) subsystem user tests
- Talked a lot with experts to learn about their approaches to data reduction
- Condensed this expertise into algorithms
- Tested and refined them using datasets of existing observatories (PdB, VLA, SMA, IRAM 30m, APEX, HHT, Effelsberg, GBT). Since 2009 also with ALMA SD and IF data.

Pipeline heuristics

- The Pipeline Heuristics tries to capture the - sometimes diffuse - expert knowledge and encode it as data reduction recipes
- There is a recipe per reduction mode
- By now there are heuristics recipes for
  - Single field interferometry
  - Pointed Mosaics
  - Single dish data
- IF/SD combination recipe is under development

Interferometry heuristics

- The recipe is laid out in a series of 'stages'.
- The idea is that as the reduction of a dataset moves through the stages the bad data are gradually removed and the best methods for calibrating the data are found.
- In the final stages the cleaned images and other data products are calculated.

Single dish heuristics

- The SD recipe implements stages to
  - Calculate calibrated spectra for different observing switch modes (on-off, nutator, frequency switching)
  - Automatically identify lines
  - Fit spectral baselines
  - Flag bad data
  - Image the data
  - Line finding, baselining and flagging are iterated in a loop

Data products

- Final data products stored in the archive:
  - Raw data
  - Flagging tables
  - Calibration tables
  - Images / Data cubes (FITS)
  - Quality Assessment measures
  - CASA logs
  - Web results pages
  - Heuristics scripts

Commissioning of the pipeline

- We already began adjusting the heuristics to available ALMA data
- The pipeline will be commissioned during ALMA Early Science
- The automatic results will be compared against manual reductions and parameters and algorithms will be tuned

Quality assessment

Motivations

- Assess whether the observations fulfill the goals of the proposer (final products).
- QA = sets of procedures and requirements set by the observatory to guarantee to reach the goals of the proposer.
- Four levels of QA:
  - QA0: quality of the atmosphere and the array on short term
  - QA1: quality of the array on long term
  - QA2: quality of the calibration
  - QA3: feedback from the proposer

Examples

- Antenna not on source -> QA0
- Issue with front end of an antenna -> QA0
- Large phase rms due to high PWV -> QA0
- Antenna position error -> QA1
- Degraded antenna surface -> QA1
- Not enough time on bandpass -> QA2
- Residual atmospheric line after Tsys -> QA2
- Final rms larger than requested -> QA2
- Line does not appear at correct frequency -> QA3

Long-term plan vs. Cycle 0

- Long-term plan:
  - QA0 -> most complicated, objectives are:
    - Has the dataset *enough* good data?
    - What are the bad data? (file with flags to apply)
  - QA1 -> observatory calibrations, verifications
  - QA2 -> rather simple
- Cycle 0:
  - QA0+ -> generated script runs successfully?
  - QA2 -> hybrid of proper QA0 and QA2

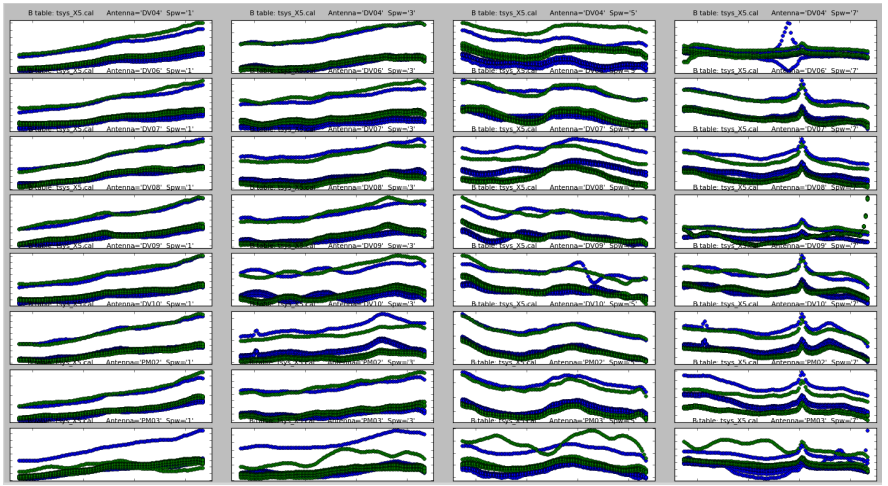
Example

NGC3256 (SV target), band 3

March 9-17, 2011

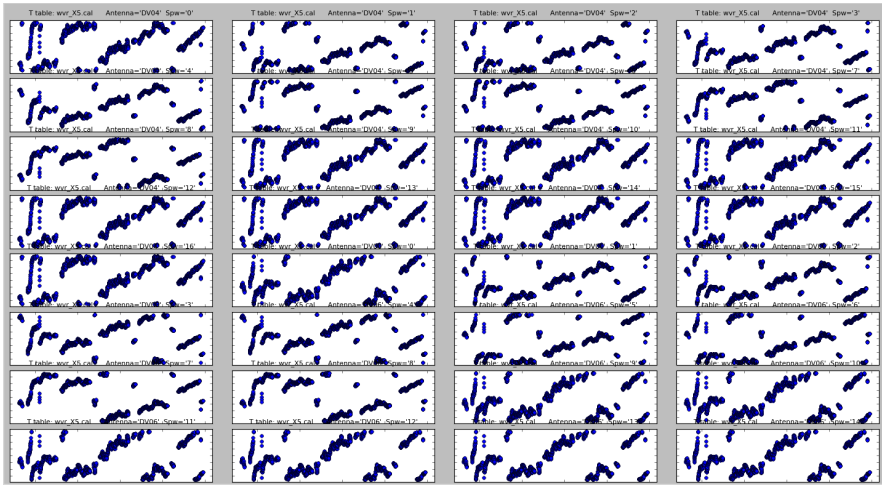
Tsys solution

```
taskname = 'gencal’
default(taskname)
vis = 'uid___A002_X1d5a20_X5.ms’
caltype = 'tsys’
caltable = 'tsys_X5.cal’
gencal()
```



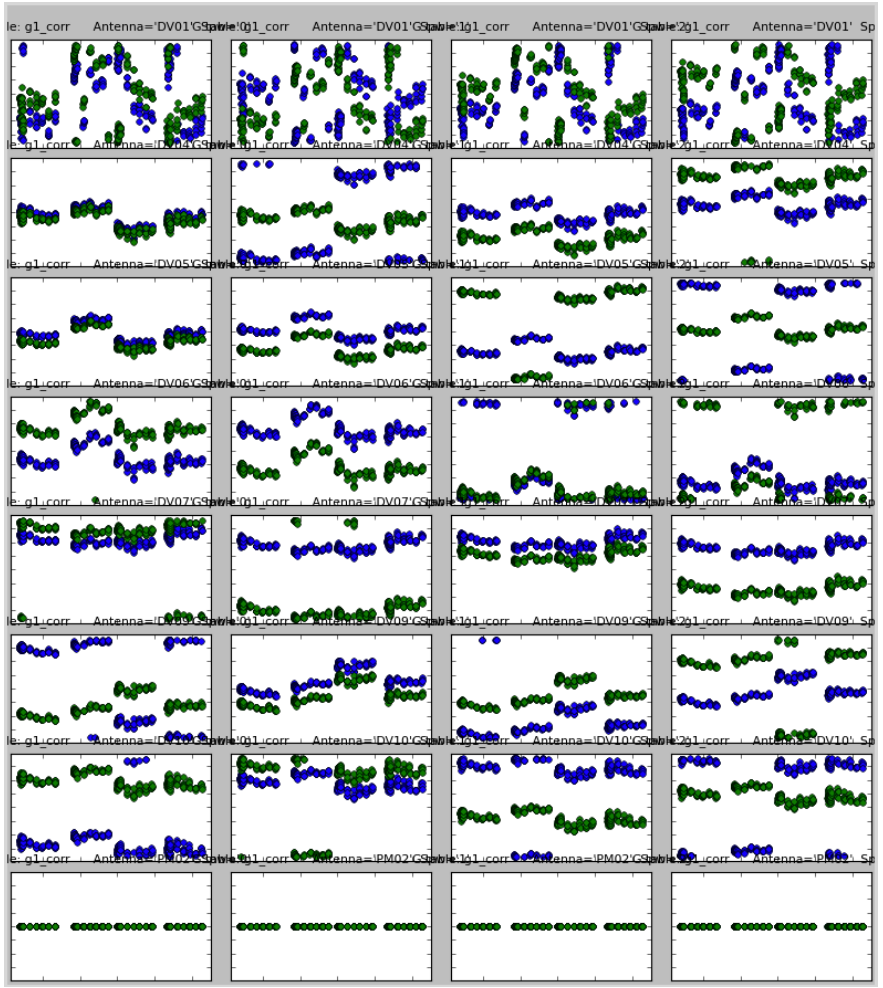
WVR solution

```
os.system('wvrgcal --ms
    uid___A002_X1d5a20_X5.ms --
    output wvr_X5.cal --toffset -1 --
    segfield')
```



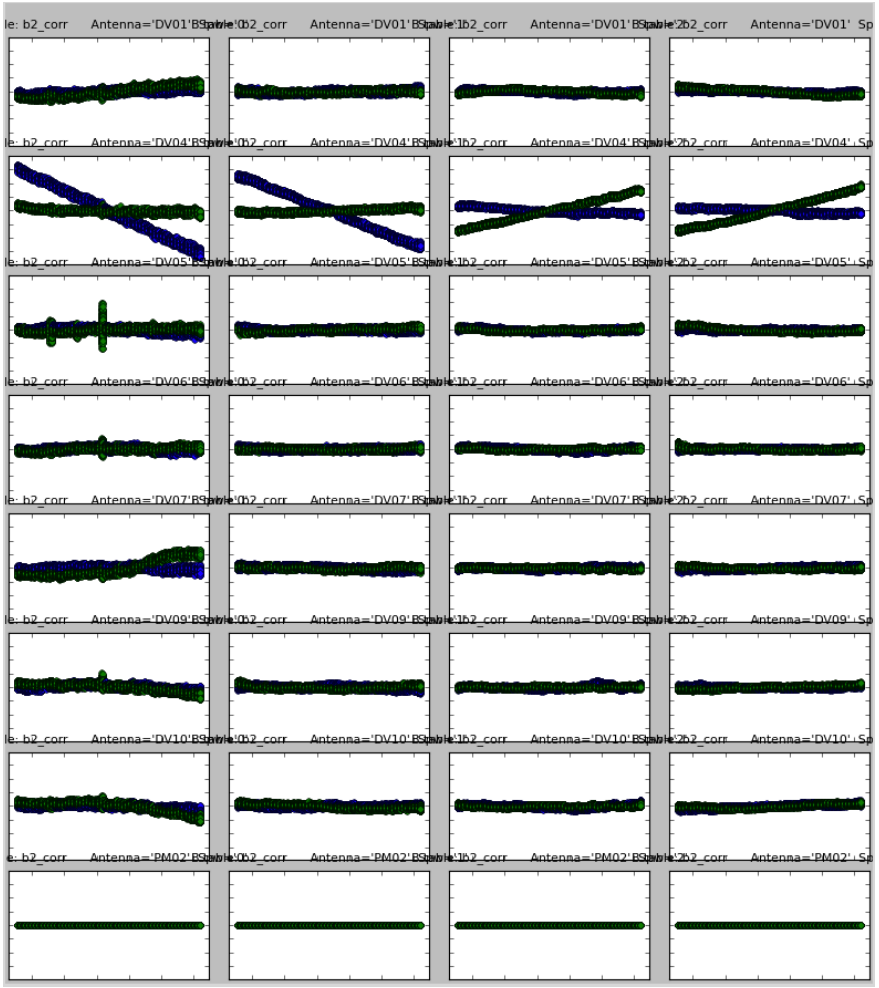
Bandpass solution (1)

```
taskname = 'gaincal'
default(taskname)
vis = 'ngc3256.ms'
caltable = 'g1_corr'
spw = '*:50~80'
field = '0'
solint = 'int'
refant = 'PM02'
calmode = 'p'
gaincal()
```



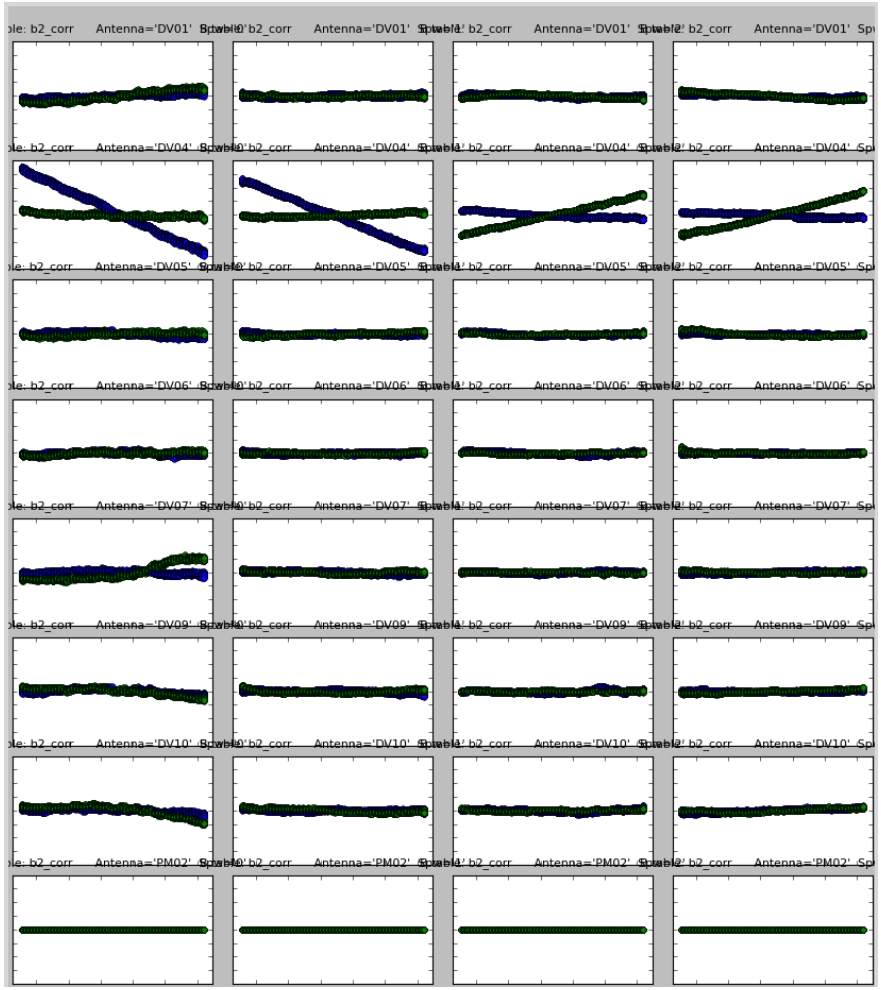
Bandpass solution (2)

```
taskname = 'bandpass'
default(taskname)
vis = 'ngc3256.ms'
caltable = 'b2_corr'
field = '0'
refant = 'PM02'
solint = 'int'
solnorm = T
gaintable = 'g1_corr'
bandpass()
```



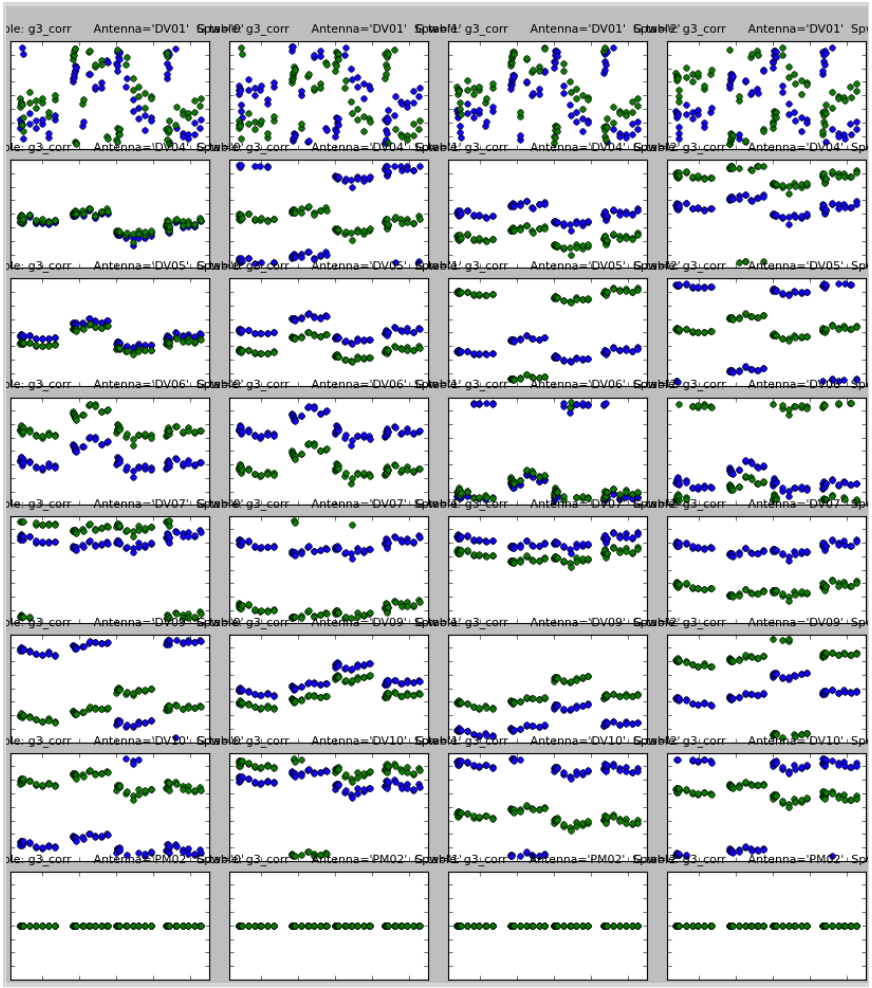
Bandpass solution (3)

```
taskname = 'flagdata’
default(taskname)
vis = 'ngc3256.ms’
flagbackup = F
spw = ['0:31~32;47~48;63~64']
field = '0’
correlation = 'YY’
flagdata()
```



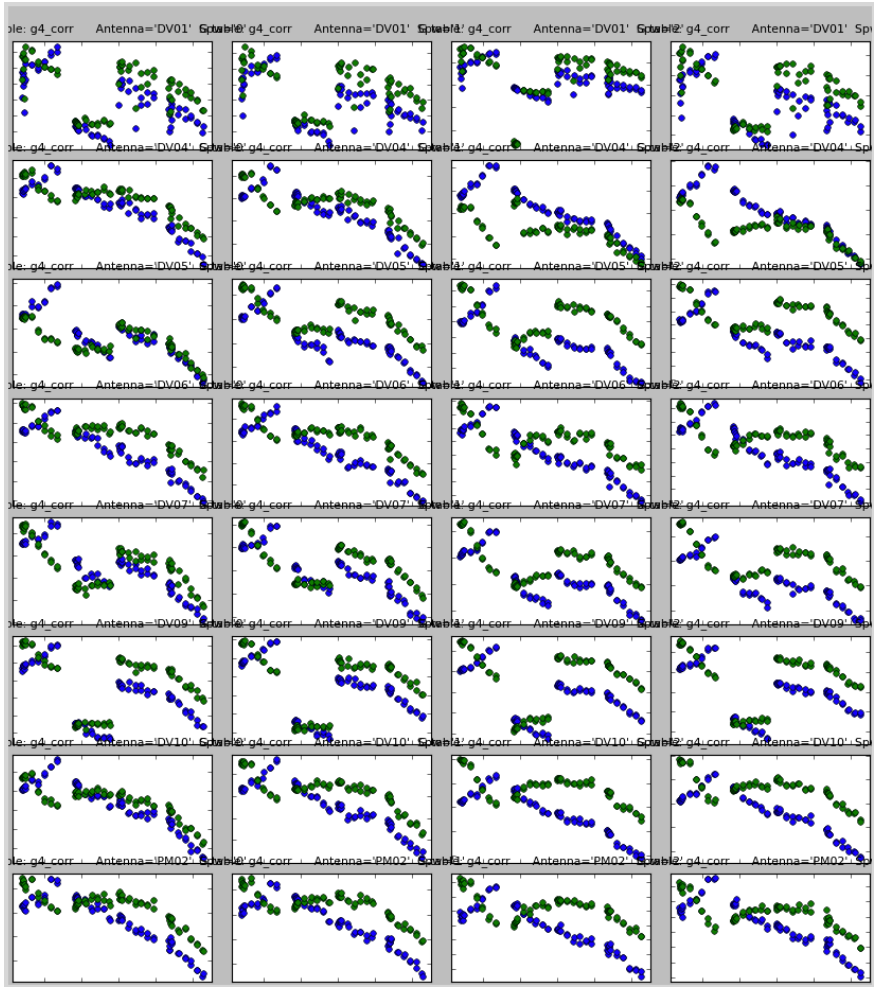
Complete phase solution

```
taskname = 'gaincal'
default(taskname)
vis = 'ngc3256.ms'
caltable = 'g3_corr'
field = '0'
solint = '30s'
refant = 'PM02'
calmode = 'p'
gaintable = 'b2_corr'
gaincal()
```

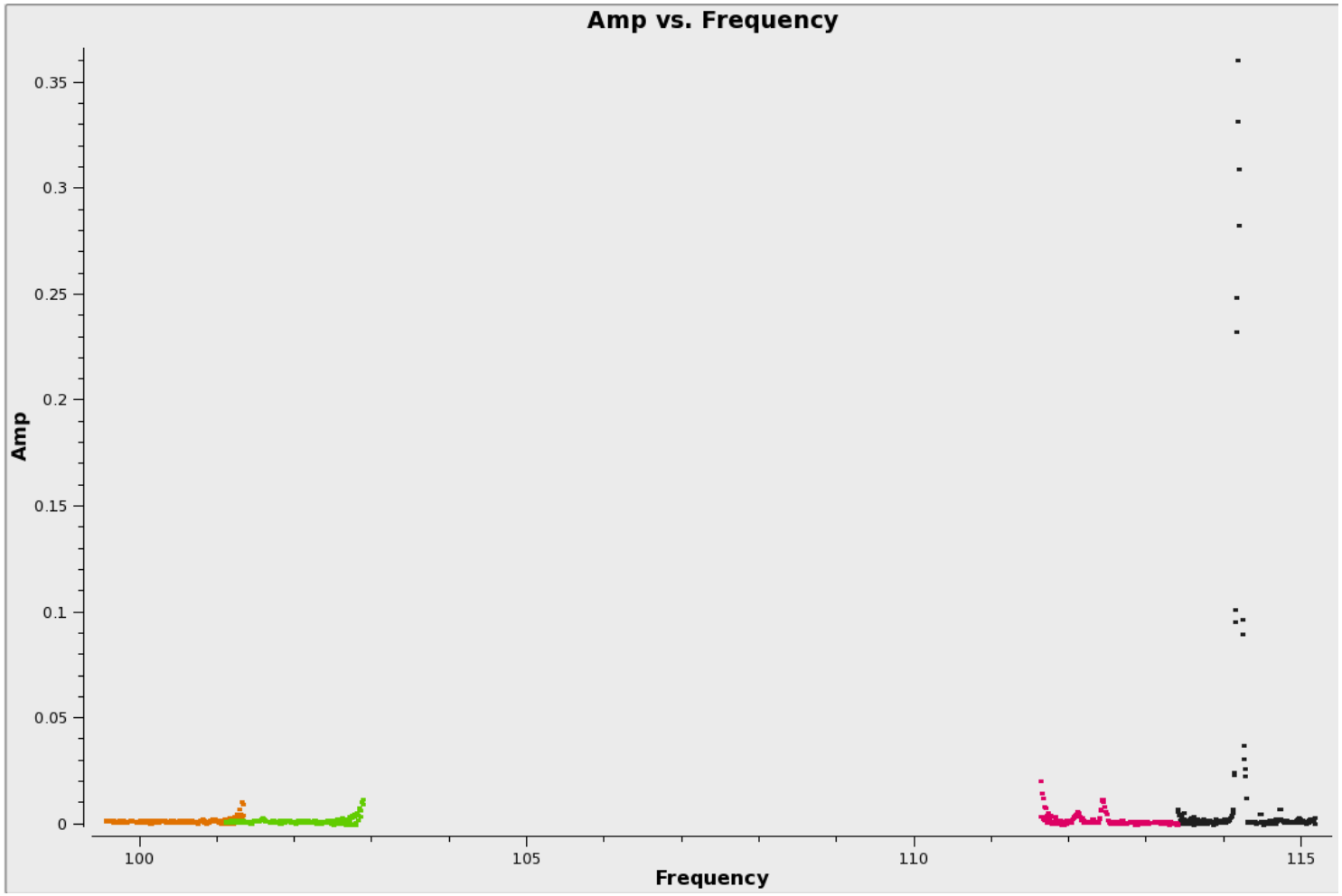


Complete amplitude solution

```
taskname = 'gaincal'
default(taskname)
vis = 'ngc3256.ms'
caltable = 'g4_corr'
field = '0'
solint = '30s'
refant = 'PM02'
calmode = 'ap'
gaintable = ['b2_corr', 'g3_corr']
gaincal()
```

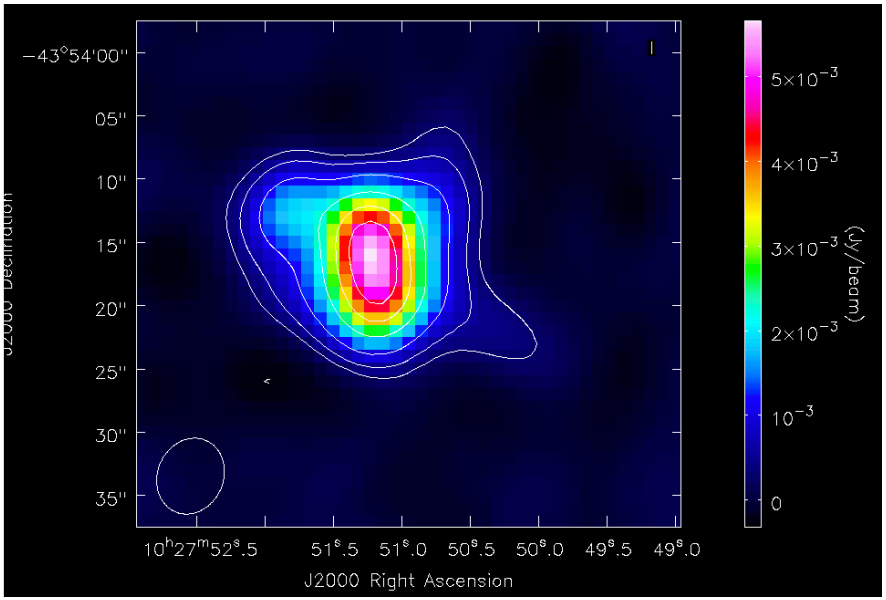


Amp vs freq, after full calibration



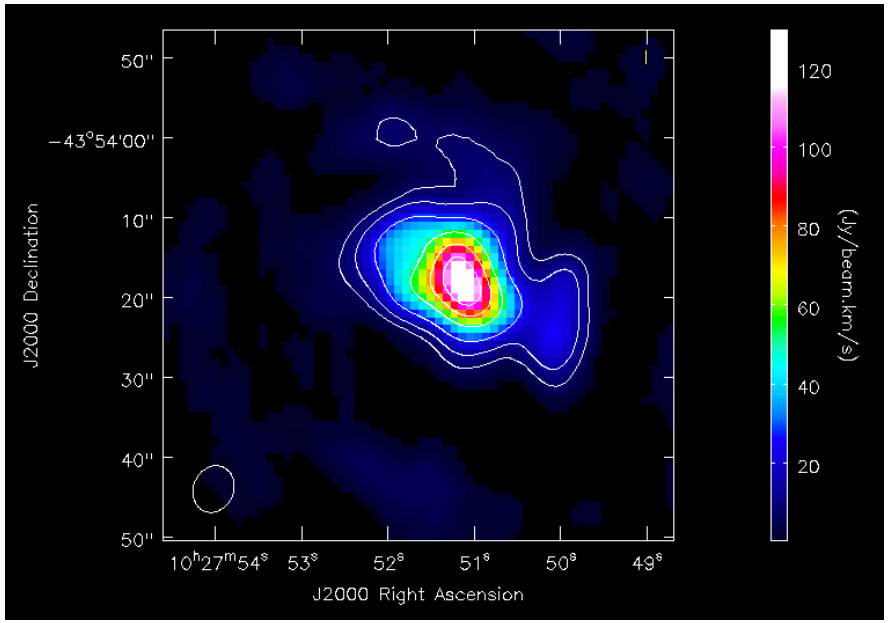
Continuum map (with one round of selfcal)

```
taskname = 'clean'
default(taskname)
vis = 'ngc3256_cont.ms'
imagename = 'ngc3256_cont'
field = '0'
cell = '1arcsec'
imsize = 512
weighting = 'briggs'
interactive = T
niter = 1000
clean()

taskname = 'gaincal'
default(taskname)
vis = 'ngc3256_cont.ms'
caltable = 'g5_corr'
field = '0'
solint = '600s'
refant = 'PM02'
calmode = 'p'
gaincal()
```



CO (1-0) moment 0 map

```
taskname = 'uvcontsub'
default(taskname)
vis = 'ngc3256_specline.ms'
field = '0'
fitspw =
      '0:16~55;71~127,1:16~55;71~127,2:16~127,
      3:16~127'
solint = 'int'
fitorder = 1
fitmode = 'subtract'
uvcontsub()

taskname = 'immoments'
default(taskname)
imagename = 'ngc3256_specline.image'
moments = [0]
includepix = [0.01, 1]
outfile = 'ngc3256_specline_moment0'
immoments()
```



CO (1-0) moment 1 map

```
taskname = 'immoments'
default(taskname)
imagename =
    'ngc3256_specline.image'
moments = [1]
includepix = [0.04, 10]
outfile =
    'ngc3256_specline_moment1'
immoments()
```