



ALMA Manual Calibration

Dominic Ludovici | 21st Synthesis Imaging Workshop | May 28th, 2026



National Radio
Astronomy
Observatory

Learning Objectives

At the end of this tutorial, participants should be able to...

1. Summarize the basic steps for calibrating ALMA data.
2. Use CASA to calibrate an ALMA observation.
3. Interpret graphs of ALMA data and calibration solutions



Tutorial Guide

- I will use Active Learning techniques to keep you engaged.
 - We will often stop for discussion and questions.
 - Please sit near someone else. Help each other and check your work against your neighbors.
- Your understanding is important to us!
 - I will be walking around the room to help while you work.
 - We have many others here to help as well.
- Ask us questions! Raise your hand and someone will come to help!

Your CASA tutorial team



Dominic Ludovici
NRAO NAASC



Natalie Butterfield
NRAO NAASC



Tristan Ashton
NRAO NAASC



Michael Sanchez
NRAO NAASC



Ed Starr
NRAO NAASC

Your CASA tutorial team



Maria Galloway-Sprietsma
ALMA Ambassador
University of Florida



Carla Cornil-Baiotto
ALMA Ambassador
Instituto de Física y
Astronomía



Jess Speedie
Former ALMA Ambassador
MIT

Overview

1. Data and Project Description
2. Getting Started in CASA
3. Initial Calibration and Data Preparation
4. Data Inspection
5. Calibration

Data Description

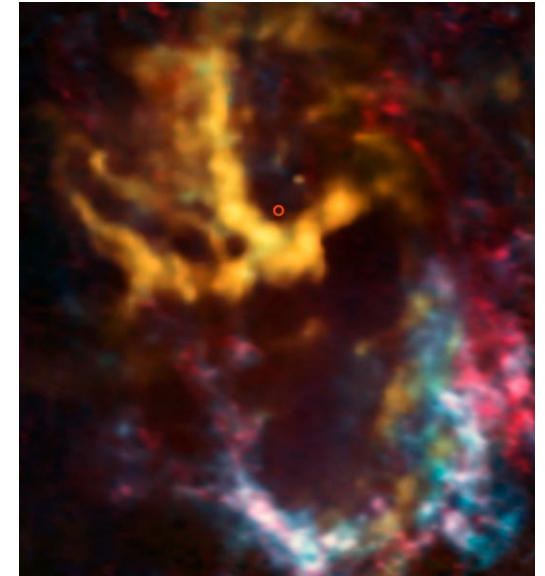
Today's data is from the ALMA Atacama Compact Array (ACA) filler projects observed during cycle 6.



Data Description

We will examine the heart of our own galaxy: SGR A*!

- Sgr A* is a variable source of radiation from the Radio up to gamma rays.
- During the observing campaign, ALMA cooperated with Spitzer, Chandra, and NuStar to observe our central black hole.
- The data contain both continuum and spectral line observations, allowing us to examine the emission from Sgr A* as well as the Circumnuclear Disk!



Sgr A* as seen by ALMA in multiple bands.

Data Description

We will image the continuum emission and CO line emission observed at Band 6.

- [Link to paper:](#)
- <https://iopscience.iop.org/article/10.3847/1538-4357/ac2d2c/pdf> and
- <https://iopscience.iop.org/article/10.3847/1538-4357/ac6104/pdf>

Getting Started

Navigating to Your Data

Starting CASA

Task Help

Running Tasks

In your directory there should be two sub-directories labeled /Calibration and /Imaging.

In **ALMA_manual_calibration_tutorial.tar** you should have:

- **uid___A002_Xdf0444_X10bd** (the raw ASDM data file containing uncalibrated data)
- **calibration.py** (the script we will work through together to calibrate the data)
- **analysis_scripts**

Getting Started

Navigating to Your Data

Starting CASA

Task Help

Running Tasks

CASA (Common Astronomy Software Applications)

- CASA is the offline data reduction package for ALMA and the VLA (data from other telescopes usually work too, but not primary goal of CASA)
- Code is C++ (fast) bound to Python which provides ease of use and scripting
- Provides data import/export, inspection, editing, calibration, imaging, viewing, and analysis
- Also supports single dish data reduction
- We have a lot of documentation (CASA docs), reduction tutorials (CASA guides), and a helpdesk.

Getting Started

Navigating to Your Data

Starting CASA

Task Help

Running Tasks

Start CASA.

Lets make sure everyone can start CASA!

If you put CASA in your path, you should be able to start it by typing “**casa**” on the command line.

If not, give the full path:
</path/to/casa/install>/bin/casa

Getting Started

Navigating to Your Data

Starting CASA

Task Help

Running Tasks

The image shows two overlapping windows. The top window is a log viewer titled 'Log Messages (:/Users/masanche/Doc...)' with a search bar and a filter set to 'Time'. It displays a list of log entries with columns for Time, Priority, Origin, and Message. The bottom window is an IPython terminal titled 'Imaging -- IPython: SDP81/Imaging -- casalogger - Python -m casashell -- 122x24'. It shows the output of the CASA startup process, including version information and initialization steps.

Time	Priority	Origin	Message
2023-04-28 14:46:20	INFO	::casa	CASA Version PIPELINE 6.4.1.12
2023-04-28 14:46:20	INFO	::casa	Found an existing telemetry logfile: /Users/masanche/.casa/casastats-6.4.1.12-126f812e3161ae1b7-20230426-145957.1
2023-04-28 14:46:20	INFO	::casa	Telemetry log file: /Users/masanche/.casa/casastats-6.4.1.12-126f812e3161ae1b7-20230426-145957.log
2023-04-28 14:46:20	INFO	::casa	Checking telemetry submission interval
2023-04-28 14:46:20	INFO	::casa	Telemetry submit interval not reached. Not submitting data.
2023-04-28 14:46:20	INFO	::casa	Next telemetry data submission in: 4 days, 2:34:56.944640
2023-04-28 14:46:21	INFO	::casa	imported analysisUtils version \$Id: analysisUtils.py,v 2.15 2023/03/01 18:52:10 thunter Exp \$ from /Users/masanch
2023-04-28 14:46:21	INFO	::casa	optional configuration file config.py not found, continuing CASA startup without it
2023-04-28 14:46:21	INFO	::casa	Using user-supplied startup.py at ~/.casa/startup.py
2023-04-28 14:46:21	INFO	::casa	Checking Measures tables in data repository sub-directory /Applications/CASA-ALMA-v6.4.app/Contents/Frameworks/Py
2023-04-28 14:46:21	INFO	::casa	IERSeop2000 (version date, last date in table (UTC)): 2022/06/23/15:00, 2022/05/24/00:00:00
2023-04-28 14:46:21	INFO	::casa	IERSeop97 (version date, last date in table (UTC)): 2022/06/23/15:00, 2022/05/24/00:00:00
2023-04-28 14:46:21	INFO	::casa	IERSpredict (version date, last date in table (UTC)): 2022/06/26/15:00, 2022/09/24/00:00:00
2023-04-28 14:46:21	INFO	::casa	TAI.UTC (version date, last date in table (UTC)): 2022/06/20/15:00, 2017/01/01/00:00:00

```
optional configuration file config.py not found, continuing CASA startup without it
Using user-supplied startup.py at ~/.casa/startup.py

IPython 7.15.0 -- An enhanced Interactive Python.

Using matplotlib backend: MacOSX
Telemetry initialized. Telemetry will send anonymized usage statistics to NRAO.
You can disable telemetry by adding the following line to the config.py file in your rcdir (e.g. ~/.casa/config.py):
telemetry_enabled = False
--> CrashReporter initialized.
casaVersion = 6.4.1.12
imported casatasks and casatools individually
Using astropy.io.fits instead of pyfits
CASA 6.4.1.12 -- Common Astronomy Software Applications [6.4.1.12]

CASA <1>:
```

Getting Started

Navigating to Your Data

Starting CASA

Task Help

Running Tasks

CASA runs within python scripts or through the interactive *IPython* (ipython.org) interface

- IPython Features:
 - shell access
 - auto-parenthesis (autocall)
 - Tab auto-completion
 - command history (arrow up and “hist [-n]”)
 - session logging
 - **casaTIME.log** – casa logger messages
 - numbered input/output
 - history/searching

Getting Started

Navigating to Your Data

Starting CASA

Task Help

Running Tasks

Tasks - high-level functionality

- function call or parameter handling interface
- these are what you should use in tutorials

Tools - complete functionality

- **tool.method()** calls
- they are internally used by tasks or can be used on their own sometimes shown in tutorial scripts and CASAGuides

Applications – some tasks/tools invoke standalone apps

- e.g. **casaviewer**, **mpicasa**

Shell commands can be run with a leading exclamation mark

- **!du -ls** or inside **os.system("shell command")**
- (some key shell commands like "ls" work without the exclamation mark
- We will use **os.system()** exclusively within this tutorial.)

Getting Started

Navigating to Your Data

Starting CASA

Task Help

Running Tasks

taskhelp

```
Calibration — IPython: SDP81/Calibration — casalogger · Python -m casashell — 110x50
CASA <1>: taskhelp
=====
CASA tasks
=====
> analysis
-----
imcollapse : Collapse image along one axis, aggregating pixel values along that axis.
imcontsub  : Estimates and subtracts continuum emission from an image cube
imdev      : Create an image that can represent the statistical deviations of the input image.
imfit      : Fit one or more elliptical Gaussian components on an image region(s)
imhead     : List, get and put image header parameters
imhistory  : Retrieve and modify image history
immath     : Perform math operations on images
immoments  : Compute moments from an image
impbcor    : Construct a primary beam corrected image from an image and a primary beam pattern.
impv       : Construct a position-velocity image by choosing two points in the direction plane.
imrebin    : Rebin an image by the specified integer factors
imreframe  : Change the frame in which the image reports its spectral values
imregrid   : regrid an image onto a template image
imsmooth   : Smooth an image or portion of an image
imstat     : Displays statistical information from an image or image region
imsubimage : Create a (sub)image from a region of the image
imtrans    : Reorder image axes
imval      : Get the data value(s) and/or mask value in an image.
listvis    : List measurement set visibilities.
rmfit      : Calculate rotation measure.
specfit    : Fit 1-dimensional gaussians and/or polynomial models to an image or image region
specflux   : Report spectral profile and calculate spectral flux over a user specified region
specsmooth : Smooth an image region in one dimension
spxfit     : Fit a 1-dimensional model(s) to an image(s) or region for determination of spectral index.
-----
> calibration
-----
accor      : Normalize visibilities based on auto-correlations
applycal   : Apply calibrations solutions(s) to data
bandpass   : Calculates a bandpass calibration solution
blcal      : Calculate a baseline-based calibration solution (gain or bandpass)
calstat    : Displays statistical information on a calibration table
clearcal   : Re-initializes the calibration for a visibility data set
delmod     : Deletes model representations in the MS
fixplanets : Changes FIELD and SOURCE table entries based on user-provided direction or POINTING table, optionally fixes the UVW coordinates
fluxscale  : Bootstrap the flux density scale from standard calibrators
fringeft   : Fringe fit delay and rates
ft         : Insert a source model as a visibility set
gaincal    : Determine temporal gains from calibrator observations
gencal     : Specify Calibration Values of Various Types
initweights : Initializes weight information in the MS
```

Getting Started

Navigating to Your Data

Starting CASA

Task Help

Running Tasks

Task Execution:

Write the full parameter set in a line:

- `taskname(arg1=val1, arg2=val2, ...)`
- e.g. `tclean(vis='input.ms',imagename='galaxy', robust=0.5, imsize=[200,200])`

unspecified parameters will be set to their default values (globals not used; i.e. not to previously set variables)

Useful in scripts, but also in 'pseudo-scripts':

- To keep a record, it is frequently a good idea to write down the full line as above in an editor, then cut and paste into CASA.
- When changes are needed, change in editor and cut and paste again. That is good practice to keep a record of the exact input.
- But note that the logger is also repeating the full task command

Getting Started

Navigating to Your Data

Starting CASA

Task Help

Running Tasks

Scripts can be run with the following:

```
execfile('<yourscript.py>')
```

Python 3 removed the `execfile` built-in function. CASA provides a convenience function that attempts to reproduce the behavior of the Python 2.7 built-in `execfile` function.

Getting Started

Navigating to Your Data

Starting CASA

Task Help

Running Tasks

CASA also has an asking interface, similar to AIPS, MIRIAD, etc.

- parameter manipulation commands
 - inp, default, saveinputs, tget, tput
- use parameters set as global Python variables
 - `<param> = <value>`
 - (e.g. `vis = 'ngc5921.demo.ms'`)
- execute
 - `<taskname>` or `go` (e.g. `tclean()`)
- return values (except when using “go”)
 - some tasks return Python dictionaries, assign a variable name to get them, e.g. `myval=imval()`
 - Very useful for scripting based on task outputs

Getting Started

Navigating to Your Data

Starting CASA

Task Help

Running Tasks

```
Calibration — IPython: SDP81/Calibration — casalogger · Python -m casashell — 110x50
[CASA <3>: inp tclean
# tclean -- Radio Interferometric Image Reconstruction
vis = '' # Name of input visibility file(s)
selectdata = True # Enable data selection parameters
  field = '' # field(s) to select
  spw = '' # spw(s)/channels to select
  timerange = '' # Range of time to select from data
  uvrange = '' # Select data within uvrange
  antenna = '' # Select data based on antenna/baseline
  scan = '' # Scan number range
  observation = '' # Observation ID range
  intent = '' # Scan Intent(s)
datacolumn = 'corrected' # Data column to image(data,corrected)
imagename = '' # Pre-name of output images
imsize = [100] # Number of pixels
cell = [] # Cell size
phasecenter = '' # Phase center of the image
stokes = 'I' # Stokes Planes to make
projection = 'SIN' # Coordinate projection
startmodel = '' # Name of starting model image
specmode = 'mfs' # Spectral definition mode (mfs,cube,cubedata, cubesource)
  reffreq = '' # Reference frequency
gridding = 'standard' # Gridding options (standard, wproject, widefield, mosaic,
  # awproject)
  vptable = '' # Name of Voltage Pattern table
  pblimit = 0.2 # PB gain level at which to cut off normalizations
deconvolver = 'hogbom' # Minor cycle algorithm
  # (hogbom,clark,multiscale,mtmfs,mem,clarkstokes)
restoration = True # Do restoration steps (or not)
  restoringbeam = [] # Restoring beam shape to use. Default is the PSF main lobe
  pbcor = False # Apply PB correction on the output restored image
outlierfile = '' # Name of outlier-field image definitions
weighting = 'natural' # Weighting scheme (natural,uniform,briggs,
  # briggsabs[experimental], briggsbw taper[experimental])
  uv taper = [] # uv-taper on outer baselines in uv-plane
niter = 0 # Maximum number of iterations
usemask = 'user' # Type of mask(s) for deconvolution: user, pb, or
  # auto-multithresh
  mask = '' # Mask (a list of image name(s) or region file(s) or region
  # string(s) )
  pbmask = 0.0 # primary beam mask
fastnoise = True # True: use the faster (old) noise calculation. False: use
  # the new improved noise calculations
restart = True # True : Re-use existing images. False : Increment imagename
savemodel = 'none' # Options to save model visibilities (none, virtual,
  # modelcolumn)
calcrs = True # Calculate initial residual image
calcpfs = True # Calculate PSF
  psfcutoff = 0.35 # All pixels in the main lobe of the PSF above psfcutoff are
  # used to fit a Gaussian beam (the Clean beam).
```

Getting Started

Navigating to Your Data

Starting CASA

Task Help

Running Tasks

Getting Help:

<https://casadocs.readthedocs.io/en/stable/>



Search docs

- Release Information
- Index
- API (tasks, tools, GUIs, etc.)
- Task List (shortcut)
- Using CASA
- CASA Fundamentals
- External Data
- Calibration & Visibilities
- Imaging & Analysis
- CARTA
- Pipeline
- Simulations
- Parallel Processing
- CASA Memos
- CASA Knowledgebase
- Community Examples
- FAQ
- Citing CASA
- Contact
- Change Log

Common Astronomy Software Applications

[View page source](#)

Common Astronomy Software Applications

CASA, the *Common Astronomy Software Applications*, is the primary data processing software for the Atacama Large Millimeter/submillimeter Array (ALMA) and Karl G. Jansky Very Large Array (VLA), and is often used also for other radio telescopes.

6.7.5 Release

CASA 6.7.5 can now be [downloaded](#) for general use. CASA 6.7.5 is available either as a downloadable tar-file, or through pip-wheel installation, which gives flexibility to integrate CASA into a customized Python environment.

Highlights:

- MacOS 26: support for MacOS 26 was added.
- pccor: new task for pulse delay calibration.
- ft: extended list of data selection parameters.
- fringefit: performance has been improved with speedup of FFT stage.
- sdcal: generates fully flagged calibration solution when OFF_SOURCE data are invalid.
- importasdm: added support for additional optional ASDM columns.

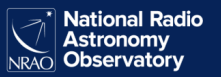
In addition, a large number of bugs were fixed.

CASA is being developed by an international consortium of scientists and software engineers based at the National Radio Astronomical Observatory (NRAO), the European Southern Observatory (ESO), the National Astronomical Observatory of Japan (NAOJ), and the Joint Institute for VLBI European Research Infrastructure Consortium (JIVE), under the guidance of NRAO.

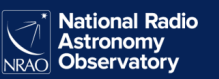
[Next](#)

© Copyright 2021, Associated Universities, Inc.

Built with [Sphinx](#) using a [theme](#) provided by [Read the Docs](#).



Initial Calibration



Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split

Downloading data from the ALMA archive will return raw data along with the scripts necessary for calibrating the data.

In this section, we will explain the steps taken to prepare the data for full calibration. This is an initial calibration that applies online corrections to the data.

- Import the raw data into a casa measurement set.
- Occasionally a dataset will require a fix to some of the metadata (i.e. the header). In this case, no table corrections are needed.
- Data that is known to be irrelevant to calibration or to be problematic (even without inspection of the data) is flagged. Examples: data taken when the telescope was not on source yet, when the system temperature load was too close to the beam, when the receivers were not yet tuned)
- Create 3 correction tables (WVR, Tsys, antenna positions) and apply them.
- The output of this initial calibration will be `uid___A002_Xdf0444_X10bd.ms.split`

Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split

When downloading data from the ALMA archive, the data arrive in .asdm.sdm format. These data must be imported before processing. You may need to remove the .asdm.sdm file extension as well. The import will take a few minutes.

From `calibration.py`:

```
importasdm('uid__A002_Xdf0444_X10bd',  
          asis='Antenna Station Receiver Source  
CalAtmosphere CalWVR CorrelatorMode SBSummary',  
          bdfFlags=True, lazy=False,  
          process_caldevice=False)
```

Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split

A dataset may require a fix to some of the metadata (i.e. the header). In this case, some coordinates in the metadata are adjusted.

This dataset does not require any of these corrections, and we are making efforts to fix data that required these fixes so that they are not necessary anymore. However, you may see something like the example below in some older scripts:

Unrelated Example:

```
es.fixForCSV2555('SDP81_B4_uncalibrated.ms')  
fixsyscaltimes(vis = 'SDP81_B4_uncalibrated.ms')
```

Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split

We proceed by flagging the data which will not use for the calibration: pointing measurements and tsys measurements, as well as autocorrelations (correlations within a given antenna). Note that the information on tsys is still contained in the measurement step, as a table (which we extract later).

From `data_prep.py`:

```
flagdata(vis = 'uid__A002_Xdf0444_X10bd.ms',  
         mode = 'manual',  
         spw = '4~11,16~23',  
         autocorr = True,  
         flagbackup = False)
```

```
flagdata(vis = 'uid__A002_Xdf0444_X10bd.ms',  
         mode = 'manual',  
         intent = '*POINTING*,*ATMOSPHERE*',  
         flagbackup = False)
```

```
flagcmd(vis = 'uid__A002_Xdf0444_X10bd.ms',  
        inmode = 'table',  
        useapplied = True,  
        action = 'plot',  
        plotfile = 'uid__A002_Xdf0444_X10bd.ms.flagcmd.png')
```

```
flagcmd(vis = 'uid__A002_Xdf0444_X10bd.ms',  
        inmode = 'table',  
        useapplied = True,  
        action = 'apply')
```

Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split

The Water Vapor Radiometers (WVR) provide corrections to the phase delay due to the atmosphere.

- Key to correcting short-timescale phase variations
- Time variable phase calibration

12m antennas have WVRs. However 7m datasets do not.

Do we need to apply WVR corrections to our data today?

High Frequency data from TW Hydra to illustrate what WVR does.

Data Preparation

Importing your ASDM

Table Corrections

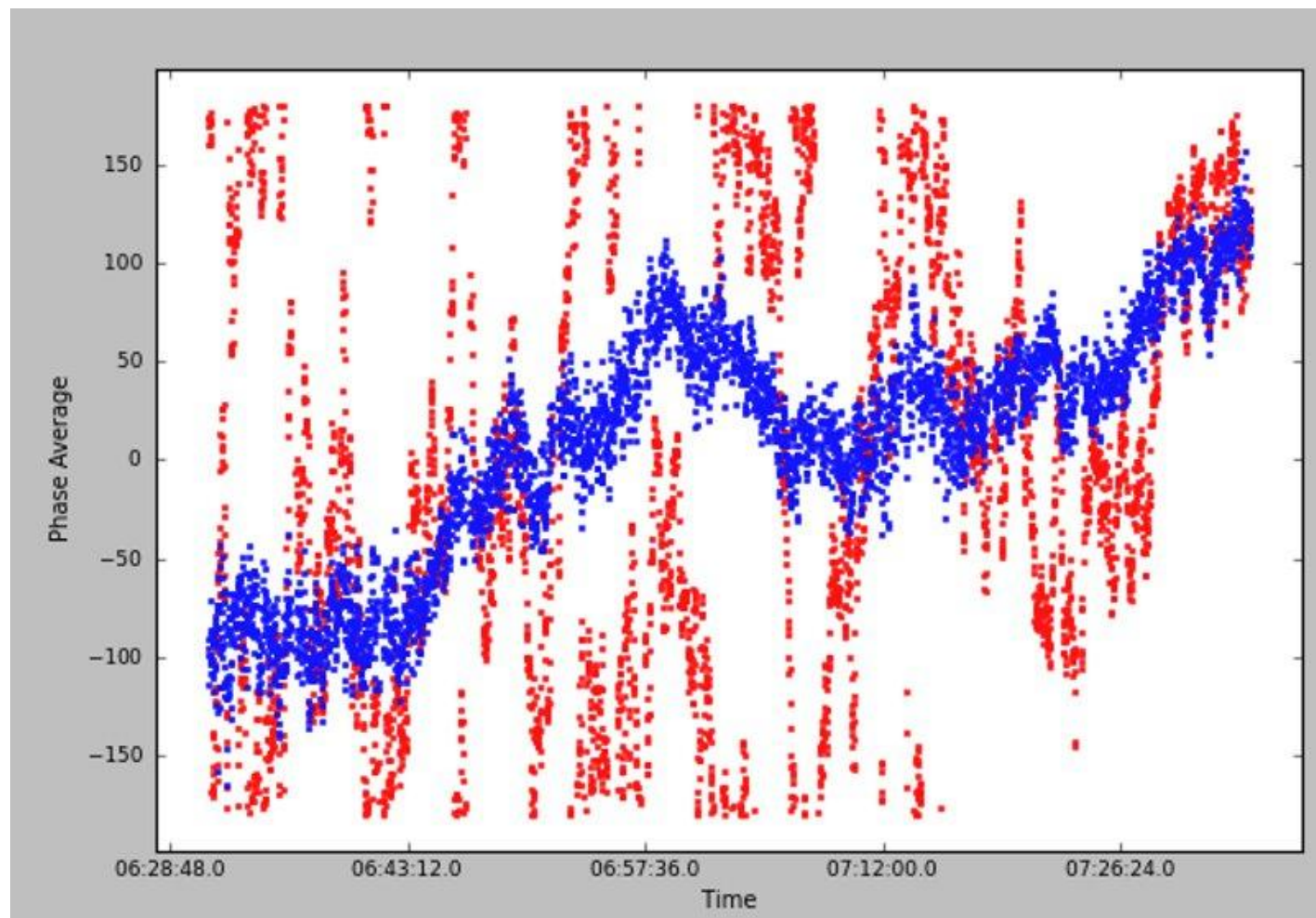
Initial Flaging

Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split



Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

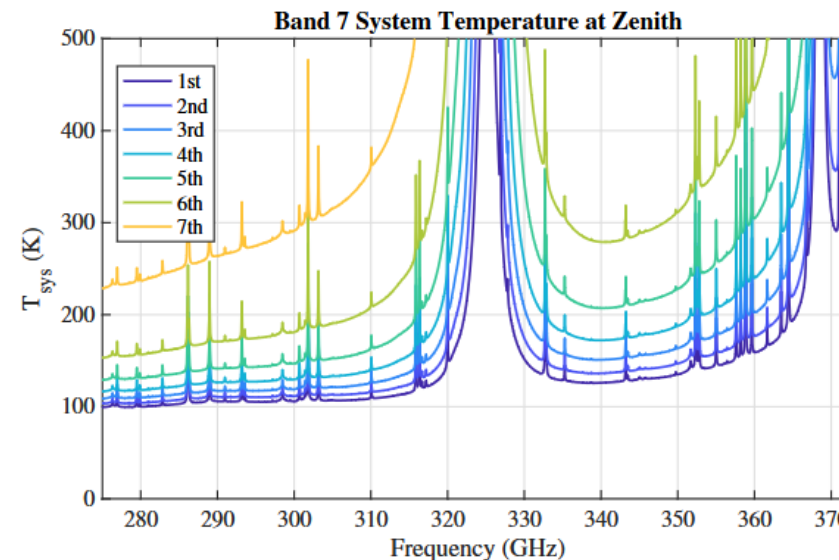
Applying Corrections and split

System Temperature (T_{sys}) corrections allow for the correction of atmospheric emission and opacity.

System Temperatures are important for

- Gain transfer across elevation
- Frequency dependent amplitude calibration

T_{sys} can vary from from ~ 50 K in Band 1 to ~ 1500 K in Band 10.



Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split

Tsys calibration table application:

From `data_prep.py`:

```
gencal(vis = 'uid___A002_Xdf0444_X10bd.ms',  
       caltable = 'uid___A002_Xdf0444_X10bd.ms.tsys',  
       caltype = 'tsys')
```

Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

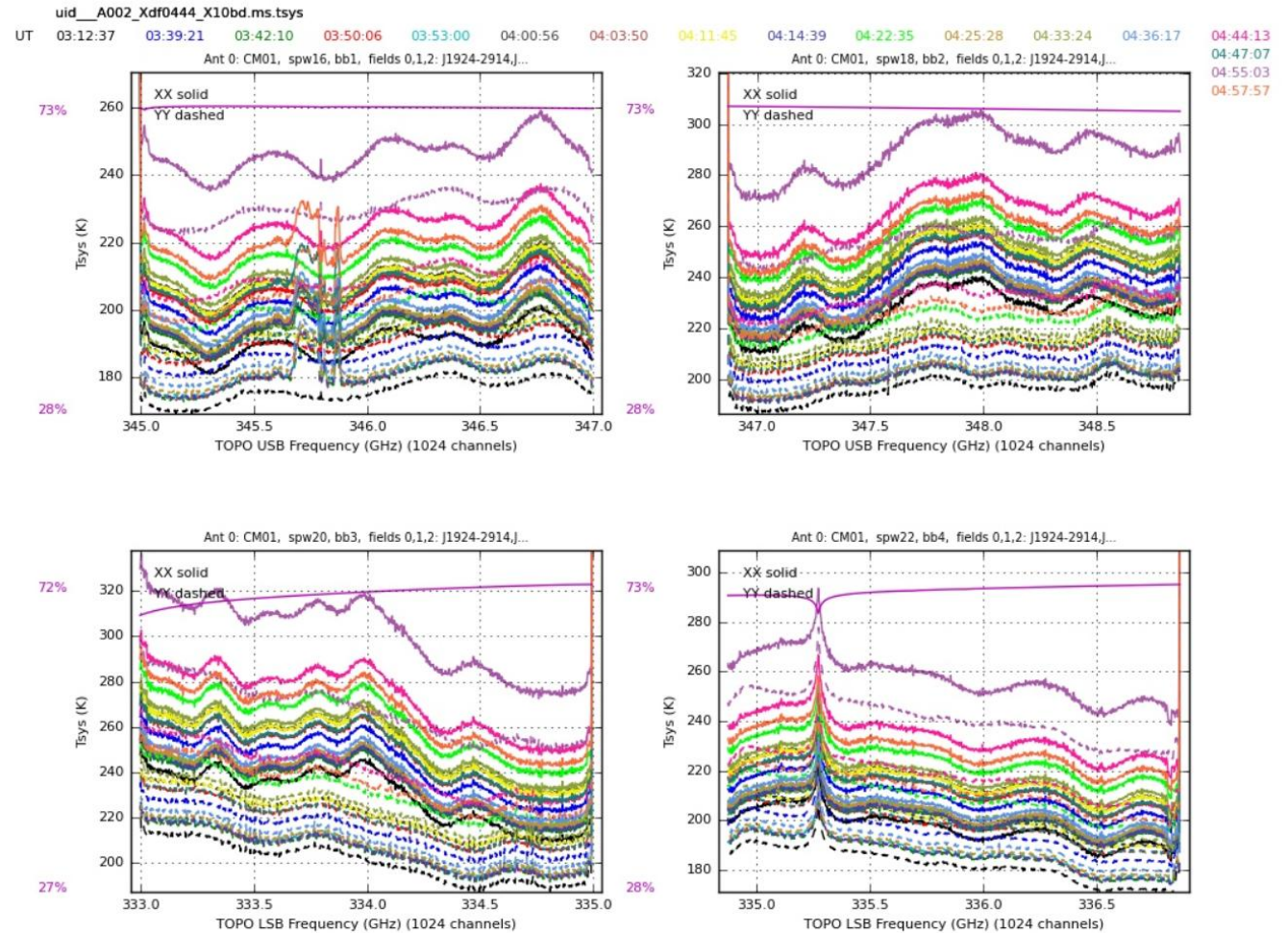
Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split

When looking at Tsys plots, look for very high spikes and discontinuities. These look okay, but some problems are there.



uid__A002_Xdf0444_X10bd.ms ObsDate=2019-07-18 plotbandpass3 v2.26: 2025-09-16T12:07:34-04:00 thun, C6.6.18

Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

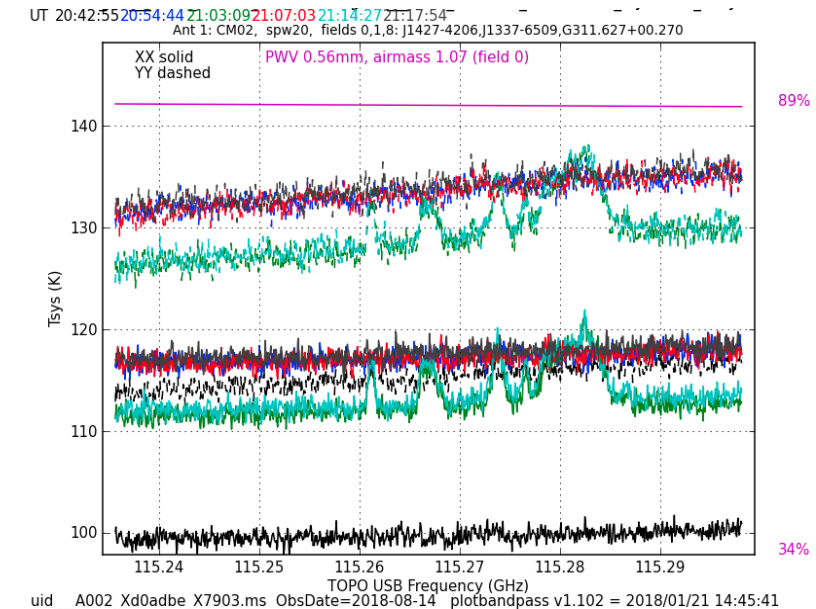
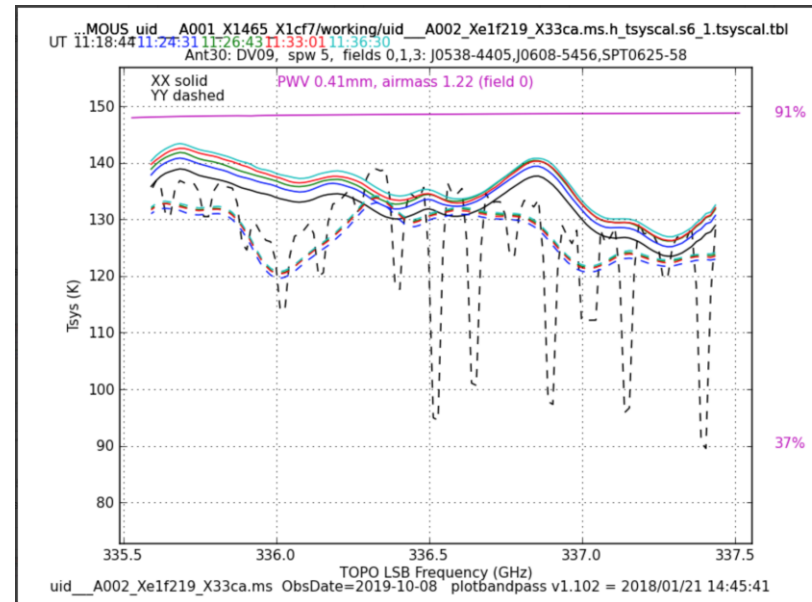
Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split

When looking at Tsys plots, look for very high spikes and discontinuities.

On the right is an example of Tsys contamination from an astronomical source. The channels involved would need flagged so we can interpolate across the contamination.



Data Inspection

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

Online Correction: Tsys

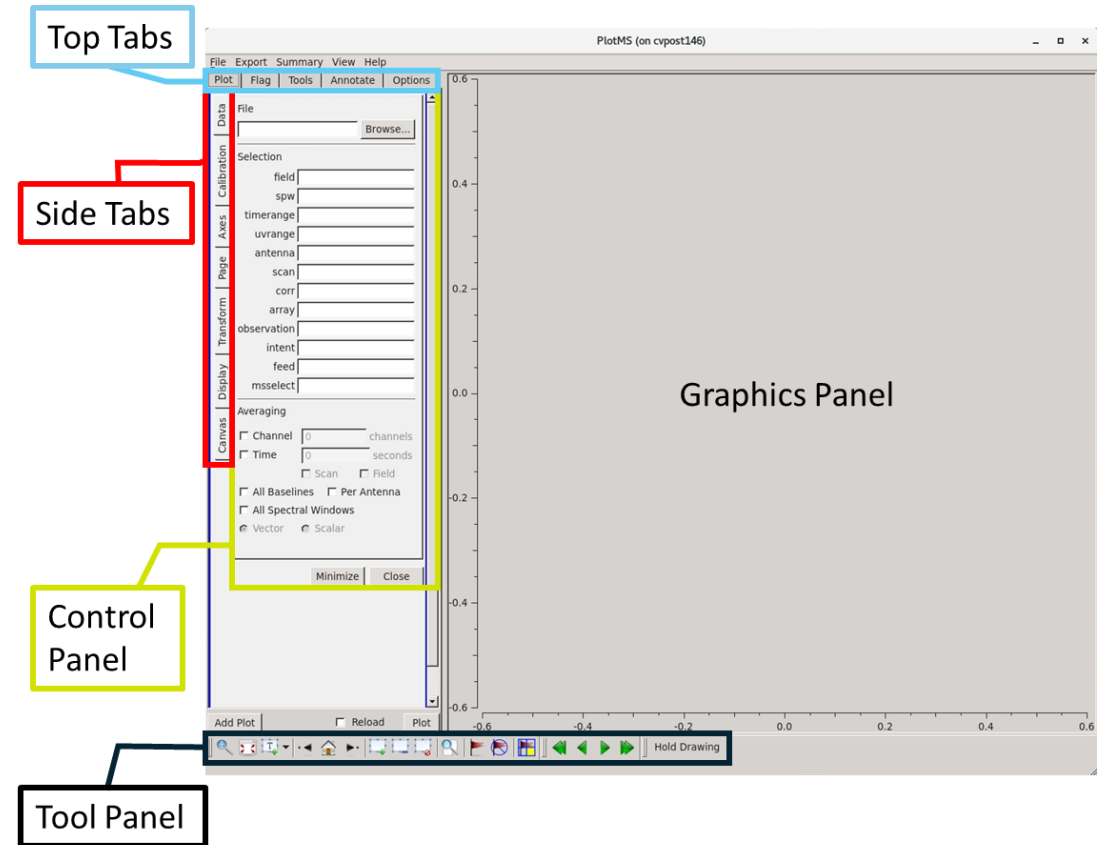
Online Correction: Antenna Position

Applying Corrections and split

Plotms is a general purpose graphical interface for plotting and flagging UV data and calibration tables.

You can start plotms from the CASA interface:

Plotms ()



Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split

Sgr A* shows tsys contamination

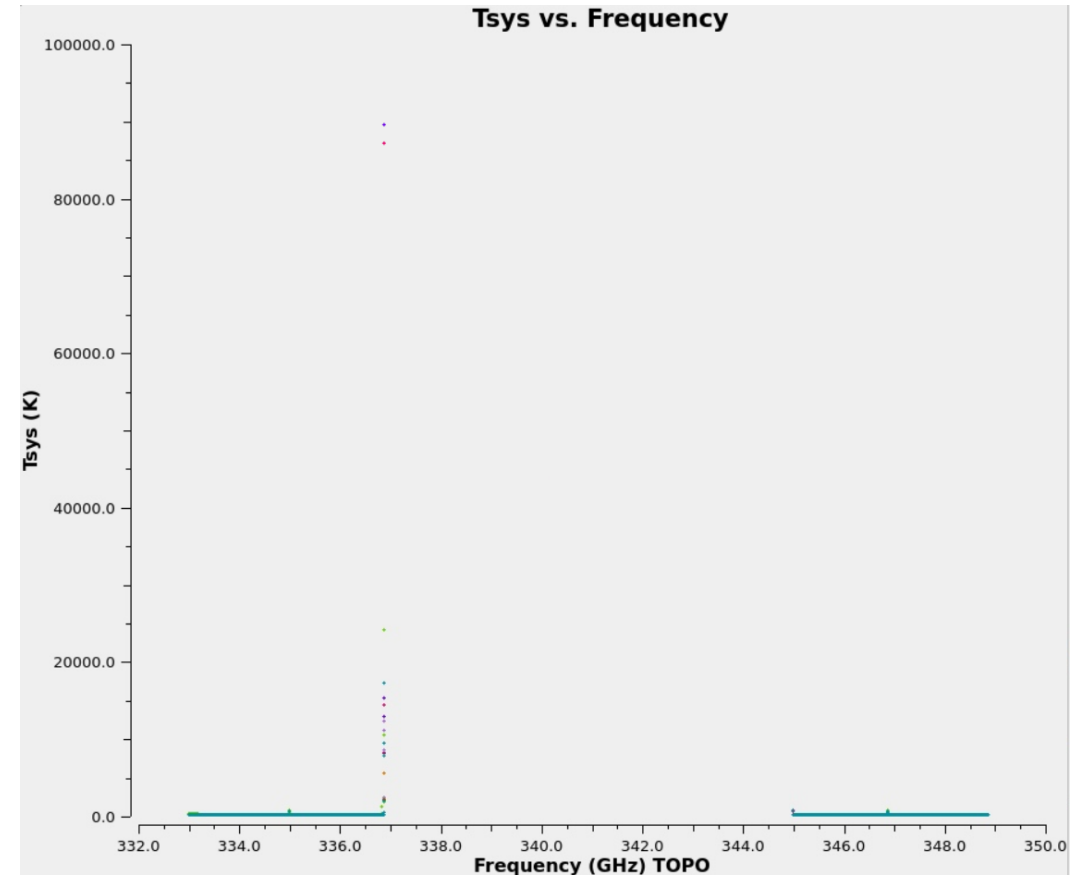
Start plotms(). We are going to make a plot!

We want to look at our tsys calibration table, plotting tsys vs frequency. We are going to average the data over all baselines, and color the points by scan.

1. In the side tab labeled “Data”, click on browse and select the file for the tsys table.
2. In the side tab labeled “Data”, check the box for all baselines.
3. In the side tab labeled “Axis”, select the axis to be Tsys vs frequency. Make sure the data column is “data”.
4. In the side tab labeled “Display” select “Colorize” and choose to colorize by scan.

Question: Plotms

- From the graph, can you tell what spectral window is which?
- What features stand out to you?



Expected plotms output

Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging




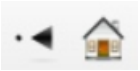
Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

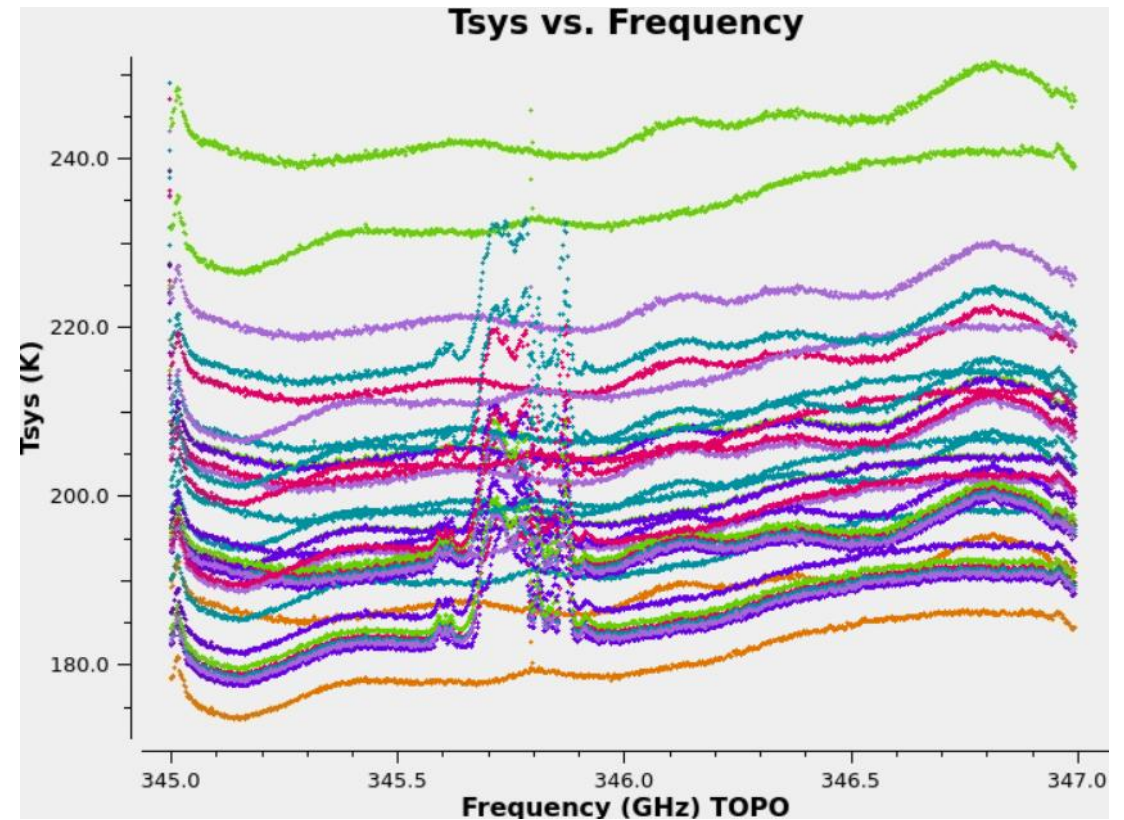
Applying Corrections and split

Sgr A* shows tsys contamination

- Highlight points with  and click locate ().
- Zoom in using 
- Zoom back out using 

Question: T_{sys}

- Can you find the channels affected by the T_{sys} contamination?



SPW 16 T_{sys} plot

Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split

Antenna position corrections update the accuracy of the antenna positions.

The positions are in meters with respect to the International Terrestrial Reference Frame (ITRF) [Earth center of mass] and normally look like this:

```
antposdict = {'data': {  
'CM05': [2225063.546167333, -5440128.205156475, -2481550.0793227106],  
'CM04': [2225074.0678211474, -5440115.249666032, -2481568.943718657],  
'CM03': [2225076.734637714, -5440122.932431717, -2481549.8241761364],  
'CM02': [2225070.958073706, -5440127.671532504, -2481544.654997596],  
'CM01': [2225080.355326109, -5440132.95741112, -2481524.789694901],  
'CM12': [2225089.4383623223, -5440119.773868732, -2481545.3989717225],  
'CM11': [2225065.4337490173, -5440120.432743067, -2481565.3131666067],  
'CM10': [2225078.929314356, -5440126.085415632, -2481541.021137885],  
'CM08': [2225063.532861316, -5440134.134661704, -2481537.2018038123],  
'CM07': [2225090.999713028, -5440126.601164743, -2481529.1336666625],  
'CM06': [2225084.240627651, -5440114.9985198965, -2481560.411347928]},  
'metadata': {'caltype': 'ALMA antenna positions', 'description': 'ALMA ITRF antenna  
positions in meters', 'product_code': 'antposalma', 'outfile': 'tmp_antennapos.json',  
'hosts': ['https://asa.alma.cl/uncertainties-  
service/uncertainties/versions/last/measurements/casa/'], 'asdm':  
'uid://A002/Xdf0444/X10bd', 'search': 'auto', 'successful_url':  
'https://asa.alma.cl/uncertainties-  
service/uncertainties/versions/last/measurements/casa//?asdm=uid%3A%2F%2FA002%2FXdf0444%2FX  
10bd&search=auto', 'timestamp': '2026-01-21 14:17:38.984734'}}}
```

Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split

Our dataset did require antenna position corrections. (How did I know?)

However, we should still follow this step to make sure that the positions are correct even if not all datasets require the update. Write the dictionary to a .json file and then generate the calibration table.

From `calibration.py`:

```
with open('antennapos.json', 'w') as f:  
    json.dump(antposdict, f)
```

```
gencal(vis='uid__A002_Xdf0444_X10bd.ms',  
       caltype='antpos',  
       infile='antennapos.json',  
       caltable='uid__A002_Xdf0444_X10bd.ms.antpos')
```

Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split

Finally, we apply our corrections and split out the data.

Tsysmap is a list mapping each SPW to the tsys SPW it uses for calibration.

From [calibration.py](#):

```
tsysmap = tsysspwwmap(vis = 'uid__A002_Xdf0444_X10bd.ms', tsystable  
= 'uid__A002_Xdf0444_X10bd.ms.tsys', tsysChanTol = 1)
```

```
applycal(vis = 'uid__A002_Xdf0444_X10bd.ms',  
        field = '0',  
        spw = '16,18,20,22',  
        gaintable = ['uid__A002_Xdf0444_X10bd.ms.tsys',  
                    'uid__A002_Xdf0444_X10bd.ms.antpos'],  
        gainfield = ['0', ''],  
        interp = 'linear,linear',  
        spwmap = [tsysmap,[]],  
        calwt = True,  
        flagbackup = False)
```

Data Preparation

Importing your ASDM

Table Corrections

Initial Flaging

Online Correction: WVR

Online Correction: Tsys

Online Correction: Antenna Position

Applying Corrections and split

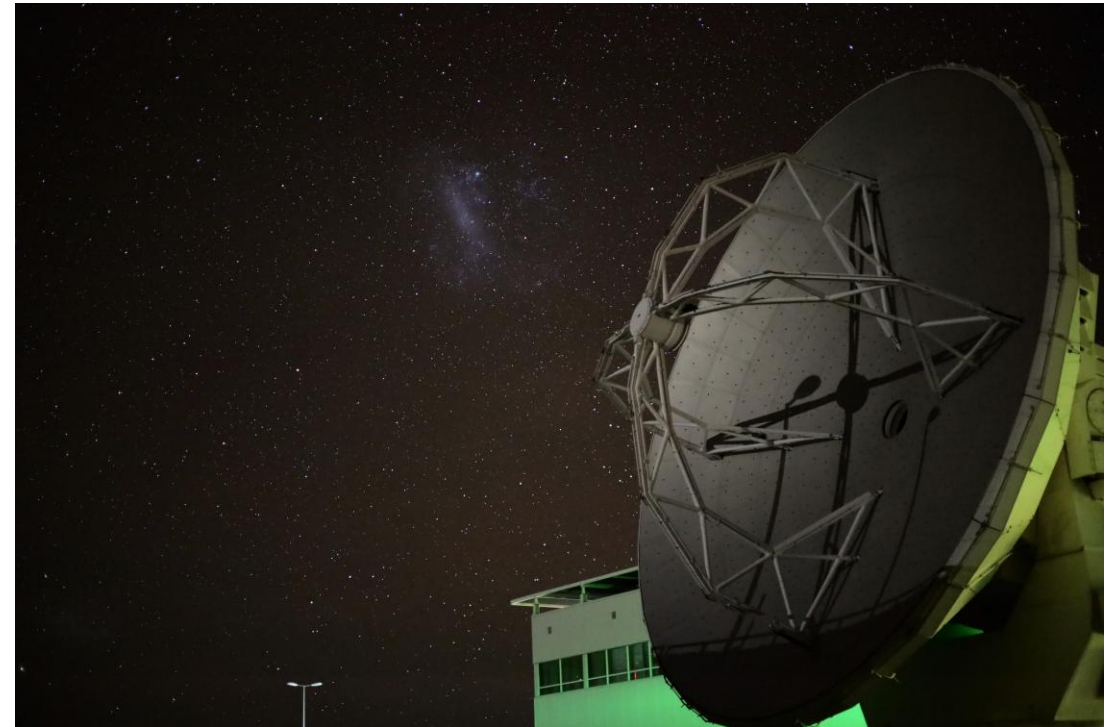
Repeat this step for field 1 and field 2. We will then split out the data.

```
mstransform(vis = 'uid___A002_Xdf0444_X10bd.ms',  
            outputvis = 'uid___A002_Xdf0444_X10bd.ms.split',  
            datacolumn = 'corrected',  
            spw = '16,18,20,22',  
            reindex = False,  
            keepflags = True)
```

This will complete the data preparation!

Discussion

- Before we move on, what are your questions, comments, and concerns?



ALMA PM Antenna at Night

Data Inspection

Data Inspection

Viewing Observation Information

Plotting Antenna Distribution

Viewing Uncalibrated Data

We are going to inspect our data to learn about the observation and check for problems.

We are still following the calibration.py script if you want to copy the commands and follow along.



Jim Torson analyzes data from the Very Large Array in its computer room in 1983. (Image Credit: NRAO)

Data Inspection

Viewing Observation Information

Plotting Antenna Distribution

Viewing Uncalibrated Data

Lets start by looking at a summary of the measurement set.

Use the task `listobs()`

From `calibration.py`:

```
listobs("uid___A002_Xdf0444_X10bd.ms.split")
```

Question: Listobs

After taking a look at the listobs output in the logger window, what information do you see that you think may be useful during processing?

You can write your listobs to a file using:

```
listobs(vis = 'uid___A002_Xdf0444_X10bd.ms.split',  
        listfile = 'uid___A002_Xdf0444_X10bd.ms.split.listobs')
```

Data Inspection

Viewing Observation Information

Plotting Antenna Distribution

Viewing Uncalibrated Data

Lets start by looking at a summary of the measurement set.

Use the task `listobs()`

```
listobs("uid___A002_Xdf0444_X10bd.ms.split")
```

MeasurementSet Name: uid___A002_Xdf0444_X10bd.ms.split

MS Version Fields: 4

Fields: 3	ID	Code Name	RA	Decl	Epoch	SrcId	nRows
0	none	J1924-2914	19:24:51.055956	-29.14.30.12106 ICRS	0	33000	
1	none	J1700-2610	17:00:53.154063	-26.10.51.72542 ICRS	1	39072	
2	none	SgrAstar	17:45:40.038000	-29.00.28.06900 ICRS	2	88968	

Spectral Windows: (4 unique spectral windows and 1 unique polarization setups)

TotBW(kHz)	CtrFreq(MHz)	BBC Num	Corrs	SpwID	Name	#Chans	Frame	Ch0(MHz)	ChanWid
16	X67236705#ALMA_RB_07#BB_1#SW-01#FULL_RES	1024	TOP0	344996.785	1953.125	2000000.0	345995.8086	1	XX YY
18	X67236705#ALMA_RB_07#BB_2#SW-01#FULL_RES	1024	TOP0	346871.763	1953.125	2000000.0	347870.7859	2	XX YY
20	X67236705#ALMA_RB_07#BB_3#SW-01#FULL_RES	1024	TOP0	334994.897	-1953.125	2000000.0	333995.8738	3	XX YY
22	X67236705#ALMA_RB_07#BB_4#SW-01#FULL_RES	1024	TOP0	336869.897	-1953.125	2000000.0	335870.8738	4	XX YY

Data Inspection

Viewing Observation Information

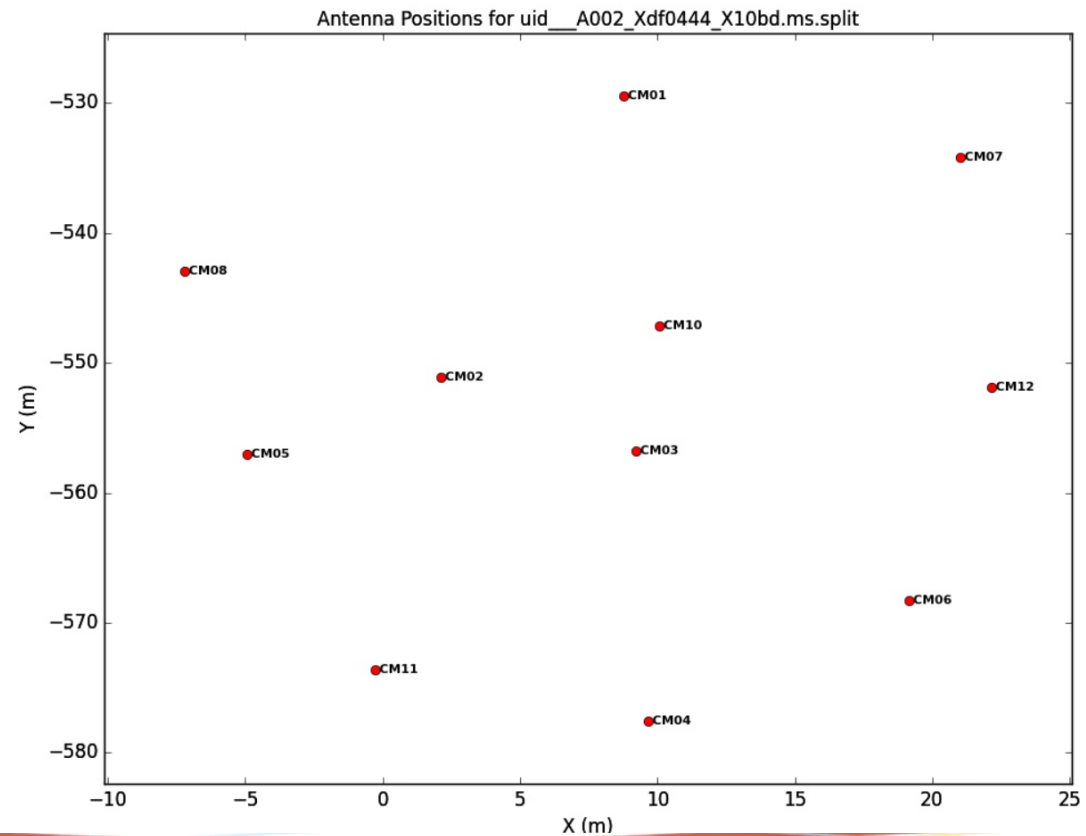
Plotting Antenna Distribution

Viewing Uncalibrated Data

Now, we will plot the antenna distribution using `listobs()`

From `calibration.py`:

```
plotants("uid___A002_Xdf0444_X10bd.ms.split",  
         figfile="plotants.png")
```



Question: Reference Antenna

- When calibrating, we typically want a reference antenna that is well behaved and towards the center of the array.
- What antennas do you think would work out well?



Data Inspection

Viewing Observation Information

Plotting Antenna Distribution

Viewing Uncalibrated Data

We are going to choose CM12 as our reference antenna. This will be used in all commands going forward.

WAIT?!?

I just said we wanted to use an antenna in the center of the array?

Well, I didn't notice before I sent out the script... Also, the 7m array is small enough that it is not a big deal.

Data Inspection

Viewing Observation Information

Plotting Antenna Distribution

Viewing Uncalibrated Data

Plotms is a general purpose graphical interface for plotting and flagging UV data and calibration tables.

You can start plotms from the CASA interface:

```
Plotms ( )
```

You can also fully specify the plotms command from the command line.
For example:

```
plotms(vis="uid___A002_Xdf0444_X10bd.ms.split",  
       xaxis="time", yaxis="amp",  
       averagedata=True,  
       avgchannel="1e3",  
       coloraxis="field")
```

Data Inspection

Viewing Observation Information

Plotting Antenna Distribution

Viewing Uncalibrated Data

The image shows a screenshot of the PlotMS software interface. The window title is "PlotMS (on cvpost146)". The menu bar includes "File", "Export", "Summary", "View", and "Help". Below the menu bar is a toolbar with "Plot", "Flag", "Tools", "Annotate", and "Options". A vertical sidebar on the left contains tabs for "Data", "Calibration", "Axes", "Page", "Transform", "Display", and "Canvas". The "Data" tab is selected, showing fields for "File" (with a "Browse..." button), "Selection" (with fields for "field", "spw", "timerange", "uvrange", "antenna", "scan", "corr", "array", "observation", "intent", "feed", "msselect"), "Averaging" (with fields for "Channel" and "Time"), and checkboxes for "All Baselines", "Per Antenna", "All Spectral Windows", "Vector", and "Scalar". At the bottom of the sidebar are "Minimize" and "Close" buttons. The main area is a "Graphics Panel" with a plot showing a grid from -0.6 to 0.6 on both axes. At the bottom of the window is a "Tool Panel" with buttons for "Add Plot", "Reload", "Plot", and a "Hold Drawing" button.

Top Tabs

Side Tabs

Control Panel

Tool Panel

Graphics Panel

Data Inspection

Viewing Observation Information

Plotting Antenna Distribution

Viewing Uncalibrated Data

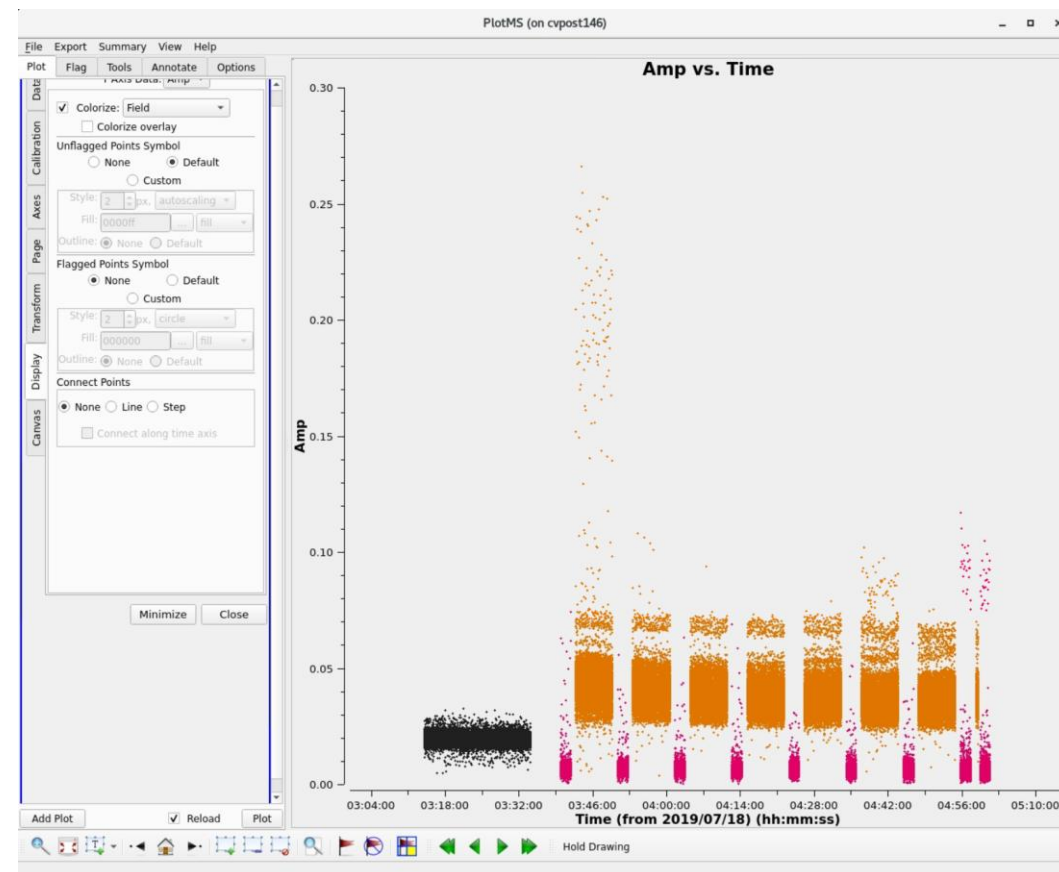
Start `plotms()`. We are going to make more plots!

We want to look at `uid__A002_Xdf0444_X10bd.ms.split`, plotting amplitude vs time. We are going to average the data by channel, and color the points by field.

1. In the side tab labeled “Data”, click on browse and select the file measurement set.
2. In the side tab labeled “Data”, check the box for Channel averaging. Then enter a number that averages all the channels (in an spw) together. (Look at your `listobs` output!)
3. In the side tab labeled “Axis”, select the axis to be Amplitude vs Time. Make sure the data column is “data”.
4. In the side tab labeled “Display” select “Colorize” and choose to colorize by field.

Question: Plotms

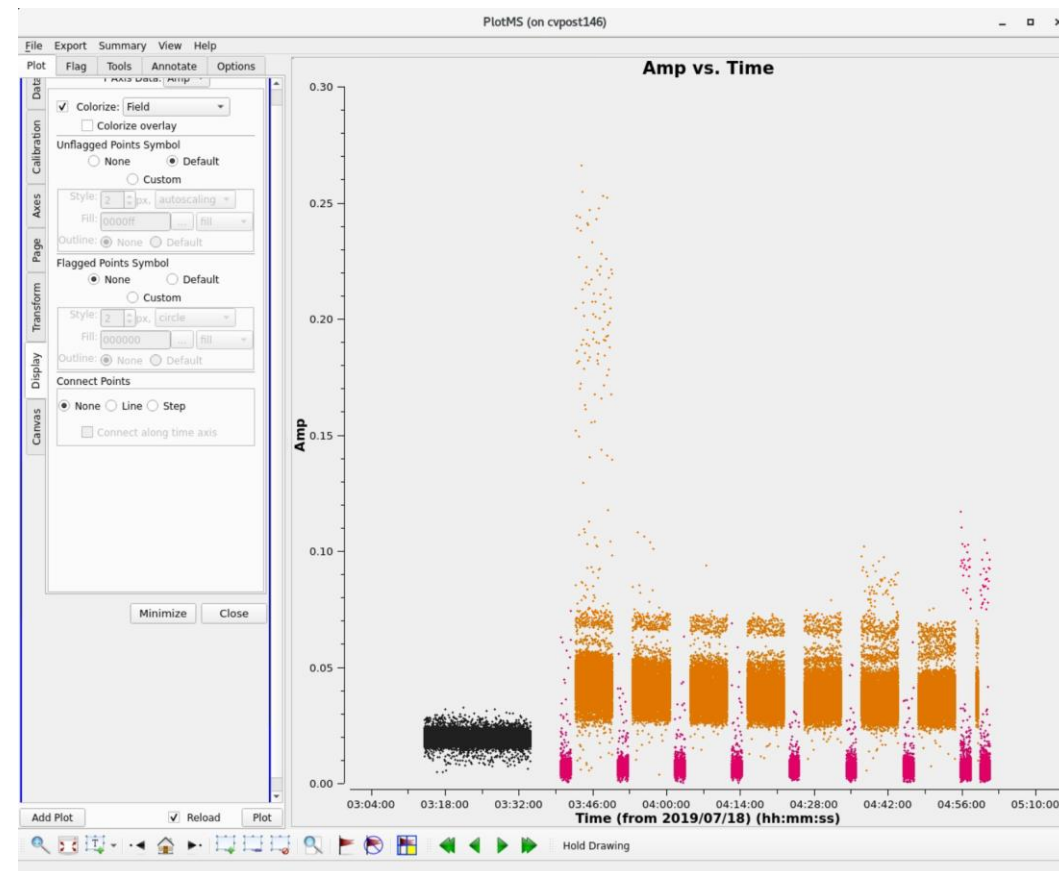
- From the graph, can you tell what source is the science target?
- Did you look at some of the options? What were some?
 - Axis options?
 - Colorize options?



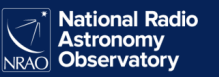
Expected plotms output

Your Turn:

- Make a plot using plotms of the uncalibrated data using a different combination of axis. Ask an assistant if you need assistance!
- What combo did you use? Did you run into any difficulties.



Calibration



Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

We are finally ready to discuss calibration!

Derive Calibration Tables

- **setjy**: set “model” (correct) visibilities using known model for a calibrator
- **bandpass**: calculate bandpass calibration table (amp/phase vs frequency)
- **gaincal**: calculate temporal gain calibration table (amp/phase vs time)
- **fluxscale**: apply absolute flux scaling to calibration table from known source

Manipulate Your Measurement Set

- **flagdata/flagcmd/flagmanager**: flag (remove) bad data
- **applycal**: apply calibration table(s) from previous steps
- **split**: split off calibrated data from your ms

Inspect Your Data and Results

- **plotms**: inspect your data and calibration tables interactively

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Before proceeding, we want to save the flagging of our data. This is good practice in case a mistake is made later.

Only run this once. The if statement before this in **calibration.py** will ensure it is not run twice.

```
flagmanager(vis = 'uid___A002_Xdf0444_X10bd.ms.split',  
            mode = 'save',  
            versionname = 'Original')
```

Later, if you want to restore to the original flagging state, we can restore the flags to this original state using:

From **calibration.py**

```
flagmanager(vis = 'uid___A002_Xdf0444_X10bd.ms.split',  
            mode = 'restore',  
            versionname = 'Original')
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Initial Flagging includes data we know to be problematic even without visual inspection:

Shadowing

- Issue at low elevations
- Issue for compact arrays

In CASA:

```
flagdata(vis = 'uid__A002_Xdf0444_X10bd.ms.split',  
         mode = 'shadow',  
         flagbackup = False)
```

Observing Log

- Many observatories will note weather or hardware problems that affect the data.

Other obvious errors

- Edge Channels

In CASA

```
flagdata(vis = 'uid__A002_Xdf0444_X10bd.ms.split',  
         mode = 'manual',  
         spw =  
         '16:0~6;1019~1023,18:0~6;1019~1023,20:0~6;1019~1023,22:  
         0~6;1019~1023',  
         flagbackup = False)
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

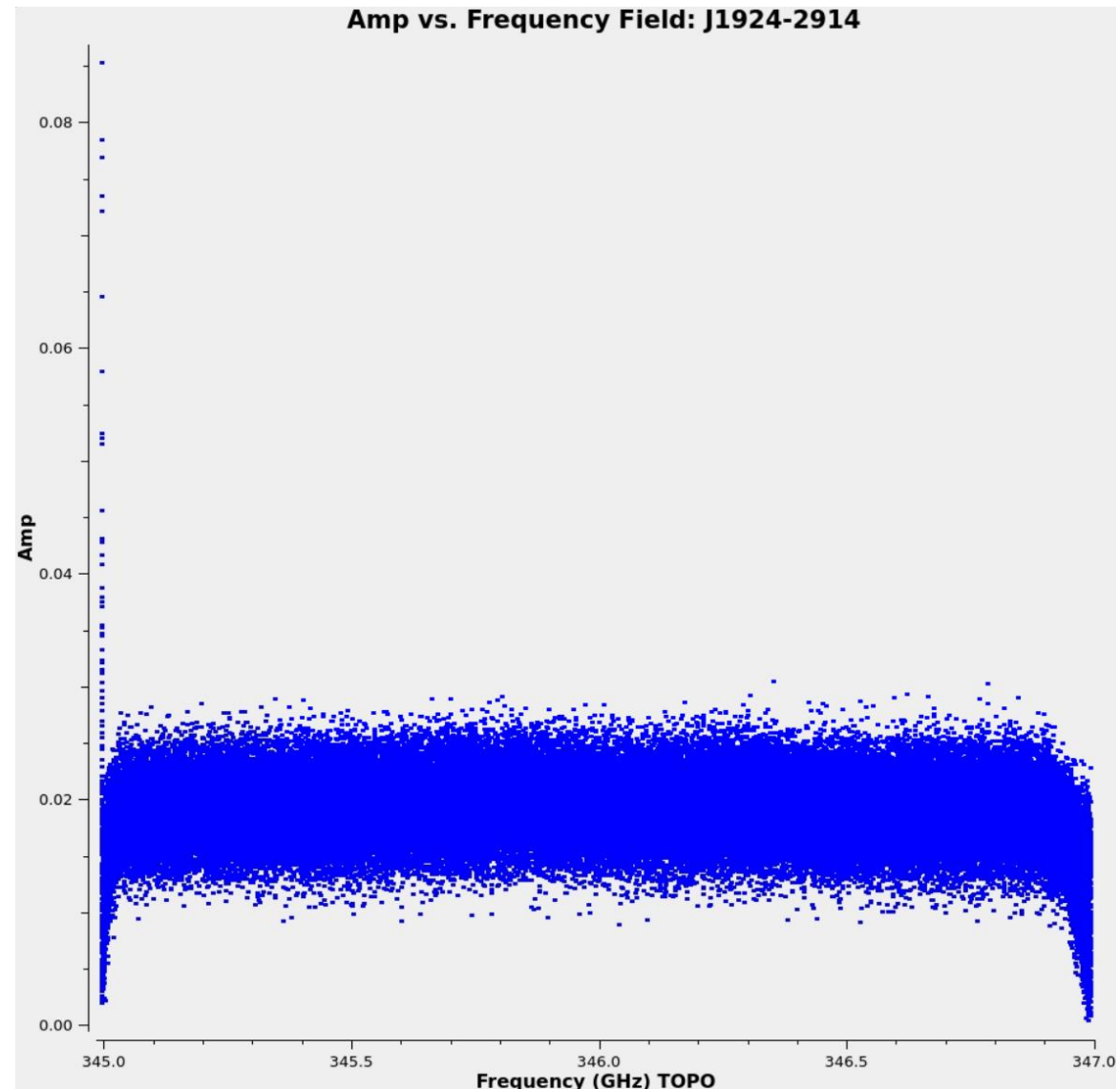
Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration



Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Your first time though the data, no other flags are applied.

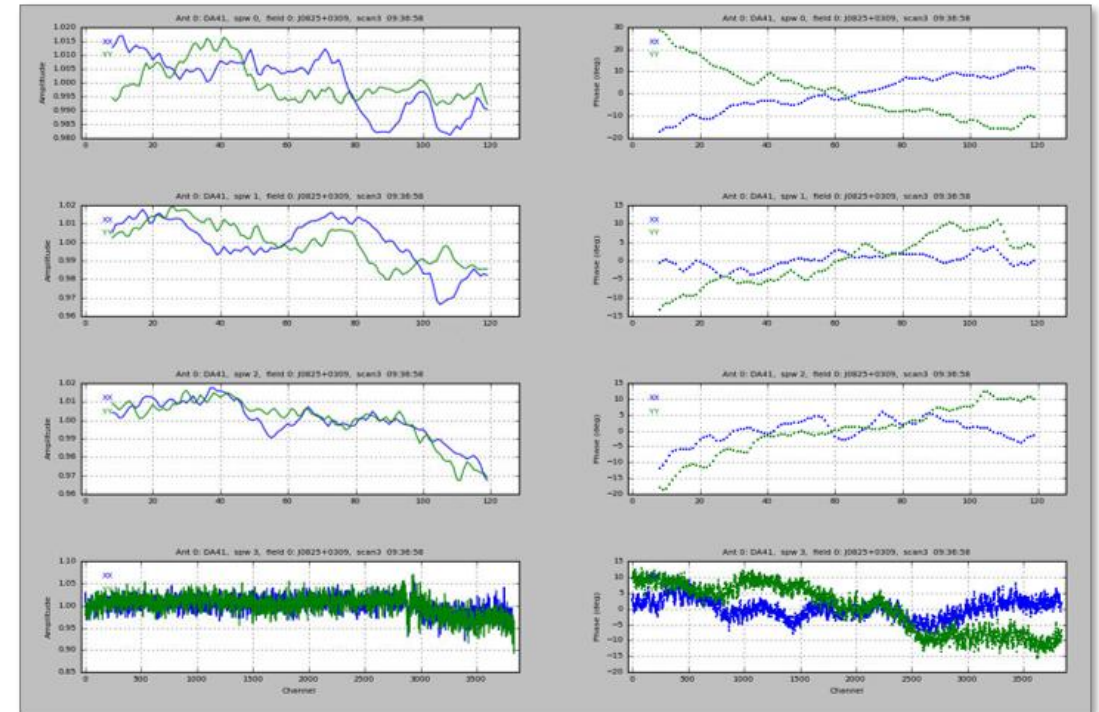
After the rest of the calibration, if you identify any bad data, flags are placed in this part of the script.

As you can see, most of the bad data are from edge channels. These will be flagged anyway.

So we will skip this for now.

Question: Bandpass

Why is bandpass calibration important?



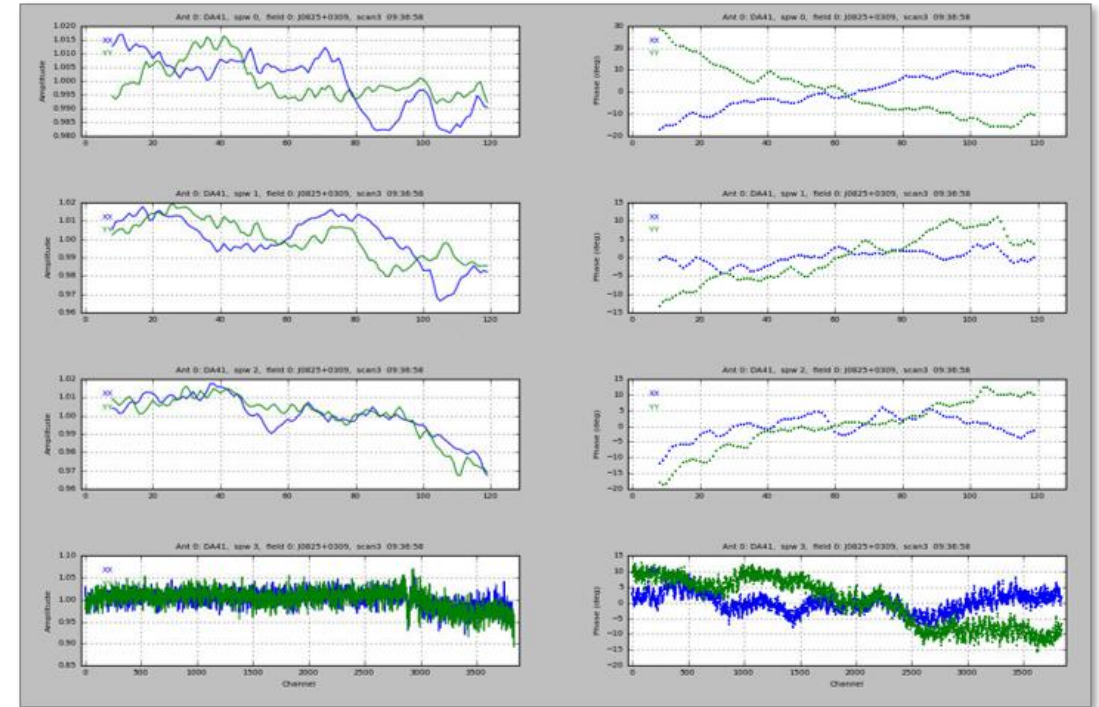
A sample of bandpass solutions.

Question: Bandpass

Why is bandpass calibration important?

Bandpass Calibration allows us to correct for the frequency dependent part of the gains $(B_{ij}(t, \nu))$.

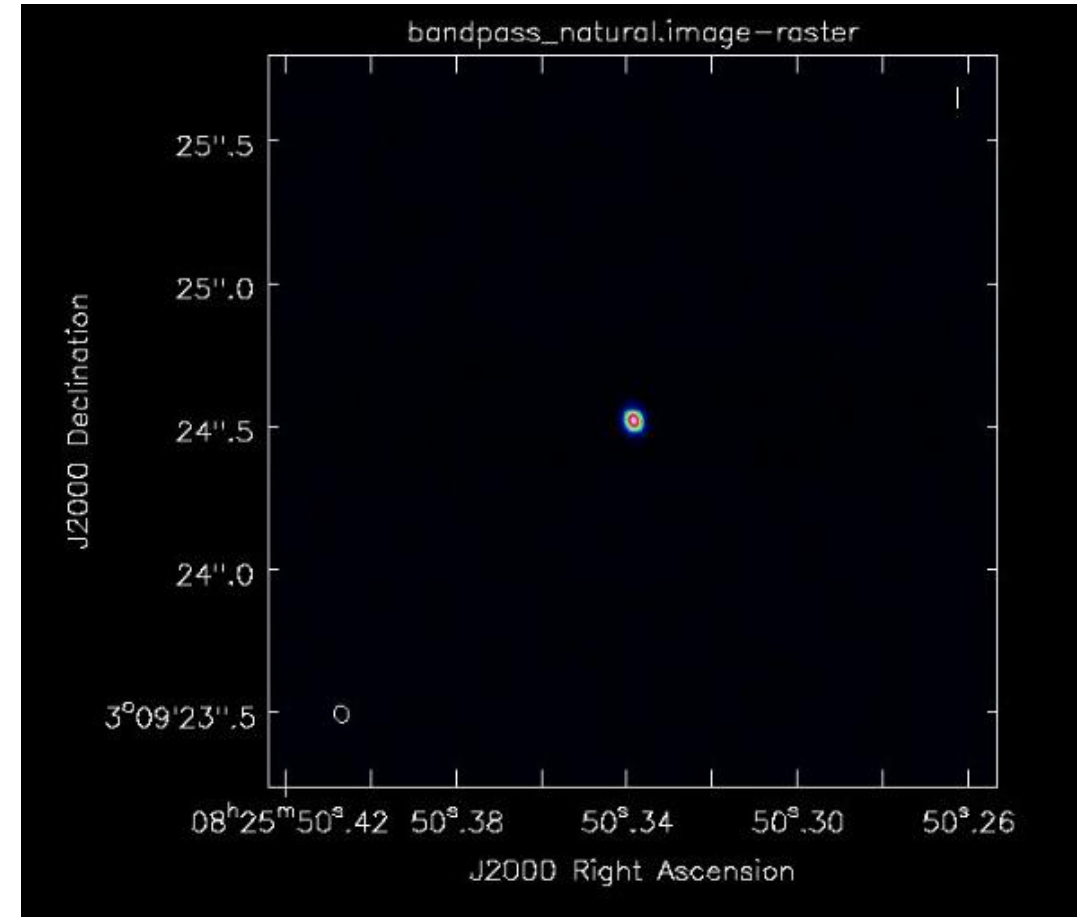
B_{ij} is usually treated as constant over the length of an observation but may have a slow time dependence.



A sample of bandpass solutions.

Question: Bandpass

What makes a good bandpass calibrator?



Bandpass Calibrator J0825+0309

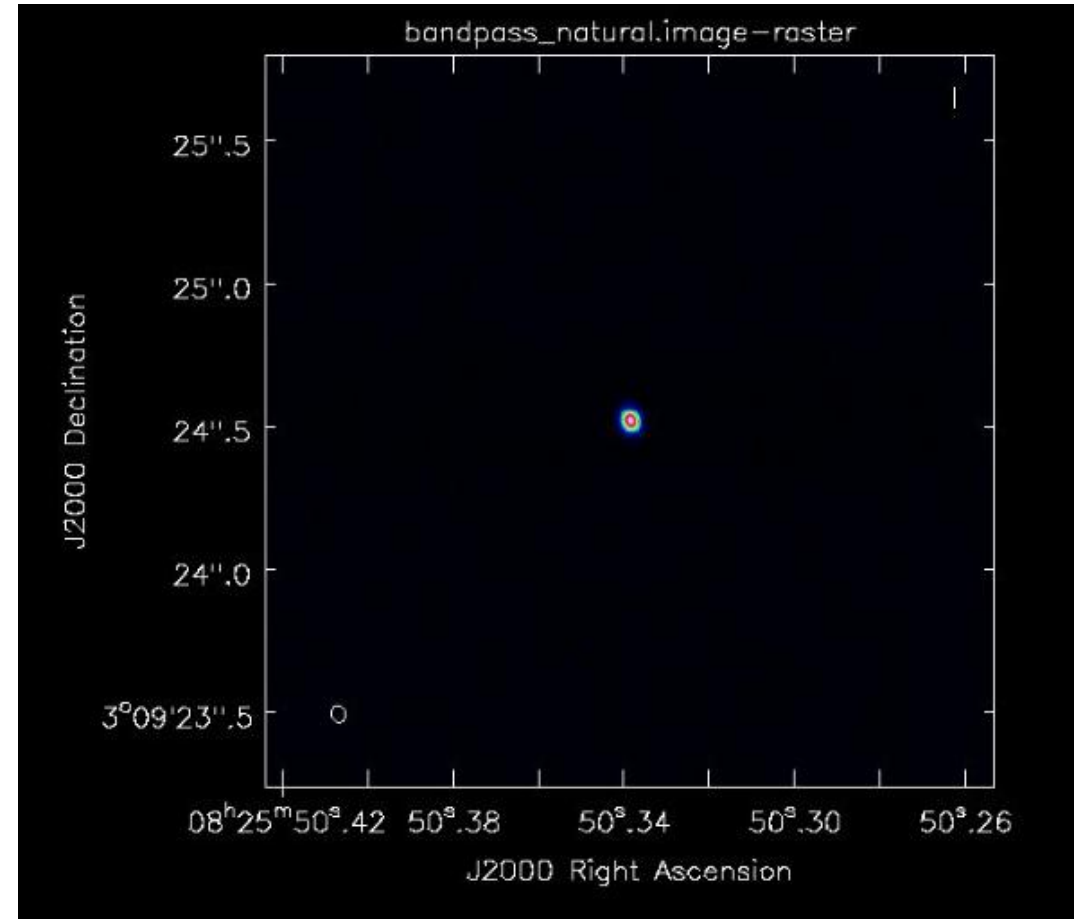
Question: Bandpass

What makes a good bandpass calibrator?

Best targets are bright, flat-spectrum sources with featureless spectra

- Although point-source not absolutely required, beware frequency dependence of resolved sources
- If necessary, can specify a spectral index using *setjy*

Don't necessarily need to be near science target on the sky



Bandpass Calibrator J0825+0309

Question: Bandpass

Bandpass corruption is typically antenna-based.

Does this give us any advantages compared to a baseline-based calibration?



Question: Bandpass

Bandpass corruption is typically antenna-based.

Does this give us any advantages compared to a baseline-based calibration?

Yes, this means we have $N-1$ baselines to consider. The problem is over-determined.



Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

- We will use *gaincal* to measure time variation of phase
- Then use *bandpass* task
 - We will calibrate channel-to-channel variation (preferred method)
 - Alternatively, could fit a smooth function
 - Pay close attention to solutions; e.g. bright calibrators are rare, esp. at Bands 8, 9, and 10
- Use *applycal* to apply the bandpass solution to other sources

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Using the Listobs output, determine which source is the bandpass.

- This is J0825+0309 (identified as field 0).

Run the following command:

```
gaincal(vis = 'uid__A002_Xdf0444_X10bd.ms.split',  
        caltable = 'uid__A002_Xdf0444_X10bd.ms.split.ap_pre_bandpass',  
        field = '0', # J1924-2914  
        spw = '16:0~1023,18:0~1023,20:0~1023,22:0~1023',  
        scan = '3',  
        solint = 'int',  
        refant = 'CM12',  
        calmode = 'p')
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

While gaincal runs:

Gaincal is the general purpose task to solve for time-dependent amplitude and phase variations for each antenna.

Here we carry out a short-timescale phase (“p”) solution (“int”) on the bandpass calibrator. This is saved as a calibration table “phase_int_bpass.cal”.

We only fit the phase to the central channels to avoid edge channel effects.

The Field and scan designations are redundant, but good to make sure the correct data for the solve.

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration


Setting the Flux Scale

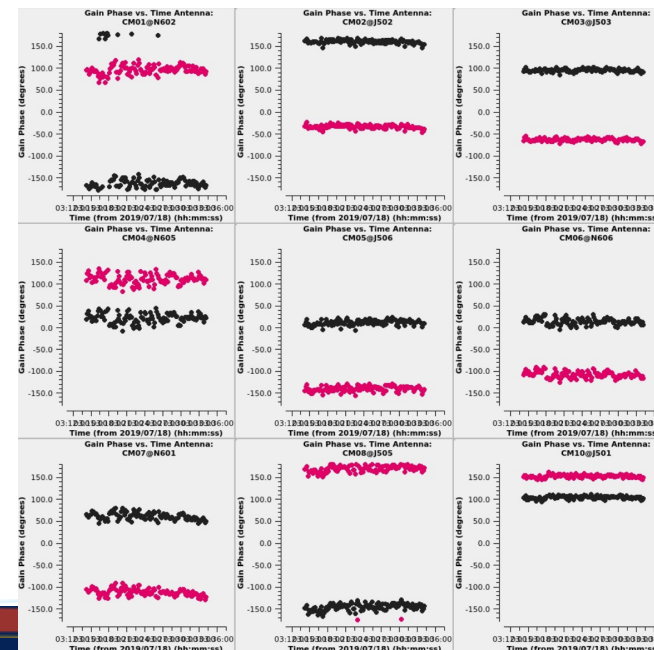
Renormalization Correction

Applying the Calibration

Checking the Calibration

Plot the solutions using plotms!

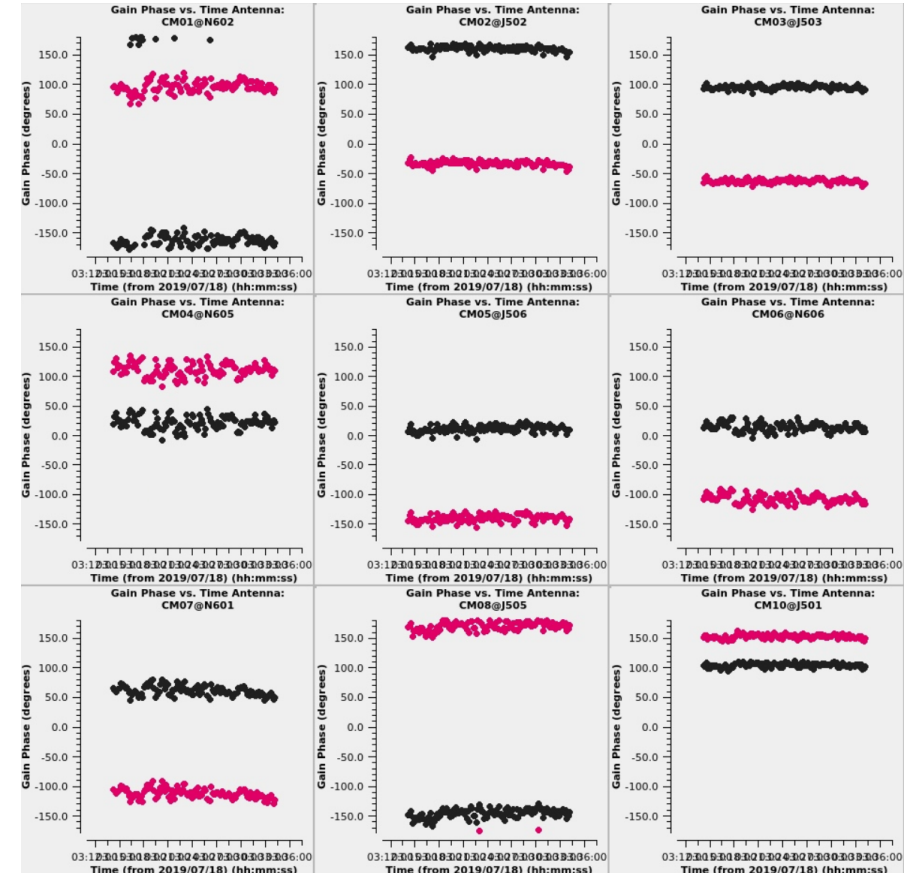
- Plot the phase_int_bpass.cal table in phase vs time. Spw 0 only. Color by correlation.
- Under the side tab labeled Axis, set the range on phase to be -180 to +180 degrees.
- New this time: iteration axis! Under the page tab, select axis => antenna
- Under the top tab labeled “Options”, set rows and columns both to three.
- Use the tool panel buttons  to navigate the plots.



Question: Phases

A few plots stand out in our phase solutions.

- Why does the phase plot for CM12 have the shape it has?
- What is the cause of the discontinuities for CM01?



Good phase plots for the bandpass.

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Now we will carry out a bandpass solution. This will solve for the amplitude and phase corrections needed for each channel on a per-antenna basis.

Lets run the bandpass task:

```
bandpass(vis = 'uid__A002_Xdf0444_X10bd.ms.split',  
         caltable = 'uid__A002_Xdf0444_X10bd.ms.split.bandpass',  
         field = '0', # J1924-2914  
         scan = '3',  
         solint = 'inf',  
         combine = 'scan',  
         refant = 'CM12',  
         solnorm = True,  
         bandtype = 'B',  
         gaintable = 'uid__A002_Xdf0444_X10bd.ms.split.ap_pre_bandpass')
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

While that is running:

- We use gaintable to feed the short-timescale phase solution to the task.
 - This table will be applied before the bandpass solution is carried out.
- We have not flux calibrated yet, so we are determining a normalized solution with solnorm=True.
- Solint=inf and combine=scan combines all data.
- Bandtype = B is the standard

Question: Bandpass

Why is it okay to combine all the bandpass data together?



ALMA CM and PM antennas

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

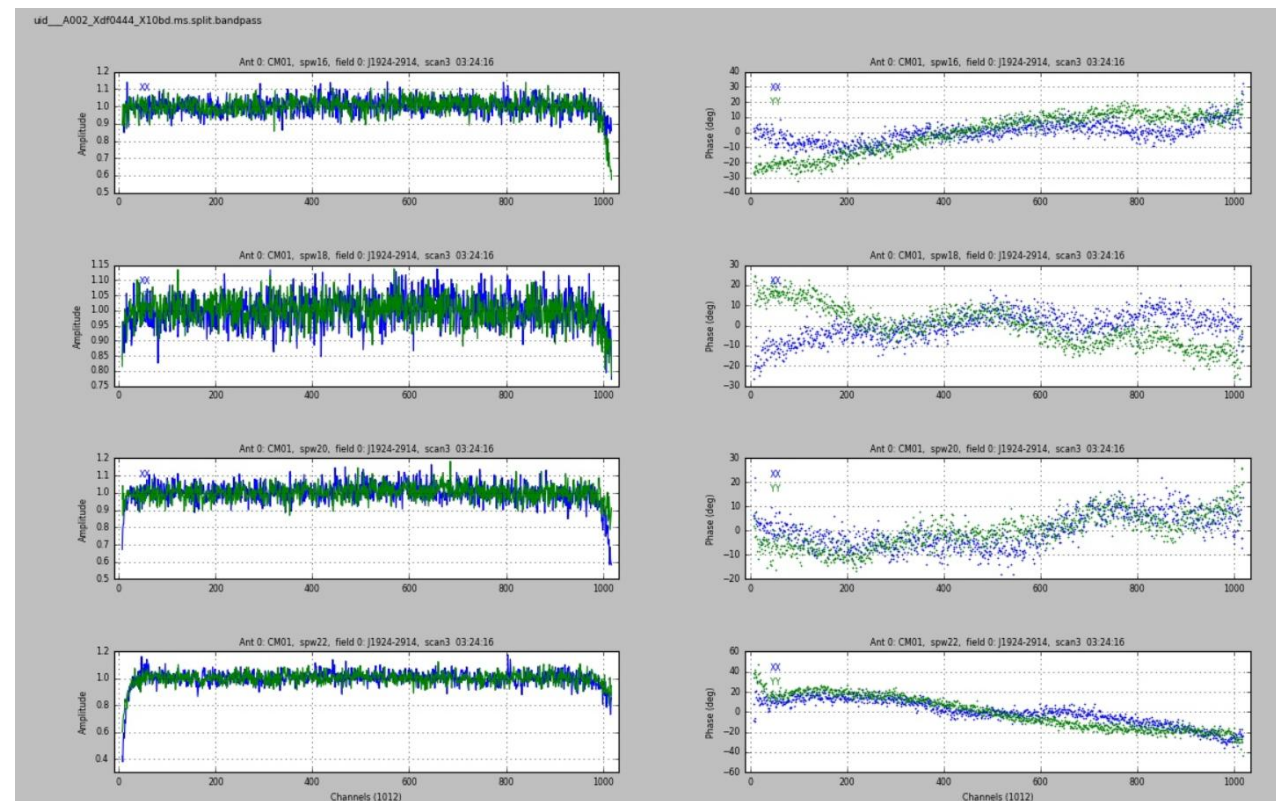
Renormalization Correction

Applying the Calibration

Checking the Calibration

We are going to use the `plotbandpass()` task to plot the bandpass solutions.

```
plotbandpass(  
  caltable="uid__A002_Xdf0444_X10bd.ms.split.bandpass",  
  xaxis="chan",  
  yaxis="both",  
  subplot=42)
```



Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

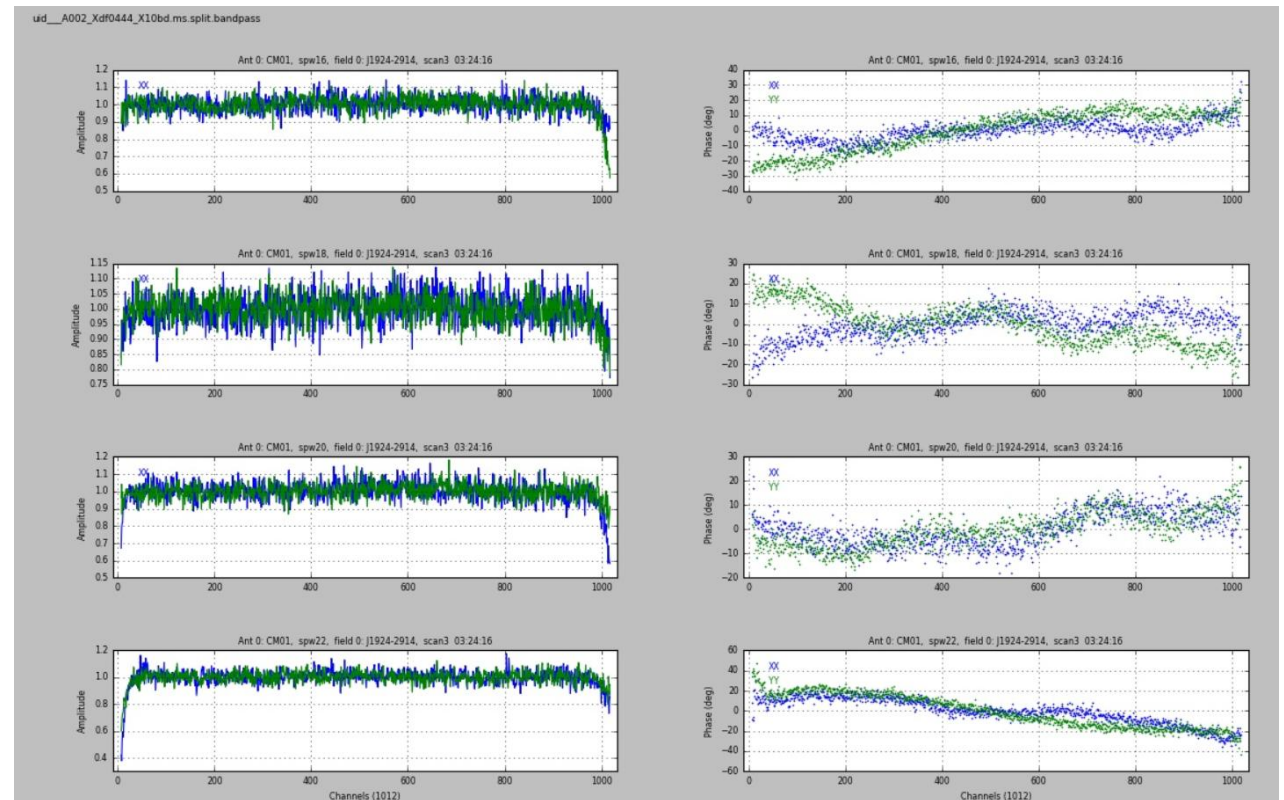
Applying the Calibration

Checking the Calibration

What are we looking for in a bandpass solution?

The bandpass solution should be smooth. Some features at the location of atmospheric lines are expected.

Keep an eye out for discontinuities.. Standard practice is to flag whole antenna if you find something wrong.



Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

All of these SPWs have a lot of noise. This is because of the increased spectral resolution. (line windows, not continuum windows)

Let's redo the bandpass calibration, with smoothing.

Redoing Bandpass Calibration

```
bandpass(vis = 'uid___A002_Xdf0444_X10bd.ms.split',  
         caltable = 'uid___A002_Xdf0444_X10bd.ms.split.bandpass',  
         field = '0', # J1924-2914  
         scan = '3',  
         solint = 'inf',  
         combine = 'scan',  
         refant = 'CM12',  
         solnorm = True,  
         bandtype = 'B',  
         gaintable = 'uid___A002_Xdf0444_X10bd.ms.split.ap_pre_bandpass')
```

Mistake in the script!!!

```
plotbandpass(  
  caltable="uid___A002_Xdf0444_X10bd.ms.split.bandpass_smooth20ch",  
  xaxis="chan",  
  yaxis="both",  
  subplot=42)
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

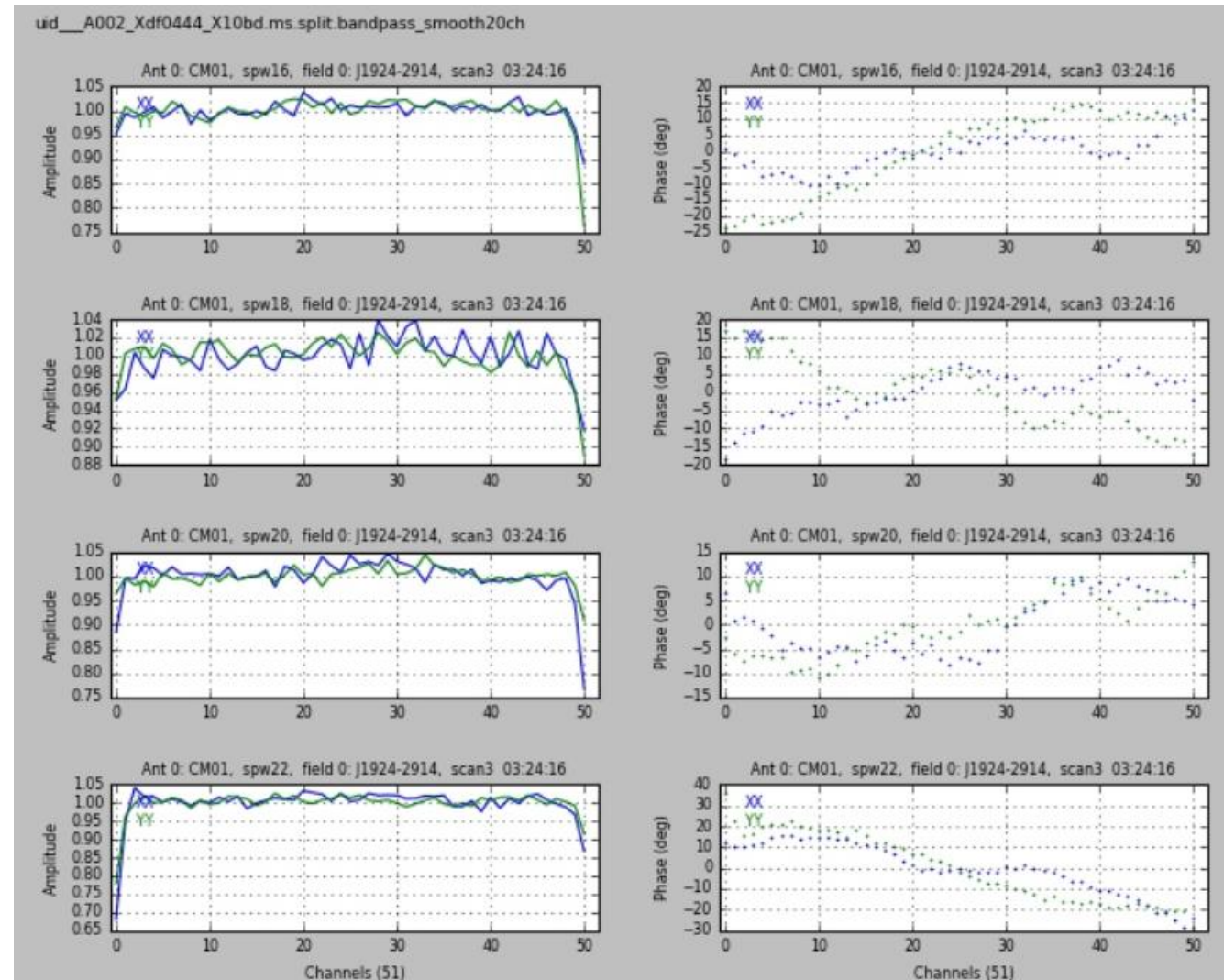
Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Use plotbandpass to plot the new table.



Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

To check the bandpass solution, we need to apply the corrections.

```
applycal(vis="uid__A002_Xdf0444_X10bd.ms.split",  
         field="0",  
         gaintable=["uid__A002_Xdf0444_X10bd.ms.split.bandpass_smooth20ch",  
                  "uid__A002_Xdf0444_X10bd.ms.split.ap_pre_bandpass"],  
         interp=["linear","linear"],  
         gainfield=["0","0"],  
         applymode='calonly')
```

We will talk about this command while it runs!

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

To check the bandpass solution, we need to apply the corrections.

```
applycal(vis="uid__A002_Xdf0444_X10bd.ms.split",  
         field="0",  
         gaintable=["uid__A002_Xdf0444_X10bd.ms.split.bandpass_smooth20ch",  
                  "uid__A002_Xdf0444_X10bd.ms.split.ap_pre_bandpass"],  
         interp=["linear","linear"],  
         gainfield=["0","0"],  
         applymode='calonly')
```

We will talk about this command while it runs!

Field is telling applycal to only apply to the bandpass calibrator.

Gaintable is the list of calibration tables we want to use.

Interp is telling what type of interpolation we use for each table.

Gainfield tells what source solution we will apply.

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Open plotms.

- Select "uid___A002_Xdf0444_X10bd.ms.split,
- Plot phase on the Y axis and use the data column for phase.
- Plot channel on the X axis.
- Select the bandpass calibrator (field = 0), spw=16, and Average over the whole scan.
- Color by correlation.

Make some notes about what you see. Then switch the phase column from data to corrected. Remake the plot, and check the results.

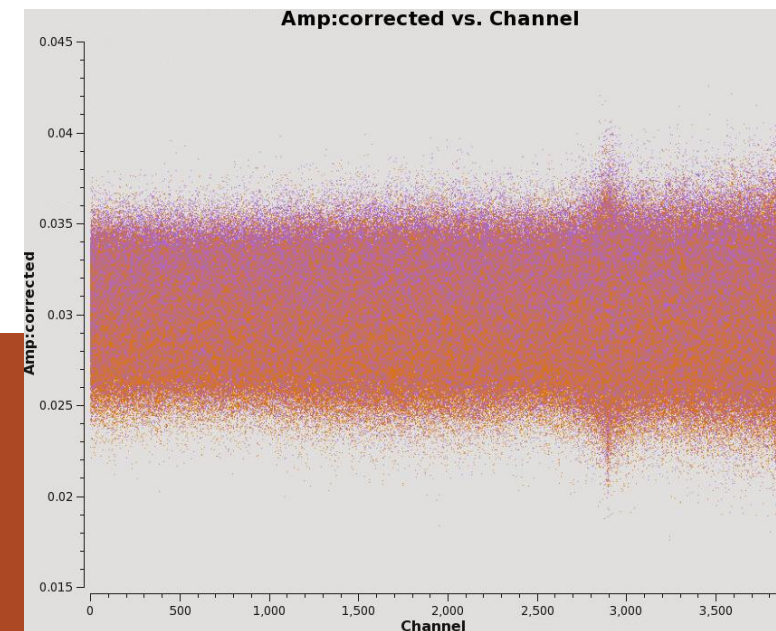
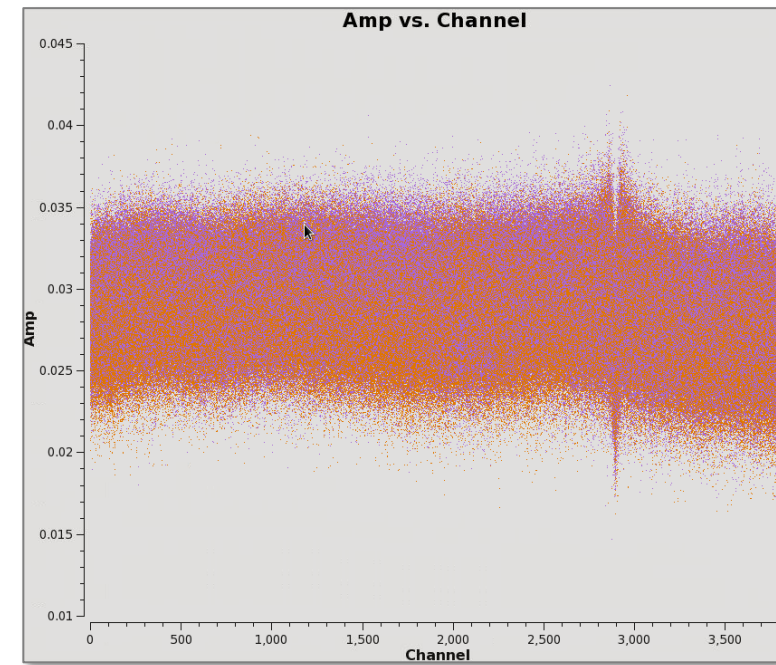
Remake the plot again using amplitude instead of phase. Compare the data column to the corrected column data.

Question: Bandpass

- What did the bandpass calibration do to the data?
- Are there any features you are concerned about?

Question: Bandpass

- What about this bandpass?
- If you saw this, would you be concerned?



Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

There is a small mistake in the script comments.

```
##### Our flux reference source is J0854+2006 (field 1). So we use the results of aU.getALMAFluxForMS for field 1 in the following setjy command."
```

Should read

```
##### Our flux reference source is J1924-2914 (field 0). So we use the results of aU.getALMAFluxForMS for field 0 in the following setjy command."
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

After successfully determining the bandpass solutions, we want to set the model for our flux calibrator. Our flux reference source is a quasar J1924-2914 (field 0).

We use a routine to parse the ALMA calibrator database, interpolate the expected flux for the calibrator reference, and put in the 'model' column of the data using `setjy`.

```
aU.getALMAFluxForMS('uid___A002_Xdf0444_X10bd.ms.split')
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Now run setjy to set the model.

```
setjy(vis = 'uid__A002_Xdf0444_X10bd.ms.split',  
      standard = 'manual',  
      field = 'J1924-2914',  
      usescratch = True,  
      fluxdensity = [2.2942105694549557, 0, 0, 0],  
      spix = -0.5846359878009907,  
      reffreq = '340.9333355239418GHz')
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Gain Calibration: Long-term phase solutions

First, we calibrate the phase for each antenna for each scan. This is the right cadence to transfer to the science target, which is visited only on a every-other-scan timescale.

```
gaincal(vis = 'uid___A002_Xdf0444_X10bd.ms.split',  
        caltable = 'uid___A002_Xdf0444_X10bd.ms.split.phase_inf',  
        field = '0~1', # J1924-2914,J1700-2610  
        solint = 'inf',  
        refant = 'CM12',  
        gaintype = 'G',  
        calmode = 'p',  
        gaintable = 'uid___A002_Xdf0444_X10bd.ms.split.bandpass_smooth20ch')
```

Question: Gain Calibrators

- What makes a good gain (phase) calibrator?

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

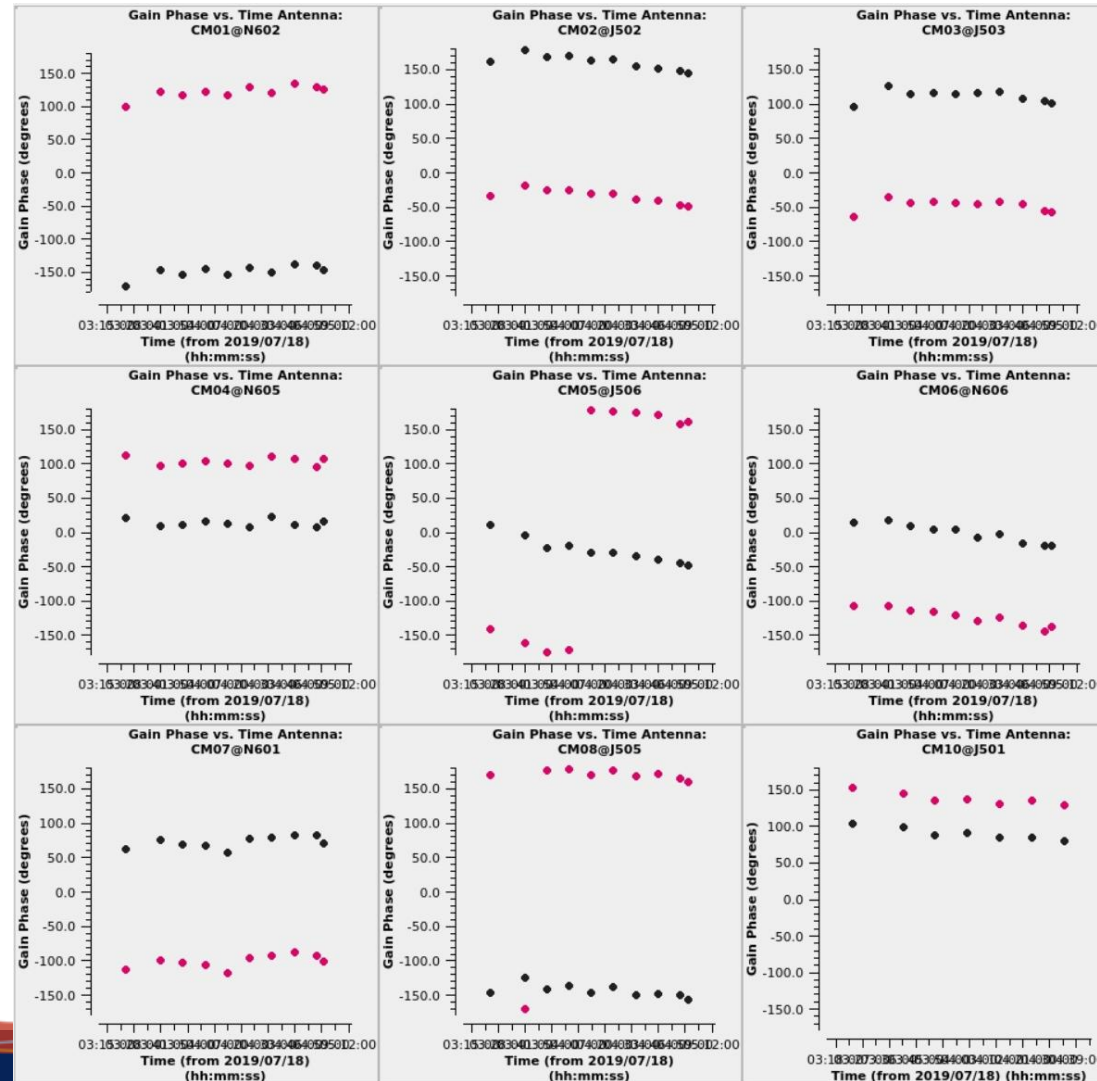
Renormalization Correction

Applying the Calibration

Checking the Calibration

Plot the resulting phase calibration

Use the plotms commands in calibration.py



Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Gain Calibration: Short-term Phase Solutions

Now we want to remove any short timescale phase variation from the sources involved in the bandpass and flux calibration. We do so using `gaincal`.

```
gaincal(vis = 'uid__A002_Xdf0444_X10bd.ms.split',  
        caltable = 'uid__A002_Xdf0444_X10bd.ms.split.phase_int',  
        field = '0~1', # J1924-2914,J1700-2610  
        solint = 'int',  
        refant = 'CM12',  
        gaintype = 'G',  
        calmode = 'p',  
        gaintable = 'uid__A002_Xdf0444_X10bd.ms.split.bandpass_smooth20ch')
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

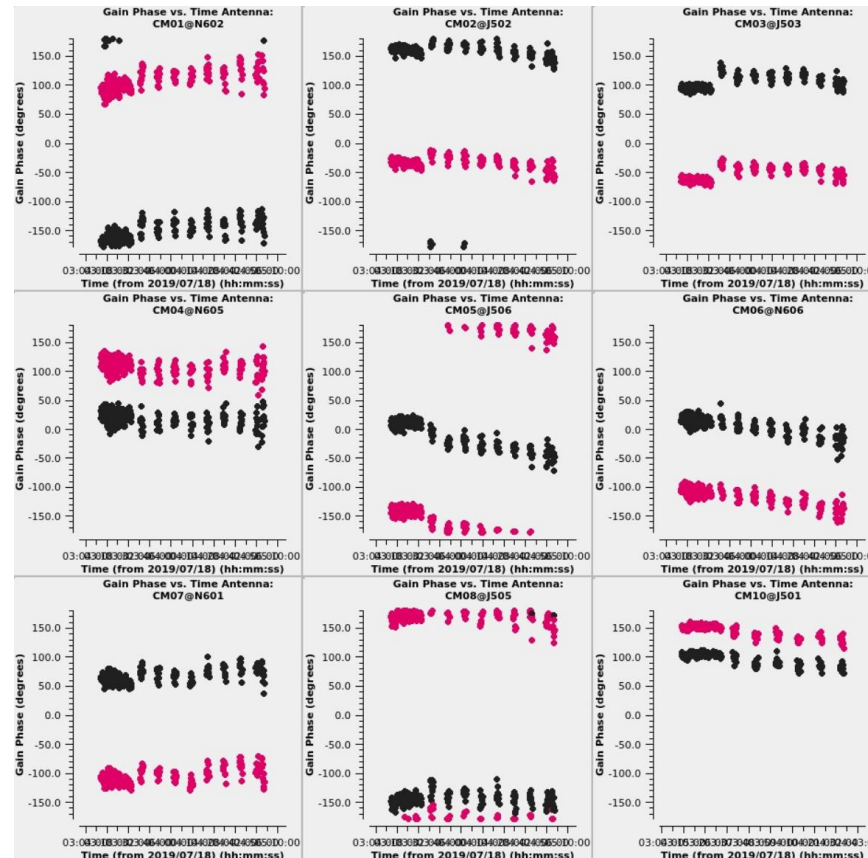
Renormalization Correction

Applying the Calibration

Checking the Calibration

Plot the resulting short timescale phase calibration.

Just like before, you need to run the four plotms commands from your calibration.py script.



Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Gain Calibration: Amplitude Solutions

To create long term amplitude corrections, we will use the `calmode = 'a'` option in `gaincal`.

```
gaincal(vis = 'uid__A002_Xdf0444_X10bd.ms.split',  
        caltable = 'uid__A002_Xdf0444_X10bd.ms.split.ampli_inf',  
        field = '0~1', # J1924-2914,J1700-2610  
        solint = 'inf',  
        refant = 'CM12',  
        gaintype = 'T',  
        calmode = 'a',  
        gaintable =  
        ['uid__A002_Xdf0444_X10bd.ms.split.bandpass_smooth20ch',  
         'uid__A002_Xdf0444_X10bd.ms.split.phase_int'])
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

The gaincal solved for the amplitude scaling to make the data match the current model.

For the quasar J1924-2914, we have taken care to set the correct model using setjy.

For the other two calibrators, however, we don't a priori know the flux.

Those have been calibrated using the default model, which is a point source of amplitude 1 Jy at the middle of the field.

We now use fluxscale to bootstrap from the (correct) flux of the quasar through the amplitude calibration table to estimates of the true flux of the other two calibrators.

This will output both a new table and the flux estimates themselves.

Run the first four lines of the “Setting the flux scale” section in calibration.py

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

```
fluxscaleDict = fluxscale(  
    vis = 'uid__A002_Xdf0444_X10bd.ms.split',  
    caltable = 'uid__A002_Xdf0444_X10bd.ms.split.ampli_inf',  
    fluxtable = 'uid__A002_Xdf0444_X10bd.ms.split.flux_inf',  
    reference = '0') # J1924-2914
```

Set the log back to the CASA logger.

```
casalog.setlogfile(mylogfile)
```

Afterwards, plot the solutions with plotms.

```
plotms(vis="uid__A002_Xdf0444_X10bd.ms.split.flux_inf",  
    xaxis="time",  
    yaxis="amp",  
    gridcols=3,  
    gridrows=3,  
    iteraxis="antenna",  
    symbolsize=10,  
    plotfile="ss20_flux_scan.png")
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

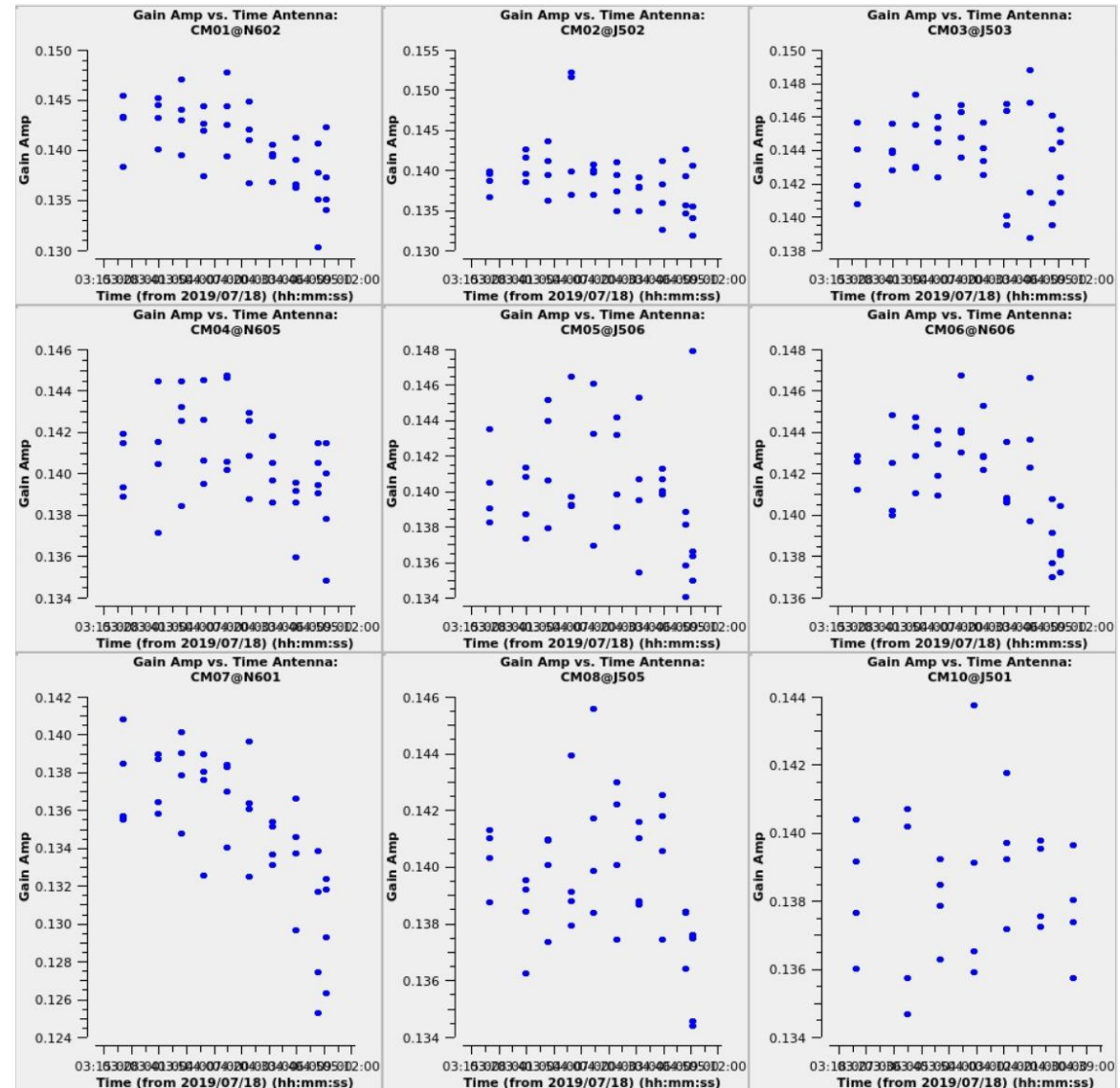
Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration



Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Change applyRenorm = False and run.

```
#####
##### Applying Renorm #####
#####

### This dataset needs renorm. A detailed description about renormalization
#and how this process is carried out can be found at the following link:
#https://help.almascience.org/kb/articles/what-are-the-amplitude-calibration-issues-caused-by-alma-s-normalization-strategy.

os.system('rm -rf uid__A002_Xdf0444_X10bd.ms.split.renorm.tbl')

from pipeline.extern.almarenorm import ACReNorm

# QA2 analyst: First run this step individually with the following variable *applyRenorm* set to False
#               in order to determine the max. renormalization scaling factor.
#               If the factor is <= 1.02, don't change anything and proceed with the remaining steps.
#               If the max. renormalization scaling factor is > 1.02, then edit the following line in this script
#               and set applyRenorm=True, and then re-run this step.
#               In any case, note here the max. renorm factor you determined: <max. renorm factor determined in this step>
applyRenorm = True
applyonly=False

if applyRenorm or applyonly!=True:

    if not applyRenorm: os.system('rm -rf RN_plots')

    rn=ACReNorm('uid__A002_Xdf0444_X10bd.ms.split')

    if applyRenorm or applyonly==True:
        diagSpectra=False
    else:
        diagSpectra=True

    rn.renormalize(createCalTable=applyRenorm, diagSpectra=diagSpectra, antHeuristicsSpectra=False, atmAutoExclude=True, fillTable=True)

    if applyRenorm or applyonly==True:
        pass # no plotting
    else:
        rn.plotSpectra()

if os.path.exists('RN_plots') and applyRenorm:
    os.system('rm -f RN_plots/*.pdf RN_plots/*_scan*_field*.png')
    os.system('mv RN_plots uid__A002_Xdf0444_X10bd.ms.split.renorm.plots')

if not applyRenorm:
    mystats = rn.rnpipestats
    casalog.post('Maximum Renormalization scaling factors:\n'+str(mystats))

    rnfactors = []
    for field in mystats.keys():
        print(' *****')
        print(field, ': max. renorm. scaling for each SPW')
        for spw in mystats[field].keys():
            print(spw, ': ', mystats[field][spw])
            rnfactors.append(mystats[field][spw]['max_rn'])
    maxScaling = max(rnfactors)
    print(' *****')
    print(' The maximum Renormalization scaling is '+str(maxScaling))
    if maxScaling>1.02:
        casalog.post('Renormalization caltable should be created before proceeding.','WARN')
        rn.close()
        sys.exit('Please edit the renorm step to set applyRenorm=True and rerun the step!')
    else:
        print(' No Renormalization needed.')
        print(' *****')

rn.close()
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Renormalization:

A visibility amplitude calibration error that affects fields containing strong line emission

Corrected in affected datasets that have >10% flux offset since Cycle 7.

Knowledgebase Article:

<https://help.almascience.org/kb/articles/what-are-the-amplitude-calibration-issues-caused-by-alma-s-normalization-strategy>

New pipeline stage for pipeline-calibrated datasets

Manually calibrated datasets are checked and corrected before delivery. This dataset does not require renorm correction. However, the script includes a demonstration using the pipeline renorm module

Calibration

- Saving Flagging State
- Flagging Known Bad Data
- Other Flags
- Pre-Bandpass Phase Calibration
- Bandpass Calibration
- Checking Bandpass Calibration
- Apply Flux Calibrator Model to Data
- Gain Calibration
- Setting the Flux Scale
- Renormalization Correction**
- Applying the Calibration
- Checking the Calibration

Traditional scheme

$$c_{ij}(f) [V^2] \xrightarrow{\text{norm}} \frac{c_{ij}(f)}{\sqrt{\langle c_{ii} \rangle_f \langle c_{jj} \rangle_f}} [V^2] \xrightarrow{\text{cal}} c_{ij}(f) [V^2] \xrightarrow{\frac{\sqrt{\langle T_i \rangle_f [K] \cdot \langle T_j \rangle_f [K]}}{\sqrt{\langle c_{ii} \rangle_f \cdot \langle c_{jj} \rangle_f [V^2]}}} c_{ij}(f) [K]$$

Averaged

ALMA scheme

$$c_{ij}(f) [V^2] \xrightarrow{\text{norm}} \frac{c_{ij}(f)}{\sqrt{c_{ii}(f)c_{jj}(f)}} [V^2] \xrightarrow{\text{cal}} c_{ij}(f) [V^2] \xrightarrow{\frac{\sqrt{T_i(f) [K] \cdot T_j(f) [K]}}{\sqrt{c_{ii}(f) \cdot c_{jj}(f) [V^2]}}} c_{ij}(f) [K]$$

Not spectrally averaged

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

$$c_{ij}(f)[V^2] \rightarrow c_{ij}(f)[V^2] \frac{\sqrt{T_i(f)[K] \cdot T_j(f)[K]}}{\sqrt{c_{ii}(f) \cdot c_{jj}(f)[V^2]}} \rightarrow c_{ij}(f)[K]$$

- Both the autocorrelations and the system temperature measurements are a total power-like measurement of the sky.
- If the target source is highly extended and bright, then the source can be picked up in these total power measurements which then impacts this normalization scheme in any channels where the emission was picked up!
- Dividing by the autocorrelations will under-scale the cross-correlations in any affected channels.
- Multiplying by T_{sys} measurement will over-scale the cross-correlations in any affected channels.
- These effects perfectly cancel each other ONLY if they are of the same field.

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

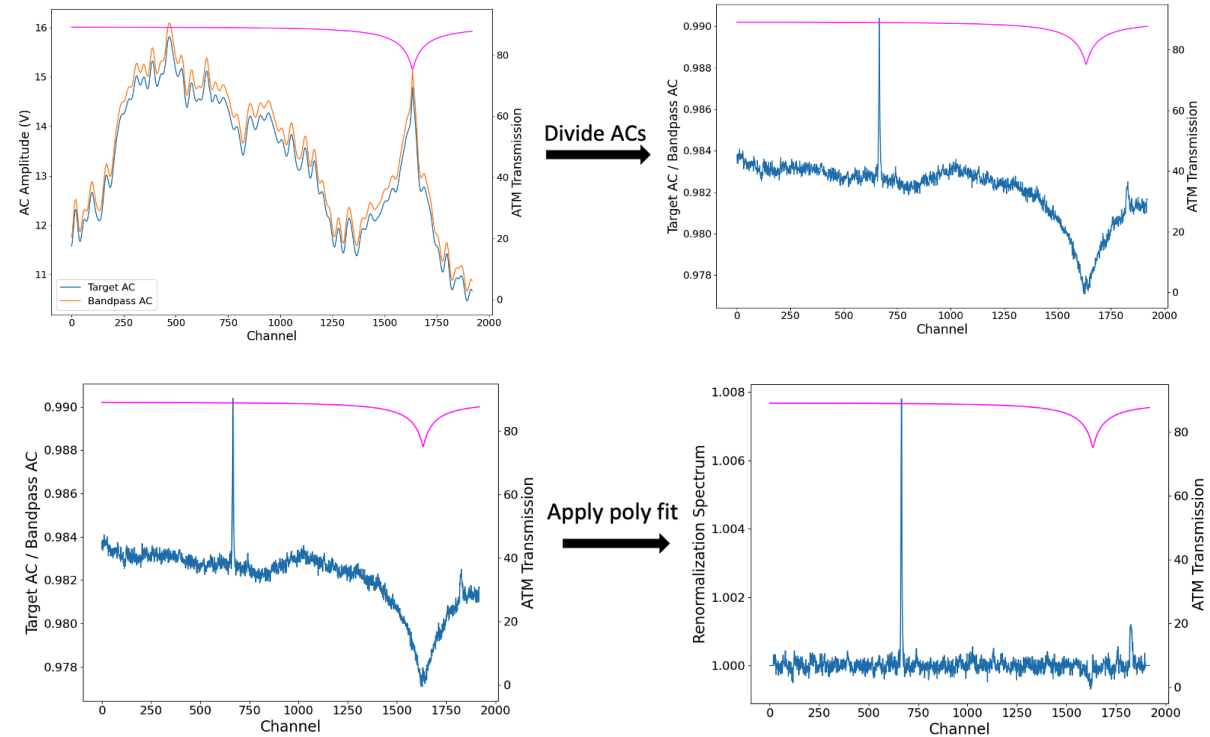
Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration



$$c_{ij}(f) [V^2] \longrightarrow c_{ij}(f) \frac{\sqrt{T_i(f) \cdot T_j(f)}}{\sqrt{c_{ii}(f) \cdot c_{jj}(f)}} \longrightarrow c_{ij}(f) [K] \cdot R_{ij}(f)$$

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

What can change?

```
os.system('rm -rf uid___A002_Xdf0444_X10bd.ms.split.renorm.tbl')
```

- Clearing name of renorm table

```
applyRenorm = True
```

- Do you want to create the table? **Only makes plots if False.**

```
rn=ACreNorm('uid___A002_Xdf0444_X10bd.ms.split')
```

- Name of measurement set

```
os.system('mv RN_plots  
uid___A002_Xdf0444_X10bd.ms.split.renorm.plots')
```

- Moving plot to MS name folder

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

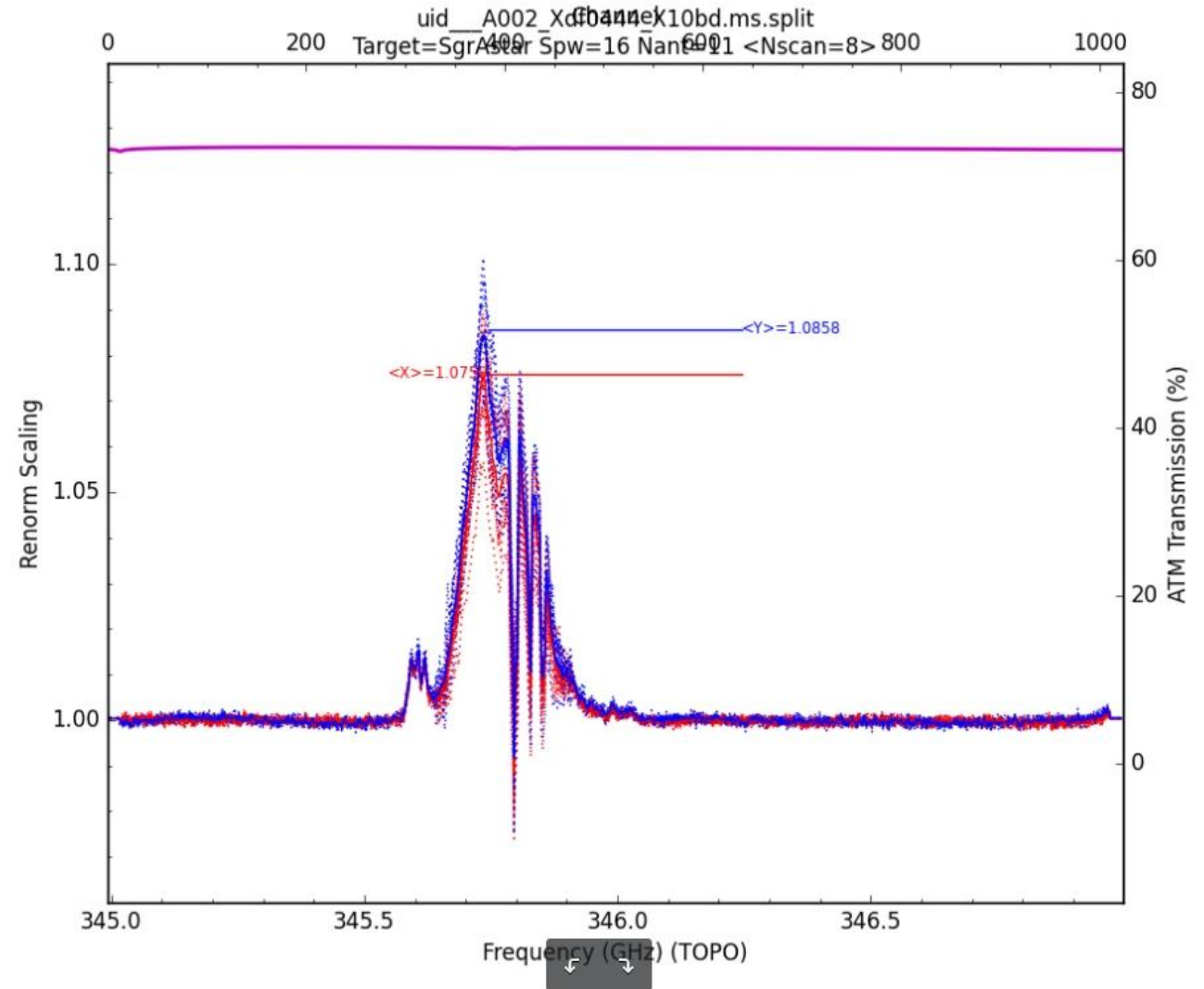
Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration



Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

We do need the table!

Run with `applyRenorm = True` and this will let you create the table.

Calibration is finished!

- We have now created all the calibration tables.
- All we need to do now is apply the calibration.

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

For our bandpass and flux calibrator (fields 0), we apply our bandpass calibration and our gain calibration (short term phase + flux).

For our phase calibrator and science target (fields 1 & 2), we apply our bandpass calibration and our gain calibration (long term phase + flux).

Before applying the calibration, we want to save the flags.

```
flagmanager(vis = 'uid__A002_Xdf0444_X10bd.ms.split',  
            mode = 'save',  
            versionname = 'BeforeApplycal')
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Now we can apply the calibration to the bandpass / flux calibrator:

```
applycal(vis = 'uid__A002_Xdf0444_X10bd.ms.split',  
         field = str(0),  
         gaintable =  
             ['uid__A002_Xdf0444_X10bd.ms.split.bandpass_smooth20ch',  
              'uid__A002_Xdf0444_X10bd.ms.split.phase_int',  
              'uid__A002_Xdf0444_X10bd.ms.split.flux_inf'],  
         gainfield = ['', '0', '0'],  
         interp = 'linear,linear',  
         calwt = True,  
         flagbackup = False)
```

Note, that renorm is not applied to the calibrators, only the target.

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Applying the Calibration

Checking the Calibration

Renormalization Correction

The script has some if statements to make it work with renorm, just in case you did not need it. Lets apply the data to the phase calibrator

```
applycal(vis = 'uid__A002_Xdf0444_X10bd.ms.split',  
         field = '1', # phasecal  
         gaintable =  
         ['uid__A002_Xdf0444_X10bd.ms.split.bandpass_smooth20ch',  
          'uid__A002_Xdf0444_X10bd.ms.split.phase_inf',  
          'uid__A002_Xdf0444_X10bd.ms.split.flux_inf'],  
         gainfield = ['', '1', '1'], # J1700-2610  
         interp = ['linear', 'linear', 'linear'],  
         calwt = True,  
         flagbackup = False)
```

We are interpolating linearly across the solutions from the phase calibrator and applying them to the science field.

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Applying the Calibration

Checking the Calibration

Renormalization Correction

The script has some if statements to make it work with renorm, just in case you did not need it. Lets apply the data to the phase calibrator

```
applycal(vis = 'uid__A002_Xdf0444_X10bd.ms.split',  
         field = '2', # SgrAstar  
         gaintable =  
         ['uid__A002_Xdf0444_X10bd.ms.split.bandpass_smooth20ch',  
          'uid__A002_Xdf0444_X10bd.ms.split.phase_inf',  
          'uid__A002_Xdf0444_X10bd.ms.split.flux_inf',  
          'uid__A002_Xdf0444_X10bd.ms.split.renorm.tbl'],  
         gainfield = ['', '1', '1', ''], # J1700-2610  
         interp = ['linear','linear','linear','linear'],  
         calwt = True,  
         flagbackup = False)
```

We are interpolating linearly across the solutions from the phase calibrator and applying them to the science field.

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

- Next, we will review the basics of data inspection and flagging.
- We will now use plotms to make a series of diagnostic plots. These plots have been picked because we have a good expectation of what the calibrators (fields 0, and 1) should look like in each space.
- Typical plots for calibrated data inspection are the following:
 - Amplitude vs time
 - Phase vs time
 - Amplitude vs frequency
 - Phase vs frequency
 - Amplitude vs UV distance
 - Phase vs UV distance
- Iterate over Antenna, Spectral Window, or source
- Typically, start by plotting the calibrators.
 - Be careful flagging the target, as you could be flagging out real structure on the target.

We will go through the plots in the order they are listed in calibration.py

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration



Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Flagging with plotms

It is possible to flag within the GUI with plotms. This is done by boxing the data with the  button, and then flagging with the  button.

Unflagging can be performed by plotting flagged points and using the  button.

FLAGGING USING PLOTMS IS NOT ENCOURAGED!!!

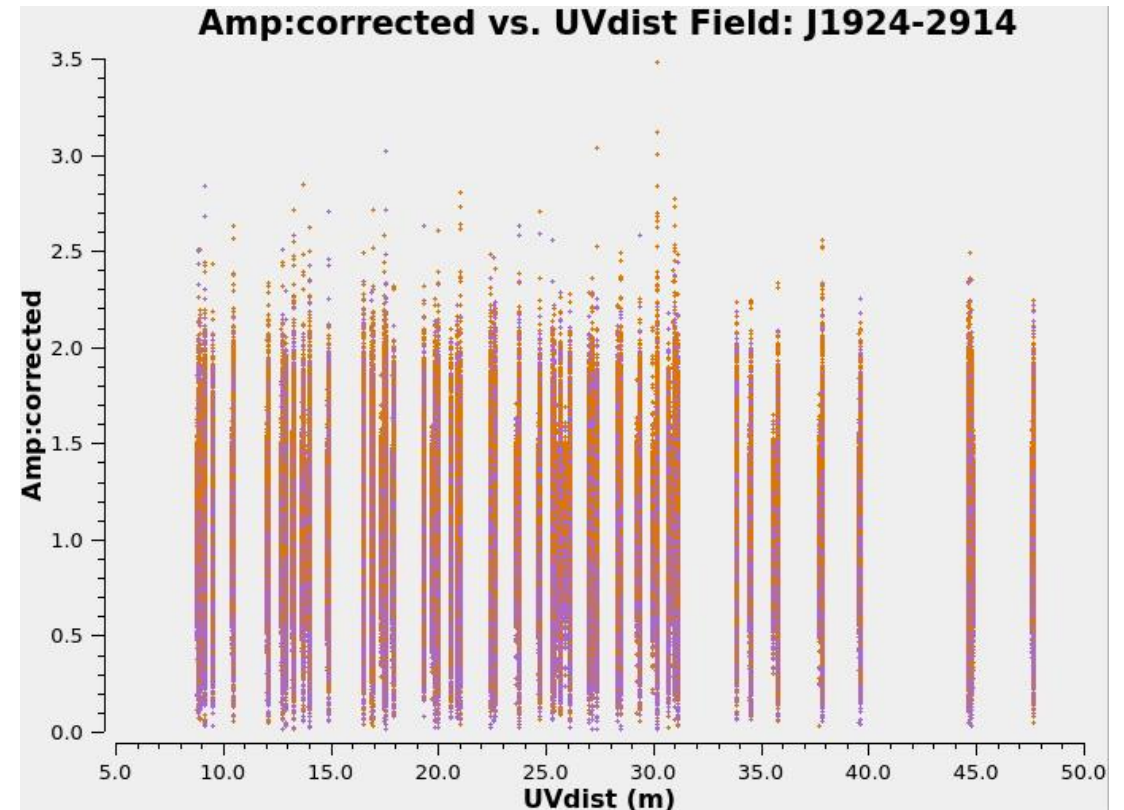
Flagging with plotms is not considered reproduceable. Due to the nature of drawing boxes in plotms, it is difficult to tell what points were flagged.

It is best to use plotms to identify bad data, and then flag using flagdata.

Question: Amp vs UVdist

Look for individual baselines or antennas not following the other baselines.

What do we expect a point source at the phase center to look like in terms of Amplitude vs UV distance?

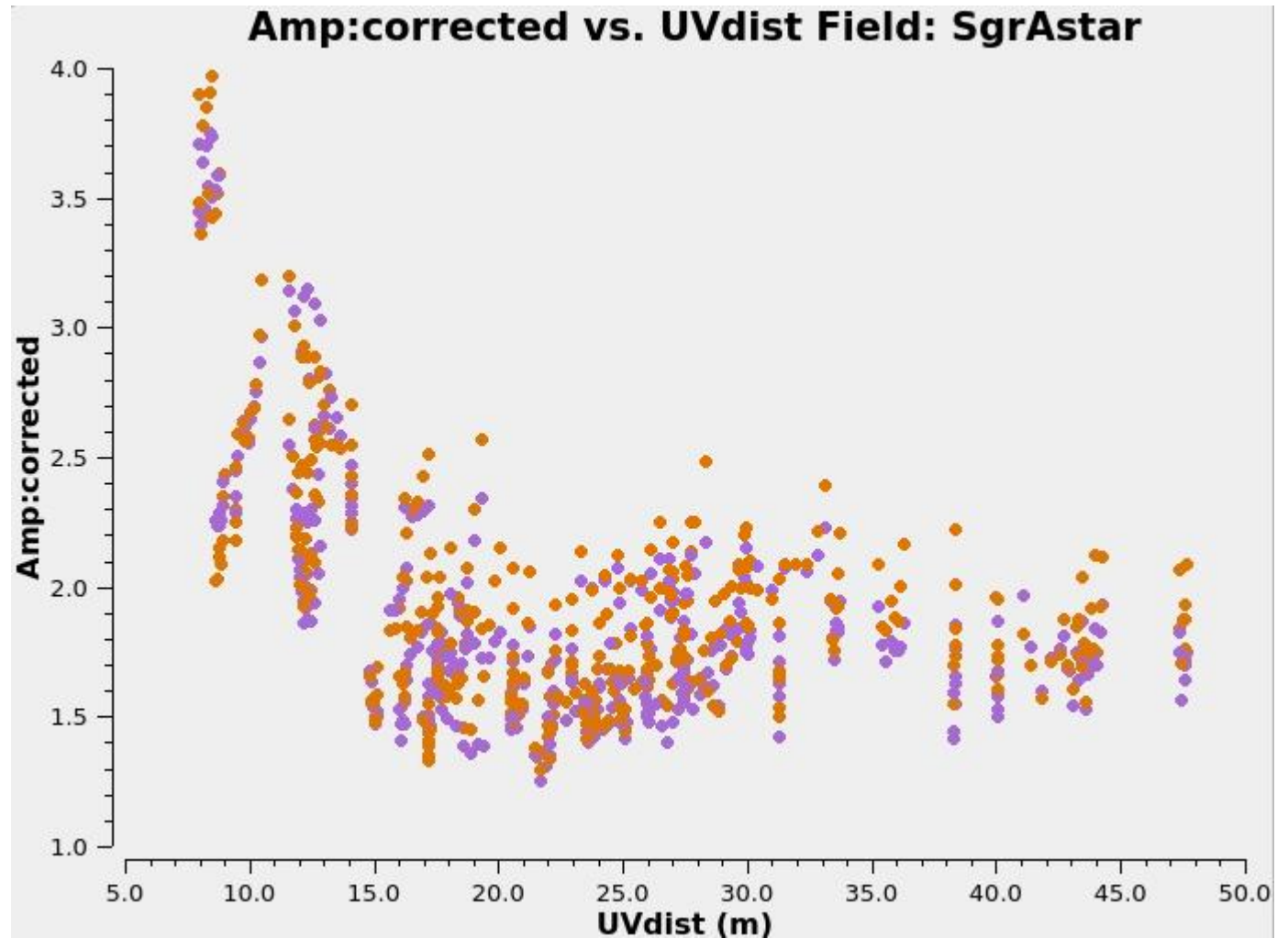


**UV distance vs amplitude for Calibrator
J1924-2914**



Calibration

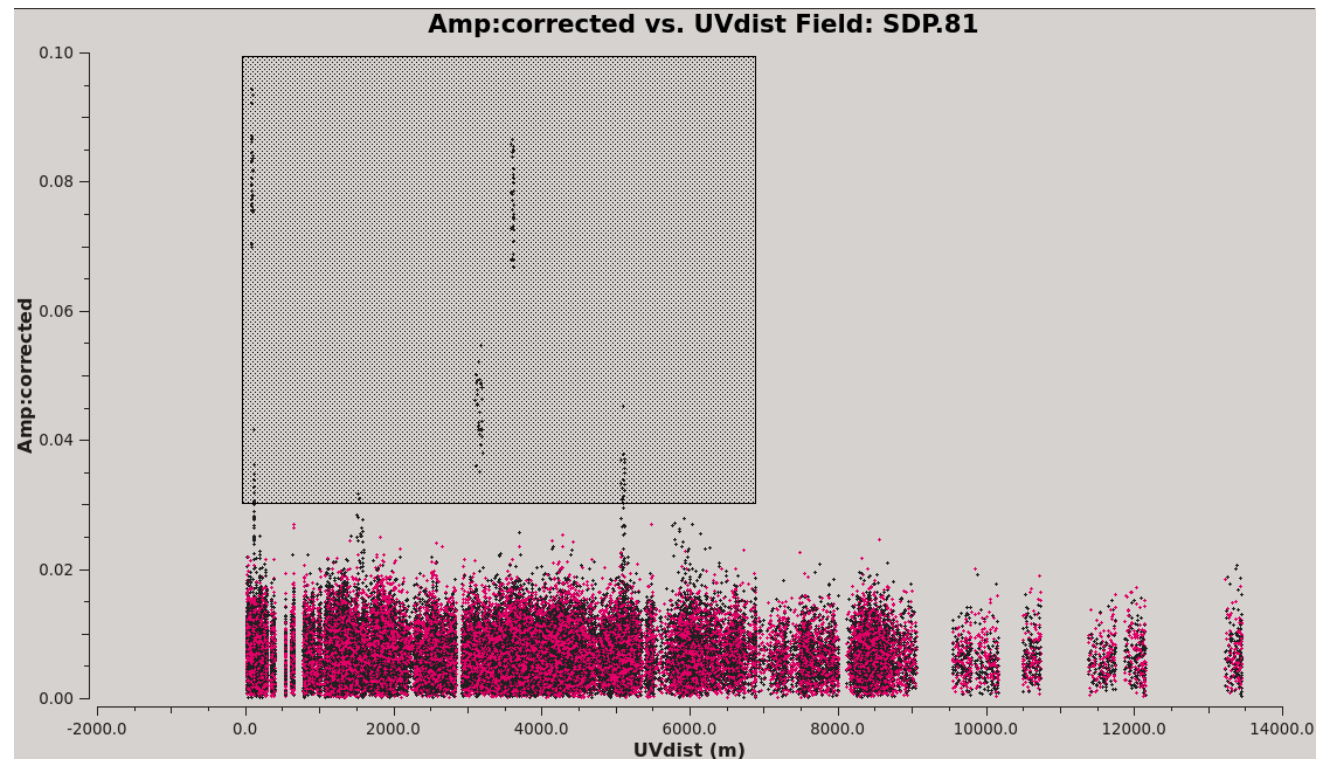
- Saving Flagging State
- Flagging Known Bad Data
- Other Flags
- Pre-Bandpass Phase Calibration
- Bandpass Calibration
- Checking Bandpass Calibration
- Apply Flux Calibrator Model to Data
- Gain Calibration
- Setting the Flux Scale
- Renormalization Correction
- Applying the Calibration
- Checking the Calibration**

Look at all three sources. Do you notice any points that stand out?



Calibration

- Highlight points with  and click locate ()
- What is in common?



```
Scan=59 Field=SDP.81 [3] Time=2014/11/03/10:19:48.1440 BL=DA50@W204 & PM04@T703 [1&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=5119.21 Y=0.0364852 (
Scan=59 Field=SDP.81 [3] Time=2014/11/03/10:19:48.1440 BL=DA62@W206 & PM04@T703 [9&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3608.39 Y=0.0727789 (
Scan=59 Field=SDP.81 [3] Time=2014/11/03/10:19:48.1440 BL=DV10@A024 & PM04@T703 [15&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=99.0455 Y=0.0777865 (
Scan=61 Field=SDP.81 [3] Time=2014/11/03/10:21:05.8080 BL=DA41@S301 & PM04@T703 [0&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3135.01 Y=0.0470909 (
Scan=61 Field=SDP.81 [3] Time=2014/11/03/10:21:05.8080 BL=DA50@W204 & PM04@T703 [1&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=5119.11 Y=0.0355896 (
Scan=61 Field=SDP.81 [3] Time=2014/11/03/10:21:05.8080 BL=DA62@W206 & PM04@T703 [9&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3606.49 Y=0.0845674 (
Scan=61 Field=SDP.81 [3] Time=2014/11/03/10:21:05.8080 BL=DV10@A024 & PM04@T703 [15&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=99.1765 Y=0.0752248 (
Scan=61 Field=SDP.81 [3] Time=2014/11/03/10:21:05.8080 BL=DV21@A004 & PM04@T703 [22&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=119.004 Y=0.0361741 (
Scan=63 Field=SDP.81 [3] Time=2014/11/03/10:22:23.4720 BL=DA41@S301 & PM04@T703 [0&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3131.89 Y=0.0453449 (
Scan=63 Field=SDP.81 [3] Time=2014/11/03/10:22:23.4720 BL=DA62@W206 & PM04@T703 [9&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3604.52 Y=0.0779968 (
Scan=63 Field=SDP.81 [3] Time=2014/11/03/10:22:23.4720 BL=DV10@A024 & PM04@T703 [15&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=99.309 Y=0.0817332 (
Scan=65 Field=SDP.81 [3] Time=2014/11/03/10:23:41.1360 BL=DA41@S301 & PM04@T703 [0&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3128.73 Y=0.0477873 (
Scan=65 Field=SDP.81 [3] Time=2014/11/03/10:23:41.1360 BL=DA50@W204 & PM04@T703 [1&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=5118.47 Y=0.0371042 (
Scan=65 Field=SDP.81 [3] Time=2014/11/03/10:23:41.1360 BL=DA62@W206 & PM04@T703 [9&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3602.45 Y=0.0681095 (
Scan=65 Field=SDP.81 [3] Time=2014/11/03/10:23:41.1360 BL=DV10@A024 & PM04@T703 [15&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=99.4368 Y=0.0755574 (
Scan=67 Field=SDP.81 [3] Time=2014/11/03/10:24:58.8000 BL=DA41@S301 & PM04@T703 [0&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3125.55 Y=0.0501428 (
Scan=67 Field=SDP.81 [3] Time=2014/11/03/10:24:58.8000 BL=DA62@W206 & PM04@T703 [9&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3600.29 Y=0.0858272 (
Scan=67 Field=SDP.81 [3] Time=2014/11/03/10:24:58.8000 BL=DV10@A024 & PM04@T703 [15&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=99.5662 Y=0.0753881 (
Scan=67 Field=SDP.81 [3] Time=2014/11/03/10:24:58.8000 BL=DV21@A004 & PM04@T703 [22&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=118.901 Y=0.0346102 (
Scan=69 Field=SDP.81 [3] Time=2014/11/03/10:26:16.4640 BL=DA41@S301 & PM04@T703 [0&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3122.34 Y=0.0359895 (
Scan=69 Field=SDP.81 [3] Time=2014/11/03/10:26:16.4640 BL=DA62@W206 & PM04@T703 [9&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3598.06 Y=0.0727033 (
Scan=69 Field=SDP.81 [3] Time=2014/11/03/10:26:16.4640 BL=DV10@A024 & PM04@T703 [15&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=99.6965 Y=0.0839206 (
Scan=73 Field=SDP.81 [3] Time=2014/11/03/10:29:08.7360 BL=DA41@S301 & PM04@T703 [0&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3115.13 Y=0.0487719 (
Scan=73 Field=SDP.81 [3] Time=2014/11/03/10:29:08.7360 BL=DA62@W206 & PM04@T703 [9&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3592.79 Y=0.0783596 (
Scan=73 Field=SDP.81 [3] Time=2014/11/03/10:29:08.7360 BL=DV10@A024 & PM04@T703 [15&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=99.9755 Y=0.0934339 (
Scan=75 Field=SDP.81 [3] Time=2014/11/03/10:30:26.4000 BL=DA41@S301 & PM04@T703 [0&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3111.84 Y=0.0460959 (
Scan=75 Field=SDP.81 [3] Time=2014/11/03/10:30:26.4000 BL=DA62@W206 & PM04@T703 [9&24] Spw=2 Chan=<0~63> Avg Freq=156.444 Corr=XX X=3590.28 Y=0.0677774 (
```

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Example flagging command (Not specific to our current dataset)

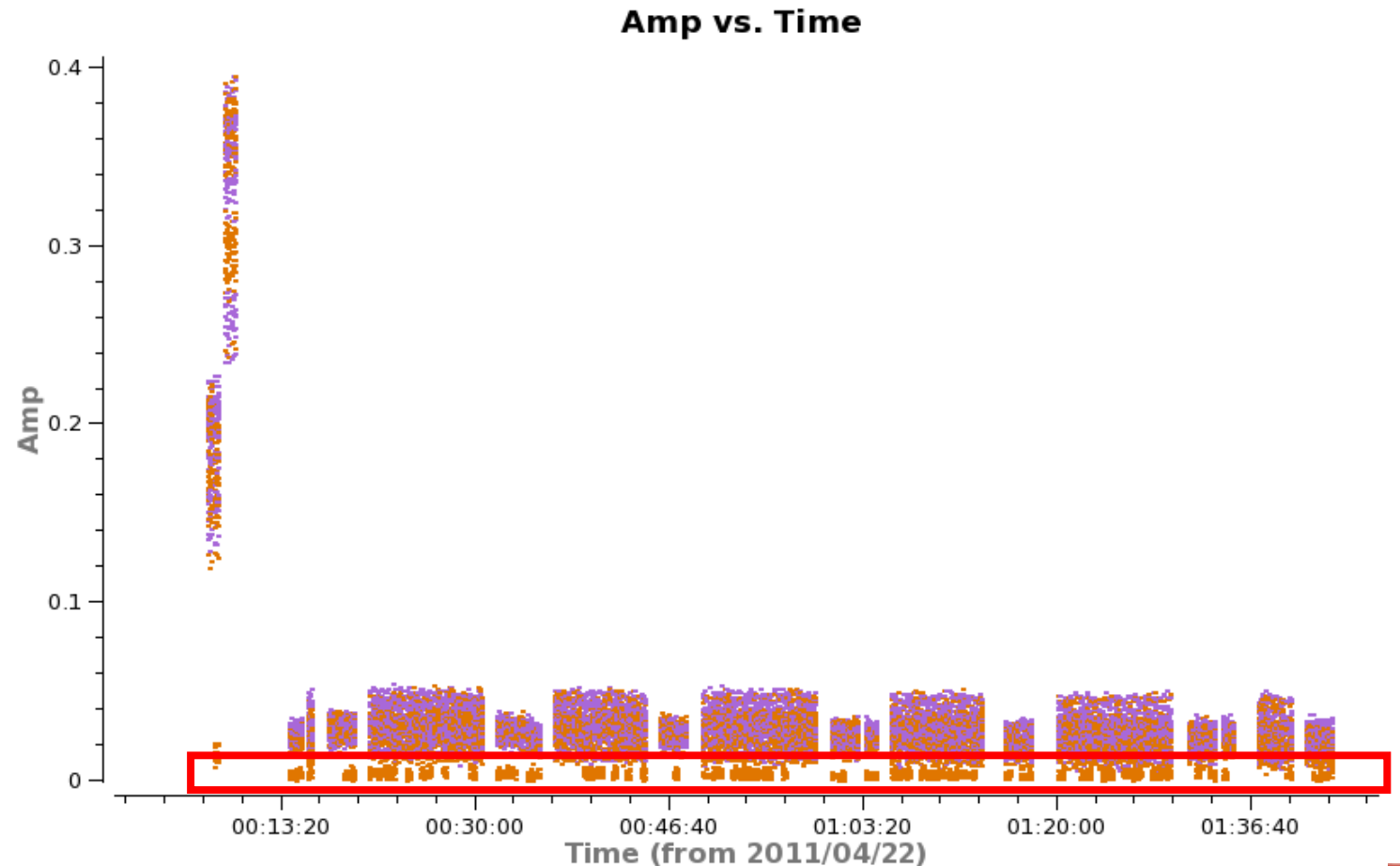
```
## Outlier antenna in spw2  
flagdata(vis = 'ms.split',  
         mode = 'manual',  
         spw = '2',  
         antenna= 'PM04',  
         flagbackup = False)
```

Calibration

- Saving Flagging State
- Flagging Known Bad Data
- Other Flags
- Pre-Bandpass Phase Calibration
- Bandpass Calibration
- Checking Bandpass Calibration
- Apply Flux Calibrator Model to Data
- Gain Calibration
- Setting the Flux Scale
- Renormalization Correction
- Applying the Calibration
- Checking the Calibration**

Amplitude vs time: Look for scans that are different than other scans of the same target.

Look for outliers (very low or very high antennas or baselines).



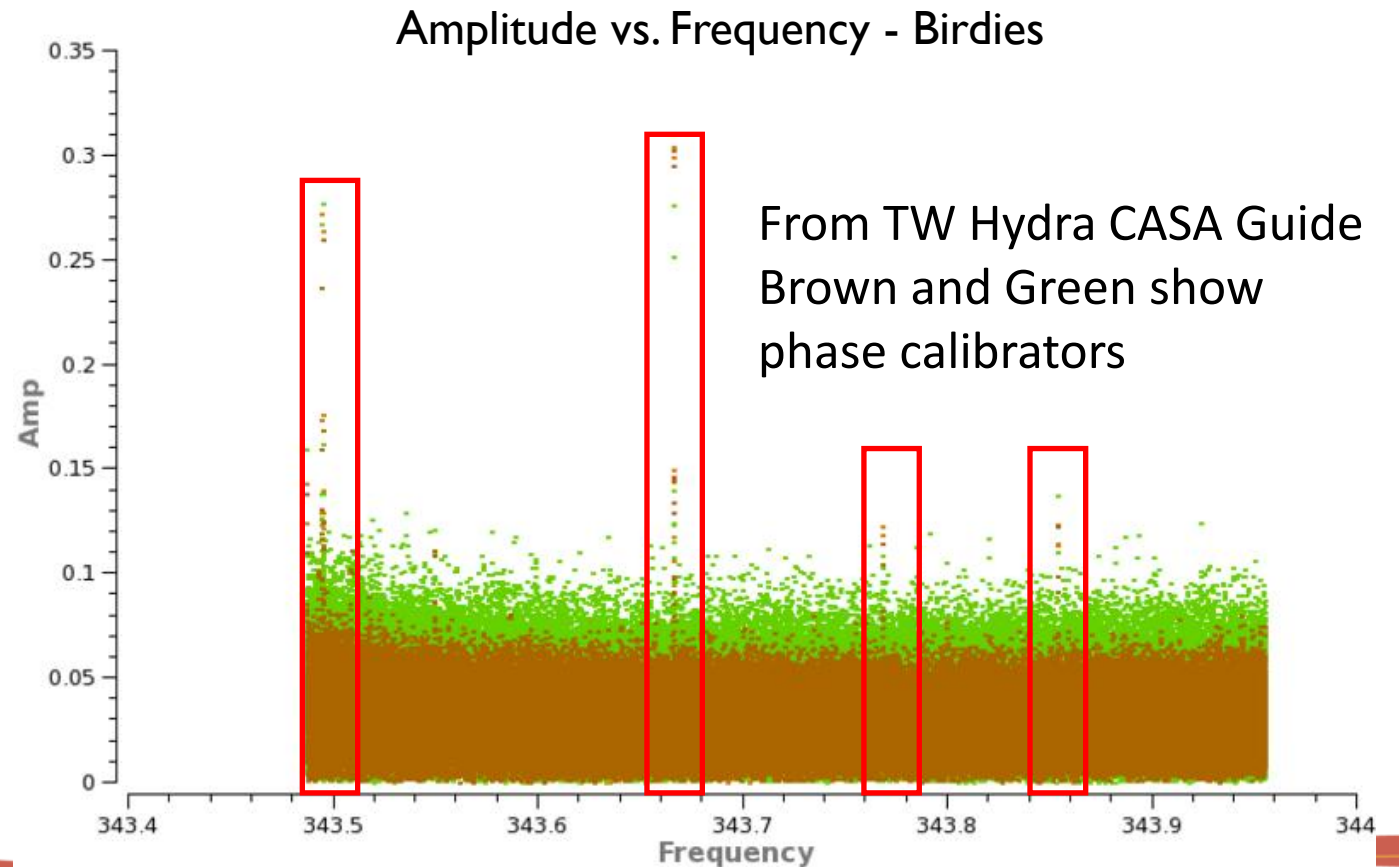
Calibration

- Saving Flagging State
- Flagging Known Bad Data
- Other Flags
- Pre-Bandpass Phase Calibration
- Bandpass Calibration
- Checking Bandpass Calibration
- Apply Flux Calibrator Model to Data
- Gain Calibration
- Setting the Flux Scale
- Renormalization Correction
- Applying the Calibration
- Checking the Calibration

For Amplitude vs frequency (or channel):

Look for bad edge channels, birdies, and other effects **IN THE CALIBRATORS!**

The target may have spectral lines that can be real. Gain calibrators can also sometimes have native lines. Lines don't need to be flagged from the gain calibrator.

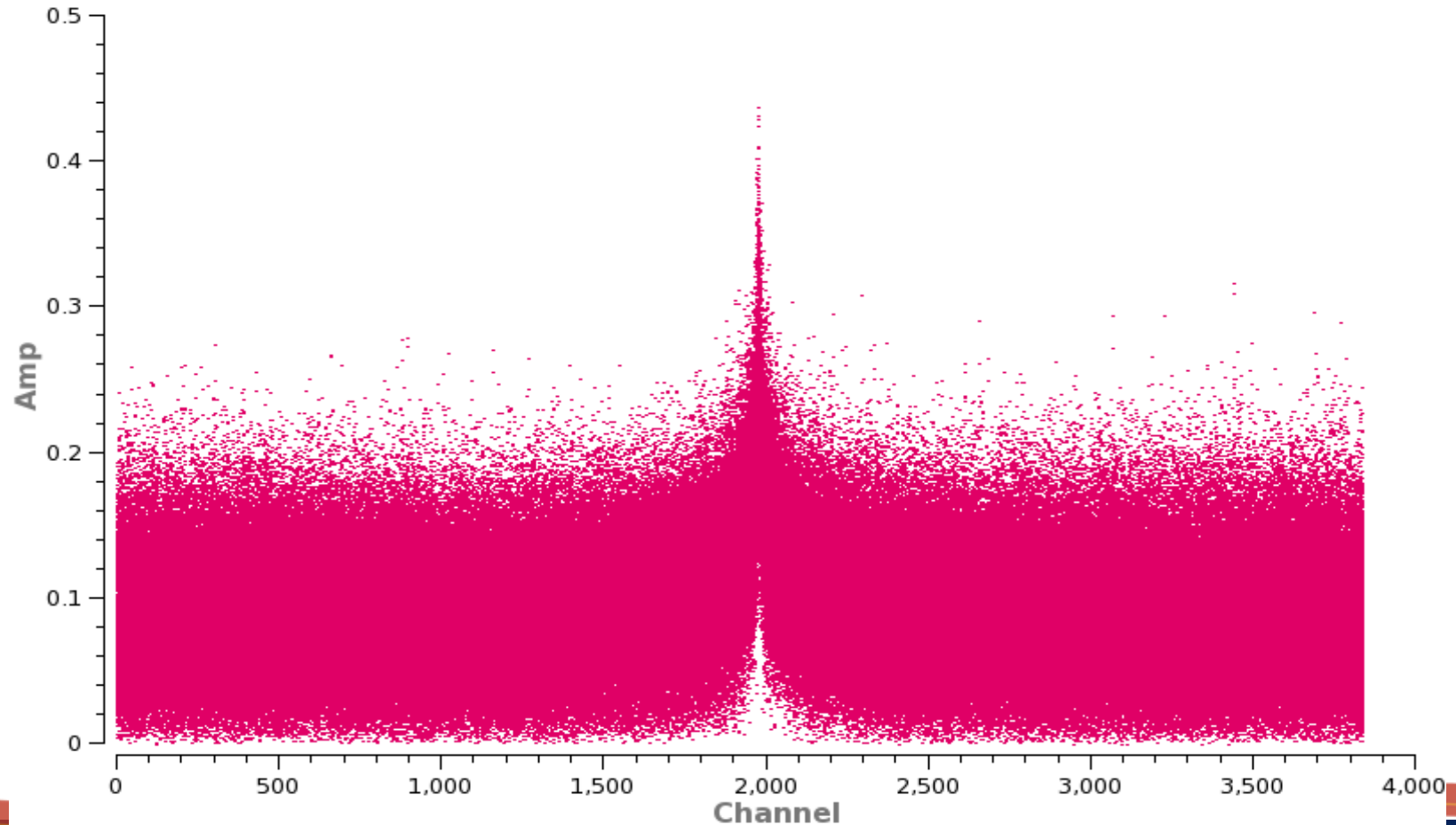


Calibration

- Saving Flagging State
- Flagging Known Bad Data
- Other Flags
- Pre-Bandpass Phase Calibration
- Bandpass Calibration
- Checking Bandpass Calibration
- Apply Flux Calibrator Model to Data
- Gain Calibration
- Setting the Flux Scale
- Renormalization Correction
- Applying the Calibration
- Checking the Calibration**

Spectral lines in flux calibrators should be flagged.

From TW Hydra Band 7 Guide
Spectral line in Titan (Flux Calibrator)
Amp vs. Channel Spw: 3

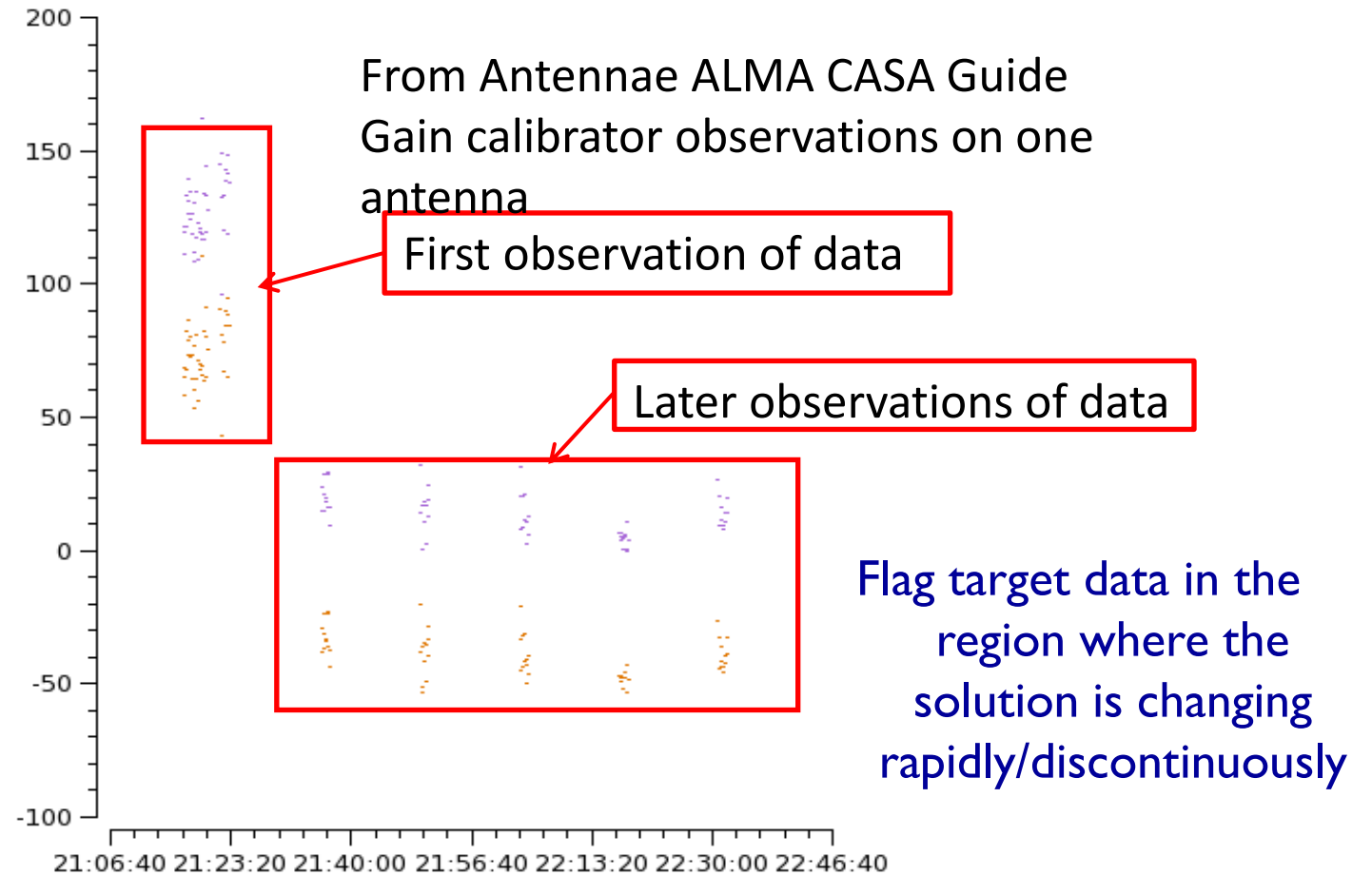


Calibration

- Saving Flagging State
- Flagging Known Bad Data
- Other Flags
- Pre-Bandpass Phase Calibration
- Bandpass Calibration
- Checking Bandpass Calibration
- Apply Flux Calibrator Model to Data
- Gain Calibration
- Setting the Flux Scale
- Renormalization Correction
- Applying the Calibration
- Checking the Calibration**

Phase vs time and phase vs frequency:

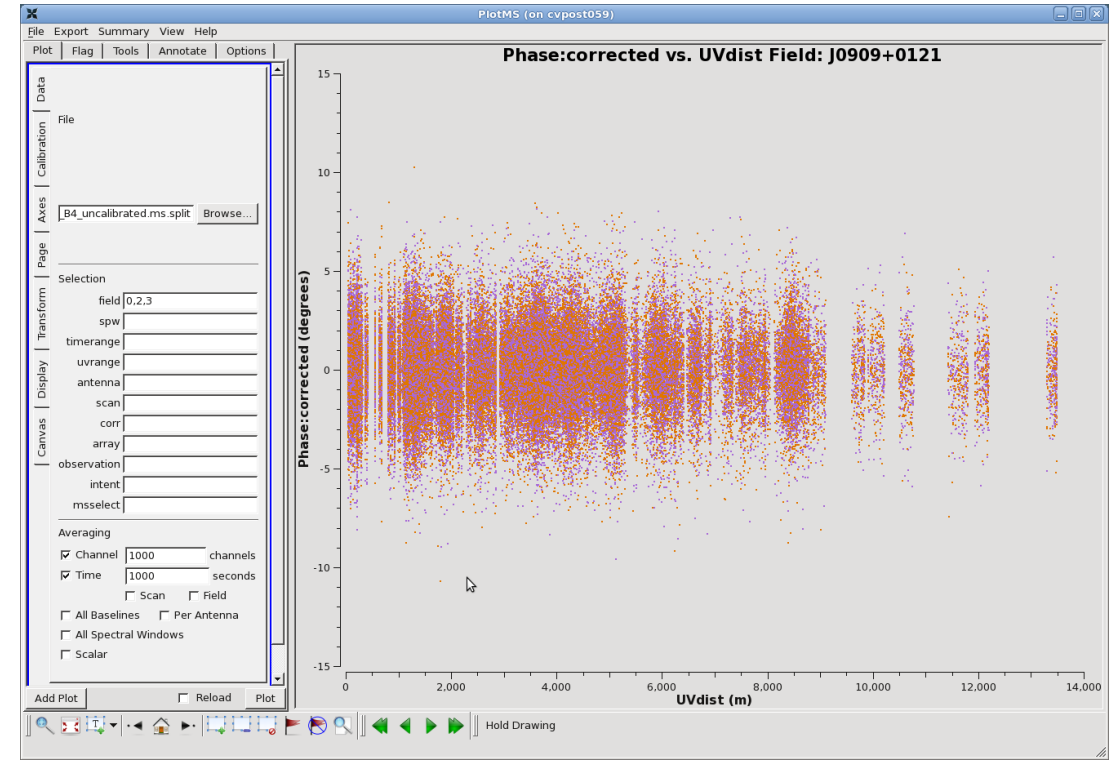
Look for discontinuities and where the phase changes rapidly. Increased phase noise is normal around atmospheric lines.



Your turn:

Look at the plots you did not look at yet:

- Amplitude vs time
- Phase vs time
- Amplitude vs frequency
- Phase vs frequency



Phase vs UV distance for J0909+0121

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

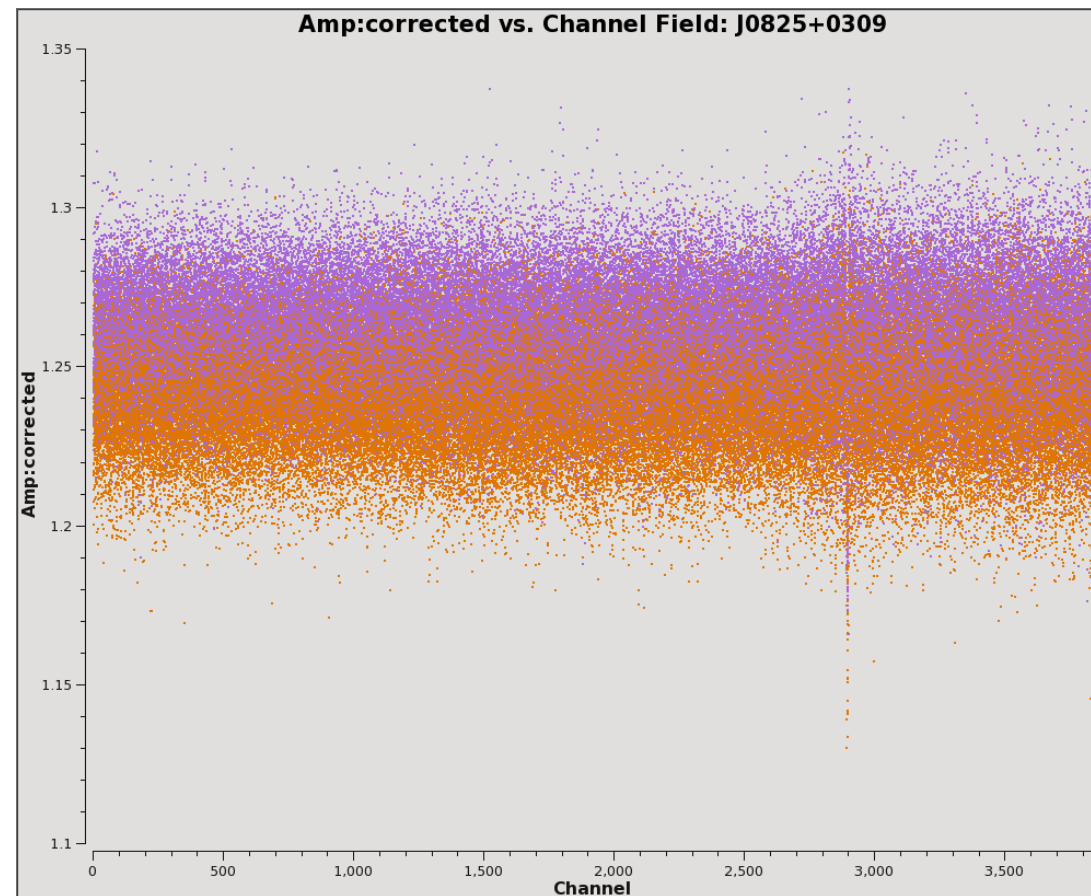
Renormalization Correction

Applying the Calibration

Checking the Calibration

Remember: We do not flag telluric lines in the data.

After writing out all the flag commands, re-run the entire calibration!



Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Final Steps:

After you finish

1. Calibrating the data
2. Inspecting the data
3. Adding flags
4. Repeat steps 1-3 as needed.

When finished, it is time to split the corrected columns to a new dataset.

Calibration

Saving Flagging State

Flagging Known Bad Data

Other Flags

Pre-Bandpass Phase Calibration

Bandpass Calibration

Checking Bandpass Calibration

Apply Flux Calibrator Model to Data

Gain Calibration

Setting the Flux Scale

Renormalization Correction

Applying the Calibration

Checking the Calibration

Normally, we would now repeat this step for all other executions in the MOUS.

For Sgr A*, there were a total of 4 observations.

To save time, we have done the other 3 executions for you.

For imaging (next workshop or on your own) you can either image your single EB, or the full 4 EB MOUS.

And we are finished!

- Thanks for sticking with me though that long workshop!
- If you have any further questions, feel free to ask!
- If you have problems or need to ask questions later, contact us at the ALMA help desk! We are happy to help with your technical questions!

Additional Resources:

- ALMA CASA Guides:
 - <https://casaguides.nrao.edu/index.php?title=ALMAGuides>
- ALMA Documentation
 - <https://almascience.nrao.edu/documents-and-tools>
- ALMA Helpdesk:
 - <https://help.almascience.org/>

Thank You!

