e-MERLIN Data School

This event has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004719

Welcome!

Dave Williams-Baldwin

Emmanuel Bempong-Manful

Justin Bray

Code of Conduct

- This workshop will be running under the auspices of the Opticon RadioNet Pilot (ORP) Code of Conduct, which can be found here: https://www.e-merlin.ac.uk/ORP_Code_of_Conduct.pdf
- All members of the workshop are expected to treat each other equally and with repect, regardless of gender, sexual orientation, gender identity, race, ethnicity, national origin, physical disability, religion, age or any other attribute.
- We do not tolerate any form of bullying, discrimination, verbal, non-verbal or physical harassment, racism, retaliation, threatening behaviour, or any other inappropriate conduct. Members must be aware that behaviours and language acceptable to one person may not be to another and therefore are expected to make every effort to ensure that words and actions communicate respect for one another.

Proposed schedule

	Monday	Tuesday	Wednesday
10:00-11:00	School Overview + Radio 101	**Calibration Overview**	Optical talk and Q+A
11:00-11:15	Coffee	Coffee	Coffee
11:15-12:30	Radio 101 + CASA demo	**Hands on Session**	e-MERLIN Proposal guide
12:30-13:30	Lunch	Lunch	Lunch
13:30-14:45	**eMCP Overview**	**Imaging**	**Advanced imaging**
14:45-15:00	Coffee	Coffee	Coffee
15:00-16:00	**Hands on Session**	**Hands on Session**	**Hands on Session**

Bold sessions include hands on help

Housekeeping

- Data school website: https://www.e-merlin.ac.uk/eMER_data_school_2023.html
- Dataset used: https://www.e-merlin.ac.uk/distribute/CY8/TS8004/TS8004.html
  - Or use your own data!
- ERIS Radio interferometry lectures: https://www.jive.eu/ERIS2022/lectures.php
- ERIS Radio calibration of the same dataset: https://www.jb.man.ac.uk/DARA/ERIS22/3C277_full.html

Getting Started

Log ins

- You should have a log in under the username "wshop#" where # is a number
- A password is provided which will get you access to that machine
- We will use the same machines all week
- Data that has been requested has been downloaded onto these machines, so you can use those datasets, or, use the training dataset for TS8004.

Where is the working area?

- The working area is:

/mirror1/scratch/DataSchool1 (for botham/wshop1)

/mirror2/scratch/DataSchool2 (for richards/wshop2)

/raid/scratch/DataSchool# (for all other logins)

- The # refers to the wshop username number, i.e 3-7
- These will be labelled as the */workingarea/* for this workshop

- You can find the datasets in:

/workingarea/Data for all logins

- For this tutorial, we will use CASA 5.8, which we have downloaded to:

/workingarea/casa-release-5.8.0-109.el6 for all logins

- You can run CASA by using:

/workingarea/casa-release-5.8.0-109.el6/bin/casa for all logins

Basic CASA syntax

- CASA works like python, and has different commands that you can call using:

  default listobs

- You can find out the inputs for a task with:

  inp listobs

- If you need help with a taks:

  help listobs

- And start a task with:

  go listobs

```
CASA <4>: inp listobs
--------> inp(listobs)
#  listobs :: List the summary of a data set in the logger or
vis                 =         ''        #  Name of input visi
selectdata          =       True        #  Data selection par
     field          =         ''        #  Selection based on
 index numbers. Default is all.
     spw            =         ''        #  Selection based on
uency/channel.
     antenna        =         ''        #  Selection based on
efault is all.
     timerange      =         ''        #  Selection based on
is entire range.
     correlation    =         ''        #  Selection based on
 is all.
     scan           =         ''        #  Selection based on
t is all.
     intent         =         ''        #  Selection based on
Default is all.
     feed           =         ''        #  Selection based on
```

Basic CASA syntax

- You can either type in the parameters per task like:

  default listobs

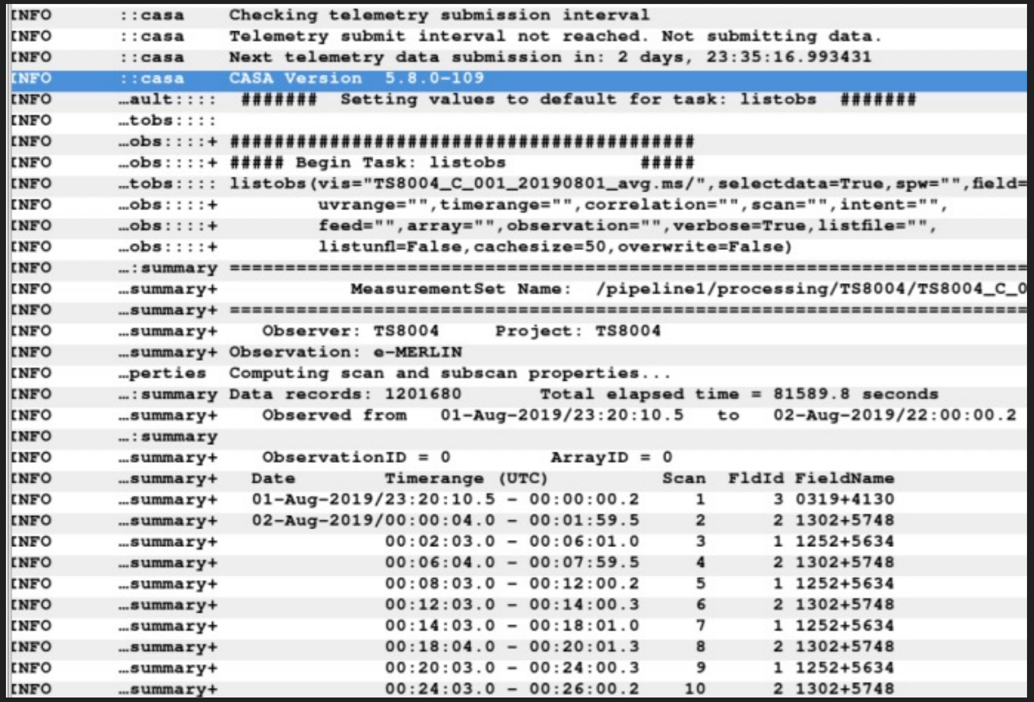
  vis='TS8004_C_001_20190801_avg.ms/'

  go listobs

- Or write it in a single line:

  listobs(vis='TS8004_C_001_20190801_avg.ms/')

- And the logs will be output into the logger



```
INFO   ::casa      Checking telemetry submission interval
INFO   ::casa      Telemetry submit interval not reached. Not submitting data.
INFO   ::casa      Next telemetry data submission in: 2 days, 23:35:16.993431
INFO   ::casa      CASA Version  5.8.0-109
INFO   …ault::::   #######  Setting values to default for task: listobs  #######
INFO   …tobs::::
INFO   …obs::::+   ##########################################
INFO   …obs::::+   ##### Begin Task: listobs            #####
INFO   …tobs::::   listobs(vis="TS8004_C_001_20190801_avg.ms/",selectdata=True,spw="",field=
INFO   …obs::::+           uvrange="",timerange="",correlation="",scan="",intent="",
INFO   …obs::::+           feed="",array="",observation="",verbose=True,listfile="",
INFO   …obs::::+           listunfl=False,cachesize=50,overwrite=False)
INFO   …:summary   ================================================================
INFO   …summary+             MeasurementSet Name:  /pipeline1/processing/TS8004/TS8004_C_0
INFO   …summary+   ================================================================
INFO   …summary+      Observer: TS8004     Project: TS8004
INFO   …summary+   Observation: e-MERLIN
INFO   …perties    Computing scan and subscan properties...
INFO   …:summary   Data records: 1201680       Total elapsed time = 81589.8 seconds
INFO   …summary+      Observed from   01-Aug-2019/23:20:10.5   to   02-Aug-2019/22:00:00.2
INFO   …:summary
INFO   …summary+      ObservationID = 0         ArrayID = 0
INFO   …summary+     Date        Timerange (UTC)          Scan  FldId FieldName
INFO   …summary+     01-Aug-2019/23:20:10.5 - 00:00:00.2     1      3 0319+4130
INFO   …summary+     02-Aug-2019/00:00:04.0 - 00:01:59.5     2      2 1302+5748
INFO   …summary+                 00:02:03.0 - 00:06:01.0     3      1 1252+5634
INFO   …summary+                 00:06:04.0 - 00:07:59.5     4      2 1302+5748
INFO   …summary+                 00:08:03.0 - 00:12:00.2     5      1 1252+5634
INFO   …summary+                 00:12:03.0 - 00:14:00.3     6      2 1302+5748
INFO   …summary+                 00:14:03.0 - 00:18:01.0     7      1 1252+5634
INFO   …summary+                 00:18:04.0 - 00:20:01.3     8      2 1302+5748
INFO   …summary+                 00:20:03.0 - 00:24:00.3     9      1 1252+5634
INFO   …summary+                 00:24:03.0 - 00:26:00.2    10      2 1302+5748
```

About the 3c277.1 dataset

```
           (nRows = Total number of rows per scan)
Fields: 5
  ID   Code Name                RA                Decl             Epoch        nRows
  0    ACAL 1331+3030           13:31:08.287300 +30.30.32.95900 J2000         80820
  1         1252+5634           12:52:26.285900 +56.34.19.48800 J2000        637500
  2         1302+5748           13:02:52.465277 +57.48.37.60932 J2000        312840
  3    CAL  0319+4130           03:19:48.160110 +41.30.42.10330 J2000        116640
  4    CAL  1407+2827           14:07:00.394410 +28.27.14.68990 J2000         53880
Spectral Windows:  (4 unique spectral windows and 1 unique polarization setups)
  SpwID  Name   #Chans   Frame   Ch0(MHz)  ChanWid(kHz)  TotBW(kHz) CtrFreq(MHz)  Corrs
  0      none     128    GEO    4816.500     1000.000    128000.0   4880.0000    RR  RL  LR  LL
  1      none     128    GEO    4944.500     1000.000    128000.0   5008.0000    RR  RL  LR  LL
  2      none     128    GEO    5072.500     1000.000    128000.0   5136.0000    RR  RL  LR  LL
  3      none     128    GEO    5200.500     1000.000    128000.0   5264.0000    RR  RL  LR  LL
The SOURCE table is empty: see the FIELD table
Antennas: 6:
  ID   Name  Station   Diam.    Long.         Lat.                Offset from array center (m)
                                                                     East         North     Elevation
  0    Mk2   Mk2       24.0 m   -002.18.14.1  +53.02.57.3       19618.7284    20856.7583    6908.7107
  1    Kn    Kn        25.0 m   -002.59.49.7  +52.36.17.2      -26823.2185   -28465.4973    7055.9694
  2    De    De        25.0 m   -002.08.40.0  +51.54.49.9       30300.7876  -105129.6730    7263.6278
  3    Pi    Pi        25.0 m   -002.26.43.3  +53.06.14.9       10141.4322    26944.5297    6845.6479
  4    Da    Da        25.0 m   -002.32.08.4  +52.58.17.2        4093.0417    12222.9915    6904.6753
  5    Cm    Cm        32.0 m   +000.02.13.7  +51.58.49.3      176453.7144   -97751.3908    7233.2945
##### End Task: listobs                  #####
##########################################
```

About the 3c277.1 dataset

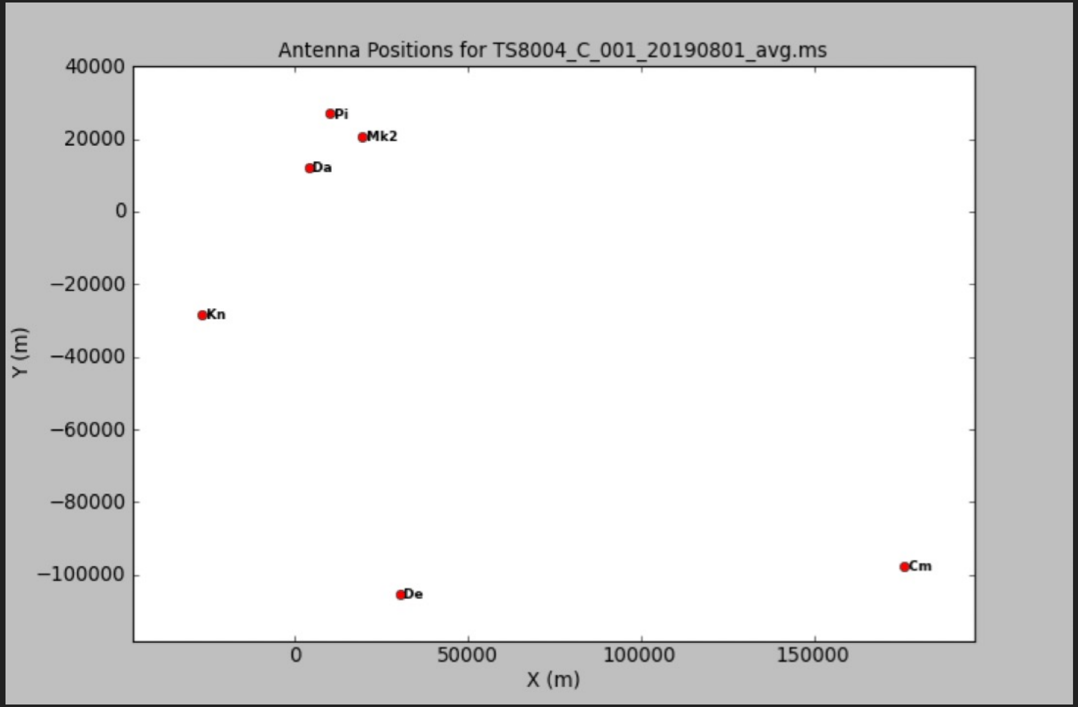
- From the listobs output, you will find out about the observations.
- There are 5 sources in the dataset:
  - Target 1252+5634
  - Phase cal 1302+5749
  - Bandpass cal 1407+2827
  - Flux cal 1331+3030
  - Point cal 0319+4130
- There are 4 spws, each with 128 channels and 4 polarisations (LL,RR,LR,RL)
- This is true for all e-MERLIN standard pipelined datasets for the _avg.ms file
- There are 6 antennas in the observation

About e-MERLIN

- If you run:

  ```
  plotants(vis='TS8004_C_001_20190801_avg.ms')
  ```

- You get the antenna layout of e-MERLIN. It is important to remember for future steps that we should use one of the core antennas for the reference antenna, i.e. Mk2/Da/Pi



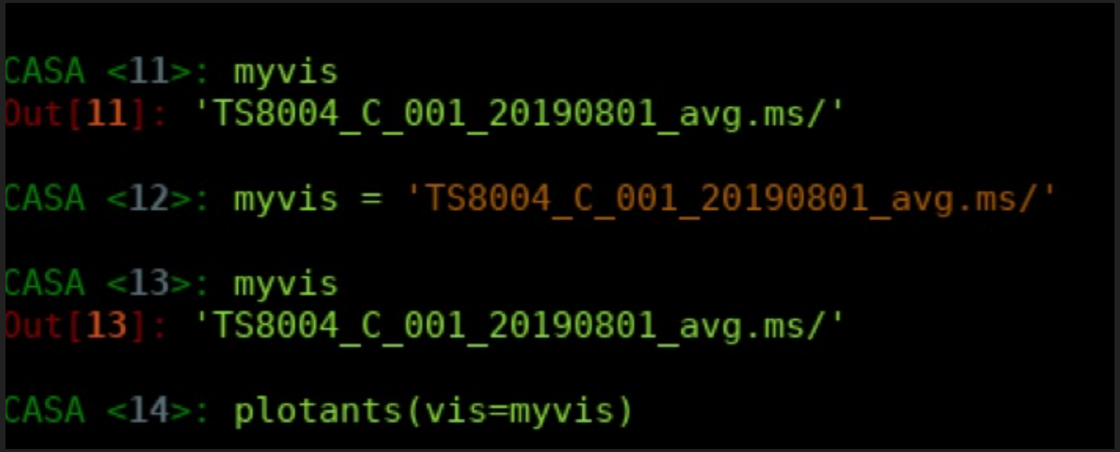
Basic CASA syntax

- You can set parameters that you can call easily later on such as:

  myvis = ='TS8004_C_001_20190801_avg.ms'

- And start a task with:

  plotants (vis=myvis)



Basic CASA syntax

- You can also save tasks for later using:

  tput listobs

- And get the last run parameters with:

  tget listobs

```
CASA <15>: tput listobs
---------> tput(listobs)

CASA <16>: tget listobs
---------> tget(listobs)
Restored parameters from file listobs.last

CASA <17>: inp listobs
---------> inp(listobs)
#  listobs :: List the summary of a data set in the l
vis                  = 'TS8004_C_001_20190801_avg.ms/'
file (MS)
selectdata           =        True        #  Data selec
     field           =          ''        #  Selection
```

eMCP files and weblogs

Files/folders of the eMCP

- **Inputs.ini** file
- **default_params.json** file
- **observatory.flags** file
- **eMCP.log** and **casa_eMCP.log** files
- (Optionally), a **manual_avg.flags** file
- MS file **_avg.ms** data and **_avg.ms.flagversions** folders

- Various folders including:
  - **weblog:** All of the html and images for your weblogs are stored here
  - **splits:** A folder with the split dataset of the target field(s)
  - **logs:** A folder with all the last versions of the casa tasks run and the casa logs
  - **eMERLIN_CASA_pipeline:** The folder with all of the eMCP materials and scripts

inputs.ini

- This file is one of the three files/folders you need to start the pipeline from scratch.
- The top set of parameters define your targets and calibrators. Multiple targets should be included separated by commas, with the associated phase calibrator included in a comma-separated list
- flag file names are included, and should not be altered

```
# Inputs for the e-MERLIN CASA pipeline:
[inputs]

fits_path =
/scratch/raw_data/TS8004/TS8004_C_001_201
90801/DATA/
inbase      = TS8004_C_001_20190801
targets     = 1252+5634
phscals     = 1302+5748
fluxcal     = 1331+3030
bpcal       = 1407+2827
ptcal       = 0319+4130

# Optional files and steps when they are
used:
# observatory.flags     [flag_apriori]
# manual.flags          [flag_manual]
#
manual_avg.flags      [flag_manual_avg]
#
manual_narrow.flags   [flag_manual_avg]
# shift_phasecenter.txt [average]
```

inputs.ini

- The second half of this file shows you all of the steps of the pipeline.
- They are split into pre_processing and calibration
- Generally, the data you received from e-MERLIN that has been calibrated with the eMCP will have had both sections run, but you will only be able to run the calibration section.

```
# Pipeline steps in groups in order of
execution:
# pre_processing
#      run_importfits
#      flag_aoflagger
#      flag_apriori
#      flag_manual
#      average
#      plot_data
#      save_flags
#
# calibration
#      restore_flags
#      flag_manual_avg
#      init_models
#      bandpass
#      initial_gaincal
#      fluxscale
#      bandpass_final
#      gaincal_final
#      applycal_all
#      flag_target
#      plot_corrected
#      first_images
#      split_fields
```

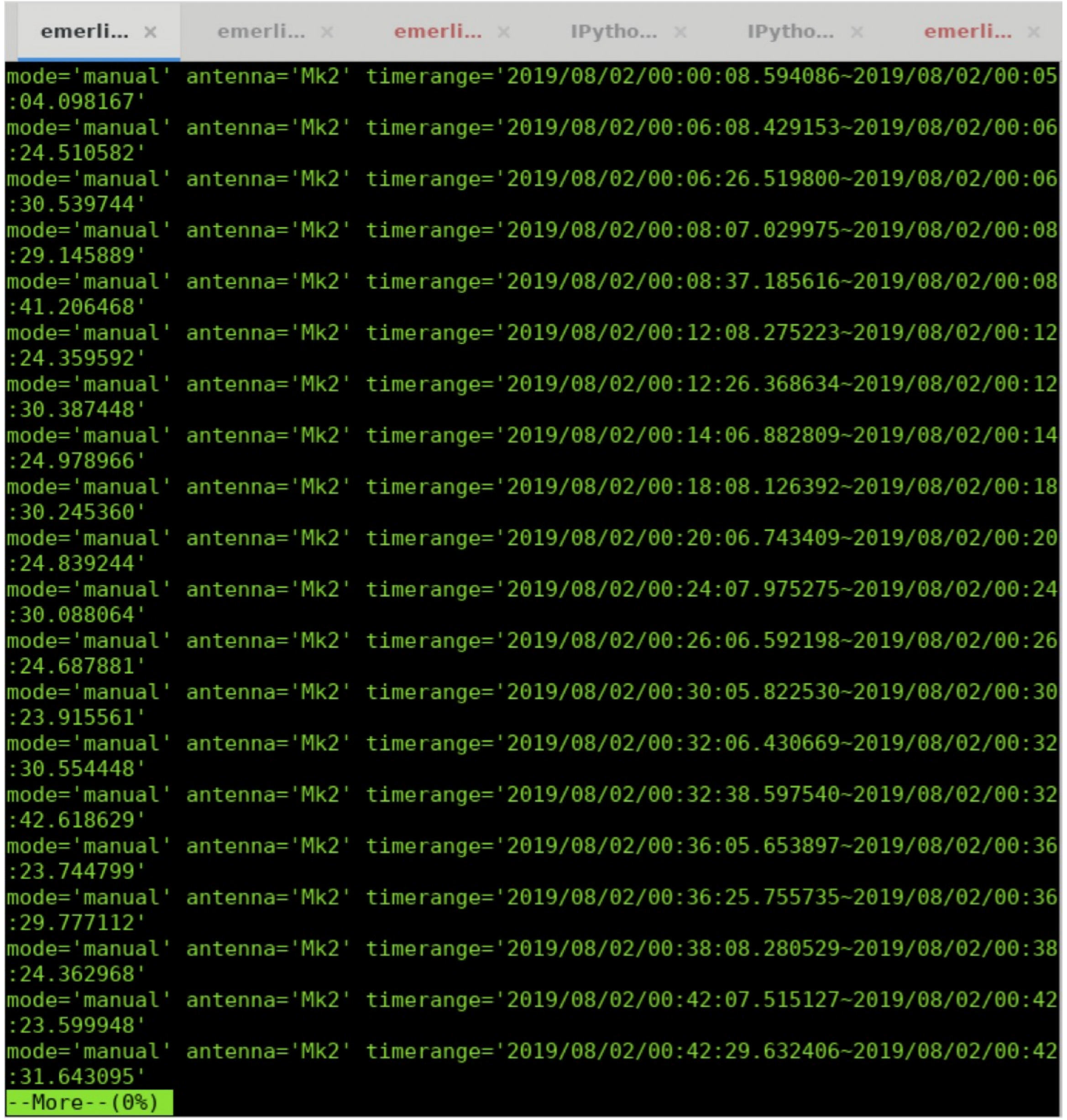
default_params.json

- This file is one of the three files/folders you need to start the pipeline from scratch.
- It holds all of the parameters that you can tweak and change for the calibration runs
- We will discuss this throughout the rest of this workshop

```
{
  "global": {
    "update_casa-data"          : true,
    "refantmode"                : "strict",
    "refant"                    : "",
    "is_mixed_mode"             : "auto",
    "applymode"                 :
"calflagstrict",
    "run_importfits"            : 1,
    "flag_aoflagger"            : 1,
    "flag_apriori"              : 1,
    "flag_manual"               : 1,
    "average"                   : 1,
    "plot_data"                 : 1,
    "save_flags"                : 1,
    "restore_flags"             : 1,
    "flag_manual_avg"           : 1,
    "init_models"               : 1,
    "bandpass"                  : 1,
    "initial_gaincal"           : 1,
    "fluxscale"                 : 1,
    "bandpass_final"            : 1,
    "gaincal_final"             : 1,
    "applycal_all"              : 1,
    "flag_target"               : 1,
    "plot_corrected"            : 1,
    "first_images"              : 1,
    "split_fields"              : 1
  },
```

observatory.flags

- You should have this file in your download area for most datasets
- It holds all of the time frames when the telescopes were not on source and is generated by Jodrell Bank
- If you don't have one of these files, then it means there was an issue with the flag file read out and you will have to do more flagging manually (see flag_apriori step)



eMCP.log files

- There are two log files, the eMCP.log file and the casa_eMCP.log file. Both of these can be found on the weblog in the Pipeline info tab
- On the top right is the output of the eMCP.log file, which shows the output of the pipeline
- On the bottom right is the casa_eMCP.log file which shows the casa logger information

```
2022-07-21 10:02:25 | INFO | Create directory: ./weblog/
2022-07-21 10:02:25 | INFO | Create directory: ./weblog/info/
2022-07-21 10:02:25 | INFO | Create directory: ./weblog/plots/
2022-07-21 10:02:25 | INFO | Create directory: ./weblog/calib/
2022-07-21 10:02:25 | INFO | Create directory: ./weblog/images/
2022-07-21 10:02:25 | INFO | Create directory: ./logs/
2022-07-21 10:02:25 | INFO | Create directory: ./weblog/plots/caltables
2022-07-21 10:02:25 | INFO | Starting pipeline
2022-07-21 10:02:25 | INFO | Running pipeline from:
2022-07-21 10:02:25 | INFO | /pipeline1/processing/TS8004/TS8004_C_001_20190801/eMERLIN_CASA_pipeline/
2022-07-21 10:02:25 | INFO | CASA version: 5.8.0
2022-07-21 10:02:25 | INFO | Pipeline version: v1.1.19
2022-07-21 10:02:25 | INFO | Using github branch: master
2022-07-21 10:02:25 | INFO | github last commit: f2b6efa
2022-07-21 10:02:25 | INFO | This log uses UTC times
2022-07-21 10:02:25 | INFO | Loading default parameters from ./default_params.json:
2022-07-21 10:02:25 | INFO | fits_path : /scratch/raw_data/TS8004/TS8004_C_001_20190801/DATA/
2022-07-21 10:02:25 | INFO | inbase    : TS8004_C_001_20190801
2022-07-21 10:02:25 | INFO | targets   : 1252+5634
2022-07-21 10:02:25 | INFO | phscals   : 1302+5748
2022-07-21 10:02:25 | INFO | fluxcal   : 1331+3030
2022-07-21 10:02:25 | INFO | bpcal     : 1407+2827
2022-07-21 10:02:25 | INFO | ptcal     : 0319+4130
```

```
2022-07-21 10:02:39     INFO    importfitsidi::::
2022-07-21 10:02:39     INFO    importfitsidi::::+      ##########################################
2022-07-21 10:02:39     INFO    importfitsidi::::+      ##### Begin Task: importfitsidi      #####
2022-07-21 10:02:39     INFO    importfitsidi::::       importfitsidi(fitsidifile=
['/scratch/raw_data/TS8004/TS8004_C_001_20190801/DATA/TS8004_C_001_20190801_00.fits'],vis="TS8004_C_001_20190801_impo
rted.ms",constobsid=True,scanreindexgap_s=15.0,specframe="GEO")
2022-07-21 10:02:39     INFO    importfitsidi::::
2022-07-21 10:02:39     INFO    importfitsidi::::       ### Reading file
/scratch/raw_data/TS8004/TS8004_C_001_20190801/DATA/TS8004_C_001_20190801_00.fits
2022-07-21 10:02:46     INFO    MSFitsIDI::readFITSFile()       Correlator: e-MERLIN
2022-07-21 10:02:46     INFO    FitsIDItoMS()::readFitsFile     Found binary table ARRAY_GEOMETRY
2022-07-21 10:02:46     INFO    FitsIDItoMS::fillAntennaTable   number of antennas = 6
2022-07-21 10:02:46     INFO    FitsIDItoMS::fillAntennaTable   array ref pos = [0, 0, 0]
2022-07-21 10:02:46     INFO    FitsIDItoMS::fillAntennaTable      antenna_no 2 -> antenna ID 0
2022-07-21 10:02:46     INFO    FitsIDItoMS::fillAntennaTable+     antenna_no 5 -> antenna ID 1
2022-07-21 10:02:46     INFO    FitsIDItoMS::fillAntennaTable+     antenna_no 6 -> antenna ID 2
2022-07-21 10:02:46     INFO    FitsIDItoMS::fillAntennaTable+     antenna_no 7 -> antenna ID 3
2022-07-21 10:02:46     INFO    FitsIDItoMS::fillAntennaTable+     antenna_no 8 -> antenna ID 4
2022-07-21 10:02:46     INFO    FitsIDItoMS::fillAntennaTable+     antenna_no 9 -> antenna ID 5
2022-07-21 10:02:46     INFO    FitsIDItoMS()::readFitsFile     Found binary table SOURCE
2022-07-21 10:02:46     INFO    FitsIDItoMS()::readFitsFile     Found binary table FREQUENCY
2022-07-21 10:02:47     INFO    FitsIDItoMS()::readFitsFile     Found binary table ANTENNA
2022-07-21 10:02:47     WARN    FitsIDItoMS()::readFitsFile     Treating POLAA and POLAB columns in input ANTENNA
table as scalar,
```

manual flag files

- There can be a few of these files:
  - manual.flags
  - manual_avg.flags
  - manual_narrow.flags
- These will be read in via the pipeline at various points to perform flags inputted manually by the user or support scientist

Example flag file for e-MERLIN

mode='manual' field='1331+305' antenna=''

timerange='10:00:00~10:11:30’

mode='manual' field='' antenna='' timerange=''

spw='0:0~30’

mode='manual' field='' antenna='Mk2'

timerange='09:05:00~16:27:00’

mode='manual' field='1258-2219' antenna=''

timerange='12:57:01~12:59:59’

mode='quack' field='1258-2219,1309-2322'

quackinterval=24.

Example flag file from CASA docs

scan='1~3’

mode='manual’

this line will be ignored spw='9' mode='tfcrop'

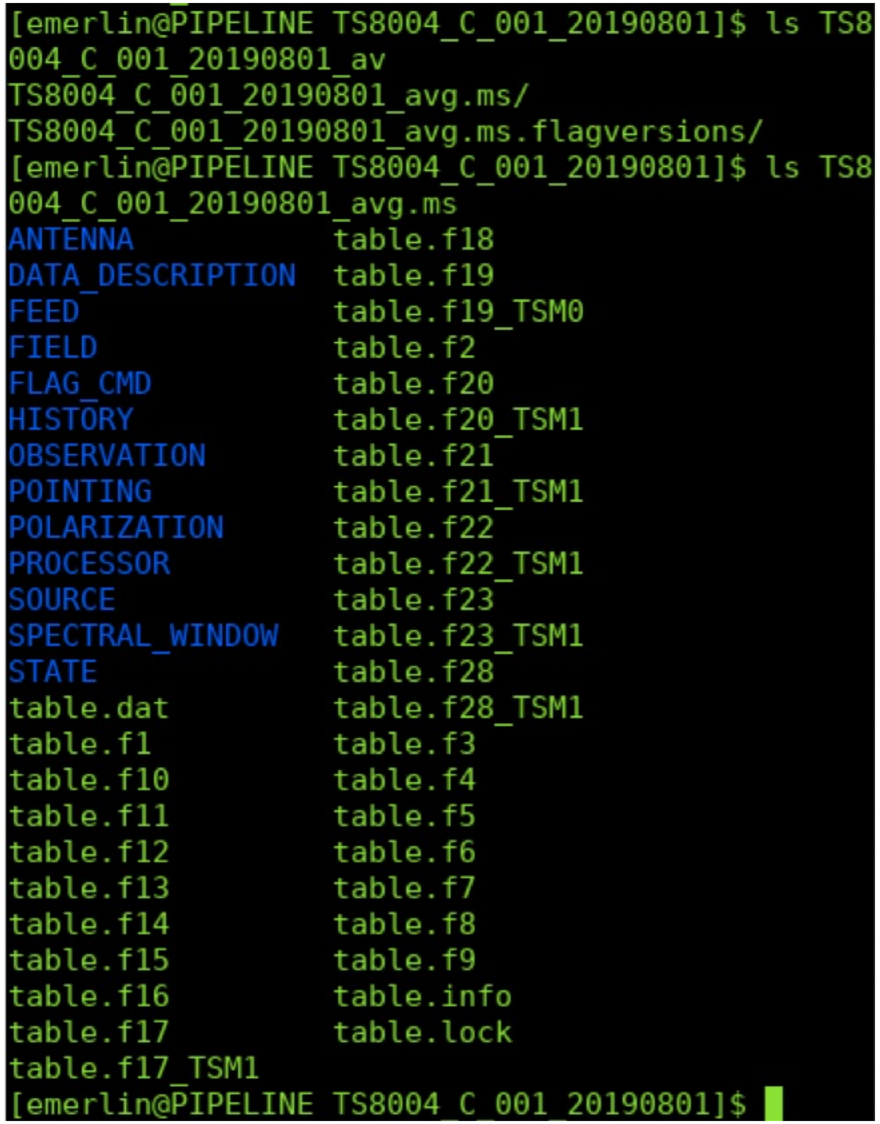
correlation='ABS_XX,YY' ntime=51.0

mode='extend' extendpols=True scan='1~3,10~12’

mode='quack' quackinterval=1.0

Measurement set

- This is your data!
- It is accompanied with a flagversions file, which is appended to throughout the pipeline so you can restore a previous flagversions if necessary
- We will average down this data further before we start re-calibrating it, to speed up the processing



Weblog folders

- These are your weblogs, i.e. these are produced by the pipeline so that you can inspect and sanity check the calibration has been performed correctly.
- We will go over the weblogs in detail next. To load these, you can use firefox and go to the:

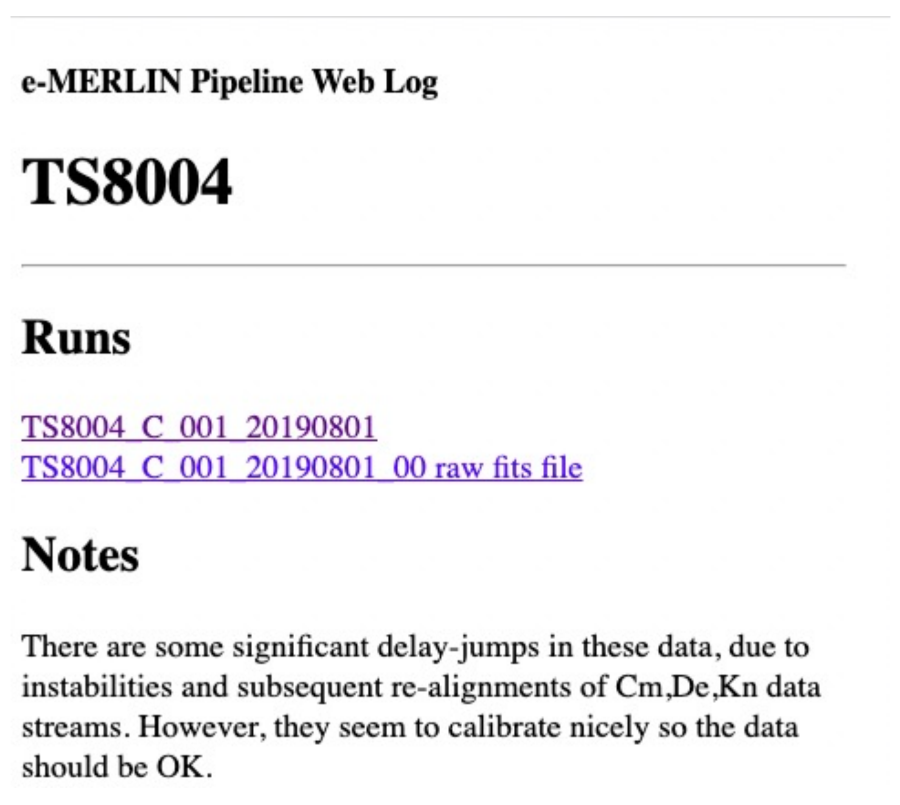
  */workingarea/*Data/TS8004_C_001/20190801/weblog/

- You can look at the weblogs on the web too (see next slide)

```
[emerlin@PIPELINE TS8004_C_001_20190801]$ ls weblog
calib                 plots
calibration.html      plots_corrected_0319+4130.html
download.html         plots_corrected_1252+5634.html
eMCP.css              plots_corrected_1302+5748.html
eMCP_logo.png         plots_corrected_1331+3030.html
emerlin-2.gif         plots_corrected_1407+2827.html
flagstats.html        plots_data_0319+4130.html
images                plots_data_1252+5634.html
images.html           plots_data_1302+5748.html
index.html            plots_data_1331+3030.html
info                  plots_data_1407+2827.html
obs_summary.html      plots.html
pipelineinfo.html
[emerlin@PIPELINE TS8004_C_001_20190801]$
```

Getting Access to the 3C277.1 data

- All of the data will be on the 3C277.1 distribution page:
  - https://www.e-merlin.ac.uk/distribute/CY8/TS8004/TS8004.html
- This is the distribution page that holds the observing "Runs" for a project
- The "Notes" heading is usually included in the data distribution email from your support scientist
- At the bottom of the page are two further headings: "Pipeline information" and "Easy way to re-run the pipeline". For the purposes of this tutorial, we only need to look at the first link: TS8004_C_001_20190801



The Home Page

- When you click on an observing run, you are taken to the home page for that run.
- This page has information on the observing parameters for that run.
- Importantly, all of the information on this page refers to the data that is averaged down an in the “_avg.ms” file made partway through the pipeline.
- For example, the integration time of e-MERLIN is 1s but we average to 4s in time to save time and space for calibration

Home

Project	TS8004
Run	TS8004_C_001_20190801
MS file	./TS8004_C_001_20190801_avg.ms
Start	2019-08-01 23:20
End	2019-08-02 21:59
Band	C
Antennas	Mk2, Pi, Da, Kn, De, Cm
Number of sources	5
Integration time	4.0s
Frequency	4.82 - 5.33 GHz
Num. spw	4
Channels/spw.*	128
Channel width	1.00 MHz
spw bandwidth	128 MHz

Observation Summary

- The Observation Summary tab has a handful of key parts
  - A listobs Summary
  - Sources List
  - Antennas
  - Source elevation
  - UV Coverage plots
- The Summary includes listobs files of the averaged (and unaveraged!) data
- The Sources table lists all of the objects observed in this run.

Summary:

Summary of current observation (listobs): txt

Other available listobs files:
TS8004_C_001_20190801.ms.listobs.txt: txt

Sources:

Target	Phase cal	Separation [deg]
1252+5634	1302+5748	1.88

Source pairs and separations: txt

Sources in MS:

Source	Intent
0319+4130	ptcal
1252+5634	targets
1302+5748	phscals
1331+3030	fluxcal
1407+2827	bpcal

Observation Summary

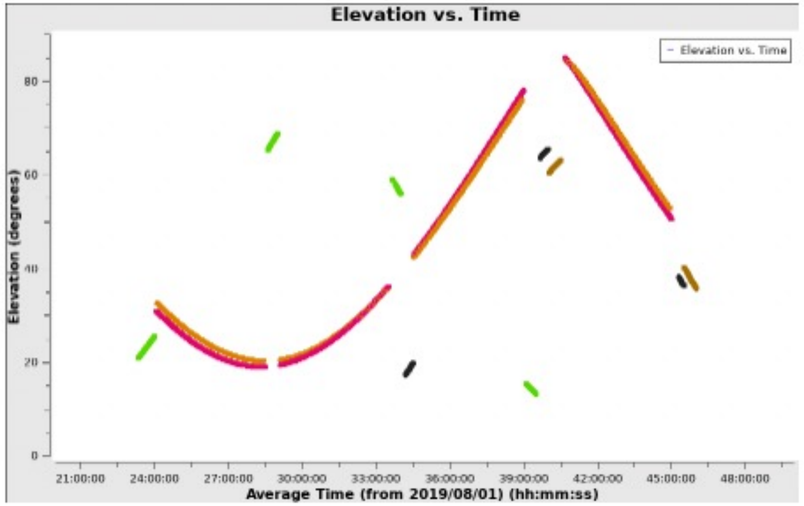
- The Antennas list includes all antennas that were used in the observing file prepared at JBO. The reference antenna list is shown here as calculated during the pipeline
- The source elevation plot is also shown, with sources colourised by the field ID.

Antennas:

Mk2
Pi
Da
Kn
De
Cm

Reference antenna: Pi,Mk2,Cm,Da,Kn,De

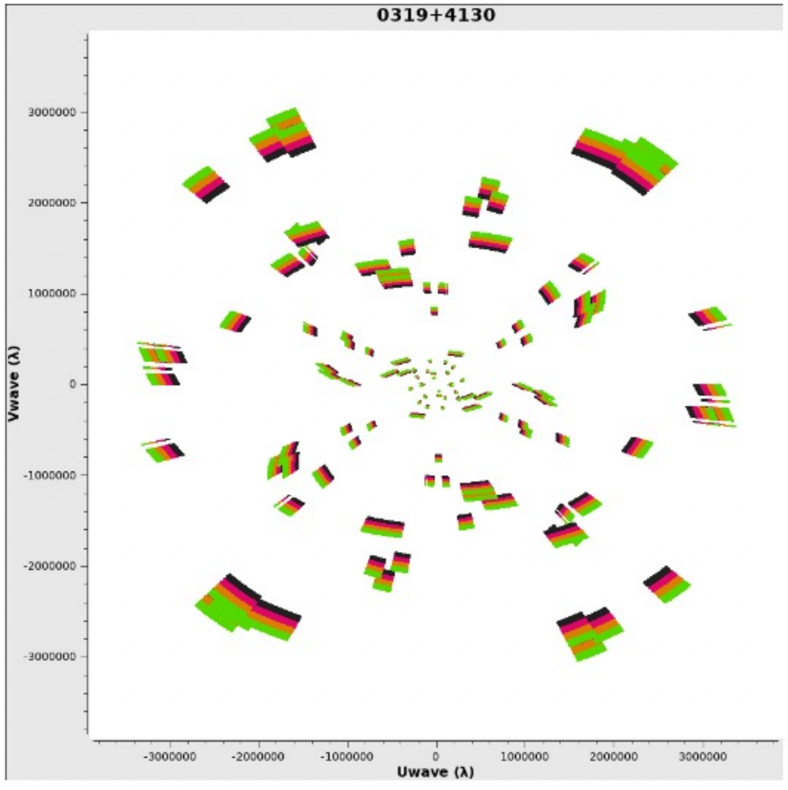
Source elevation:



Observation Summary

- UV coverage plots of all the sources in the observing run are included here. These are made prior to flagging, so you may not have all these data available for your observation at the end of the pipeline.

UV coverage:



Pipeline Info

- The Pipeline Info tab shows you first which CASA and eMCP version was used to run the pipeline
- A long form table with each of the calibration steps is shown with green denoting an executed step, and red showing a failed step.

Pipeline info

CASA version: 5.8.0
Pipeline version: v1.1.19

Execution summary

Step	Code	Execution ended	Execution time	Notes
start_pipeline	OK	2022-07-21 10:02:39	-	first execution
run_importfits	OK	2022-07-21 10:42:23	40 min	constobsid=True, scanreindexg Hanning=False, createmms=Fa
flag_aoflagger		0	-	
flag_apriori	OK	2022-07-21 10:59:30	11 min	Clip zeros. Subband edges *:0 channels 0:0~26 3:485~511. O flags: observatory.flags.
flag_manual	OK	2022-07-21 11:00:19	<1 min	No flagging file.
average	OK	2022-07-21 11:03:43	3 min	chanbin=4, timebin=4s, datacc
plot_data	OK	2022-07-21 11:26:32	18 min	
save_flags	OK	2022-07-21 11:26:34	<1 min	versionname=initialize_flags
restore_flags	OK	2022-07-21 11:27:46	<1 min	versionname=initialize_flags
flag_manual_avg	OK	2022-07-21 11:28:06	<1 min	No flagging file.
init_models	OK	2022-07-21 11:29:17	1 min	
bandpass	OK	2022-07-21 11:33:19	4 min	field=1407+2827, combine=fie solint=inf
initial_gaincal	OK	2022-07-21 12:04:22	31 min	delay solint=180s, combine=sp flagmode=tfcrop, p_solint=int,

Pipeline Info

- Log files include:
  - eMCP.log -> simplified log file
  - casa_eMCP.log -> casa log file output
- Parameter files include:
  - eMCP_info.txt -> Parameters used for each step of pipeline
  - caltables.txt -> Parameters used for each calibration table

Relevant log files:

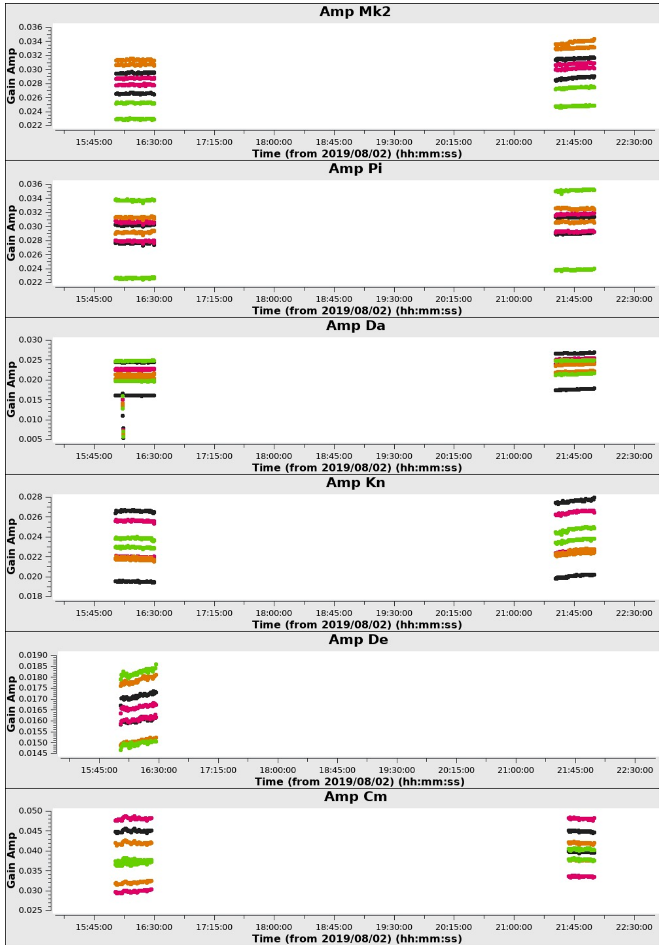
Pipeline log: eMCP.log
CASA log: casa_eMCP.log

Relevant parameter files:

Pipeline info (dict): eMCP_info.txt
Calibration info (dict): caltables.txt

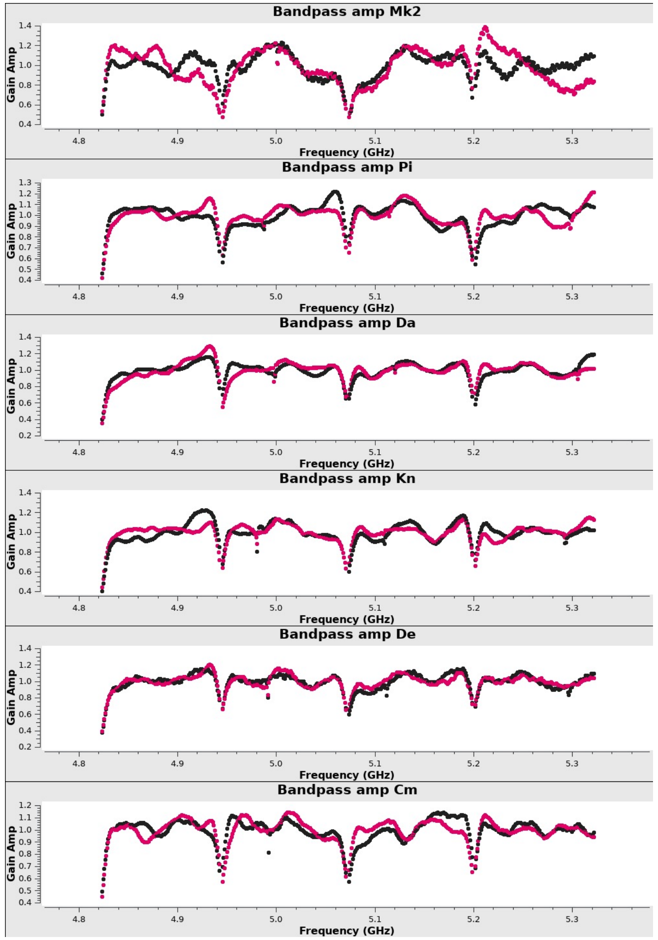
Calibration

- The calibration section will be covered during this data school, but I will give an overview of the types of plot we have on this tab:
- Amplitude/Phase/Delay plots: All shown plotted against time, and colourised by correlation (2 colours) or spw (4 colours).
  - Exception is the allcal_d.K1 plot which is colourised by field



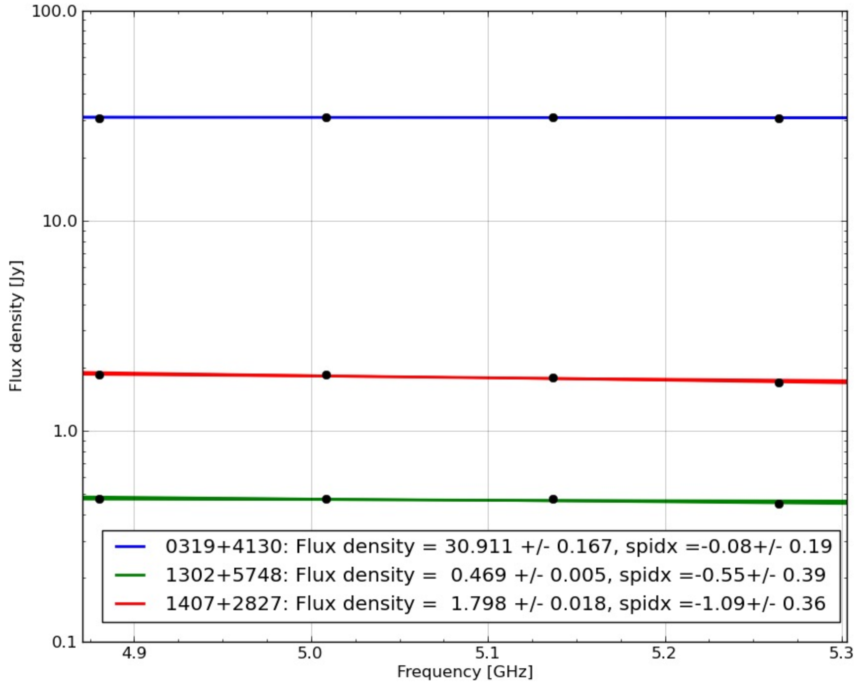
Calibration

- Bandpass plots show the gain Amp (or phase) plotted against the frequency, and colourised by correlation
- You can see the band edges in these plots, as well as the spw edges



Calibration

- Fluxscale plot: From the flux scaling step, we scale all of the data for calibrator fields using 3C286. This plot shows the result of that step, plotting flux density in Jy against Frequency. One data point per spw, and the fit is shown in the coloured lines per source.



Plots

- UV coverage plots of all the sources in the observing run are included here. These are made prior to flagging, so you may not have all these data available for your observation at the end of the pipeline.

Plots

Uncalibrated visibilities

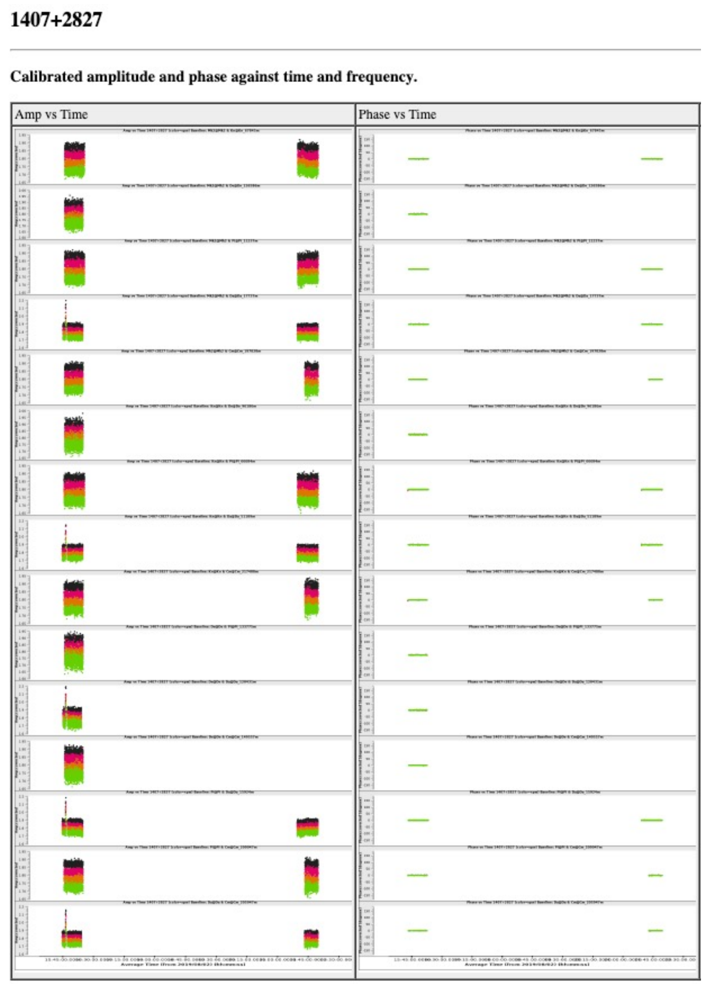
0319+4130 plots
1252+5634 plots
1302+5748 plots
1331+3030 plots
1407+2827 plots

Calibrated visibilities

0319+4130 plots
1252+5634 plots
1302+5748 plots
1331+3030 plots
1407+2827 plots

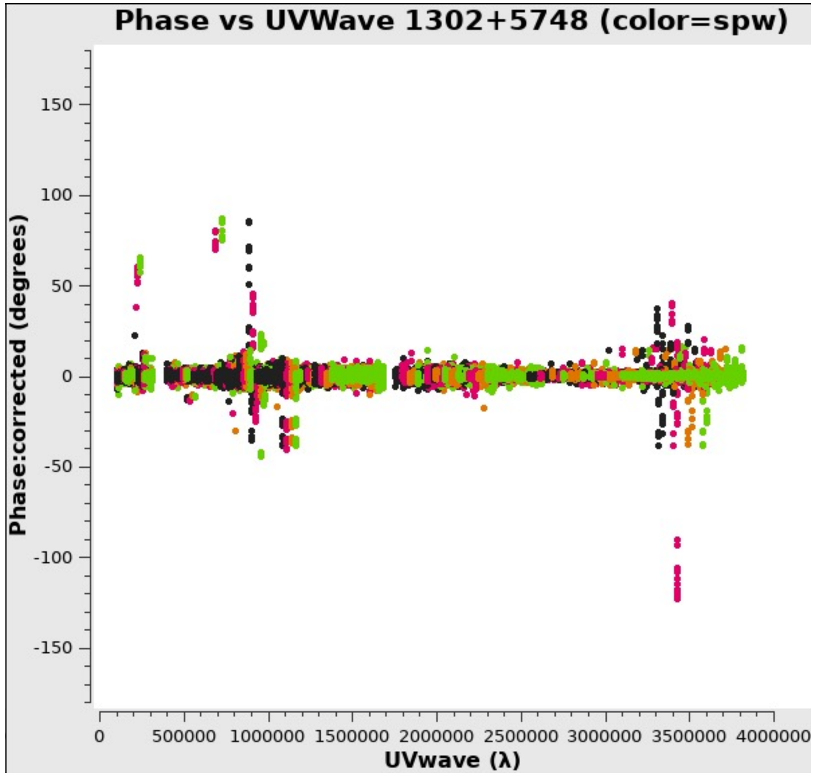
Plots

- Example calibrated plots for 1407+2827, showing calibrated amplitude and phase against time



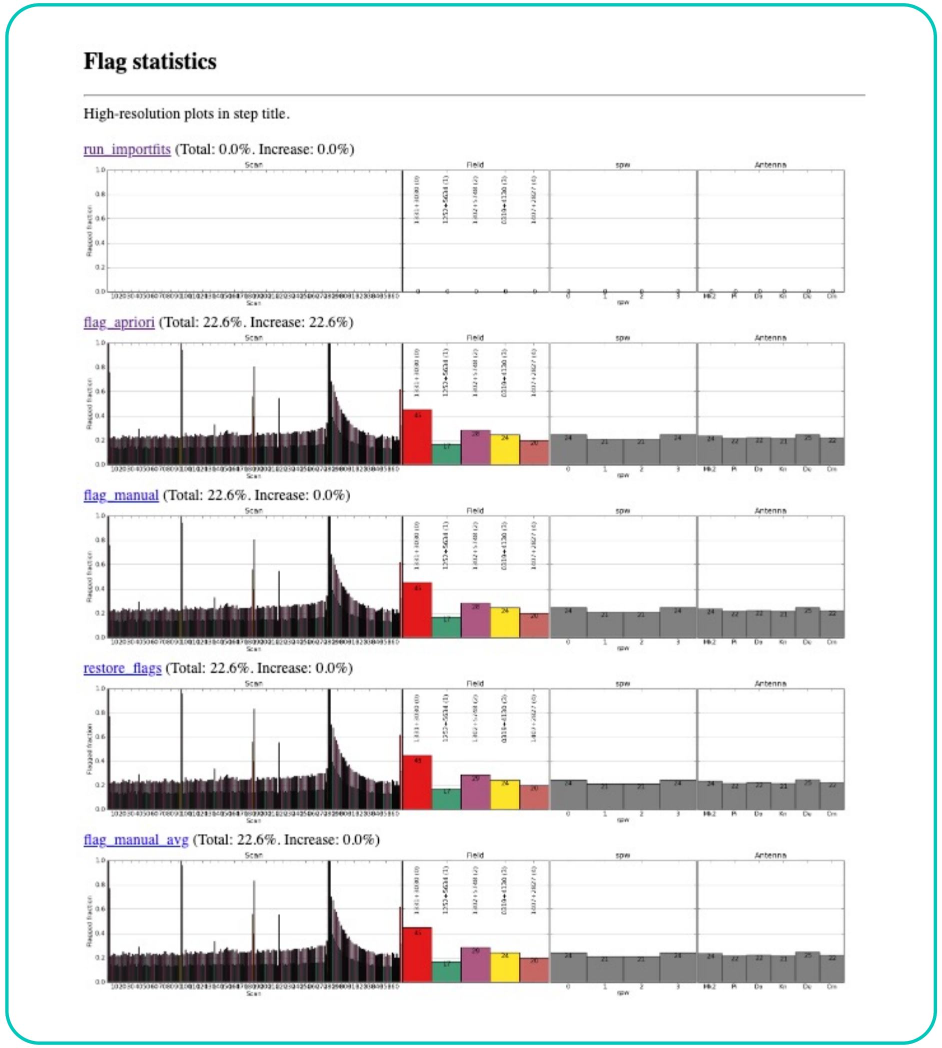
Plots

- The calibrated uv plots show the data and associated models in uv space, plotted with amplitude or phase on the y axis.
- You can often see small excursions from the model in these plots which can point to calibration errors in the data.



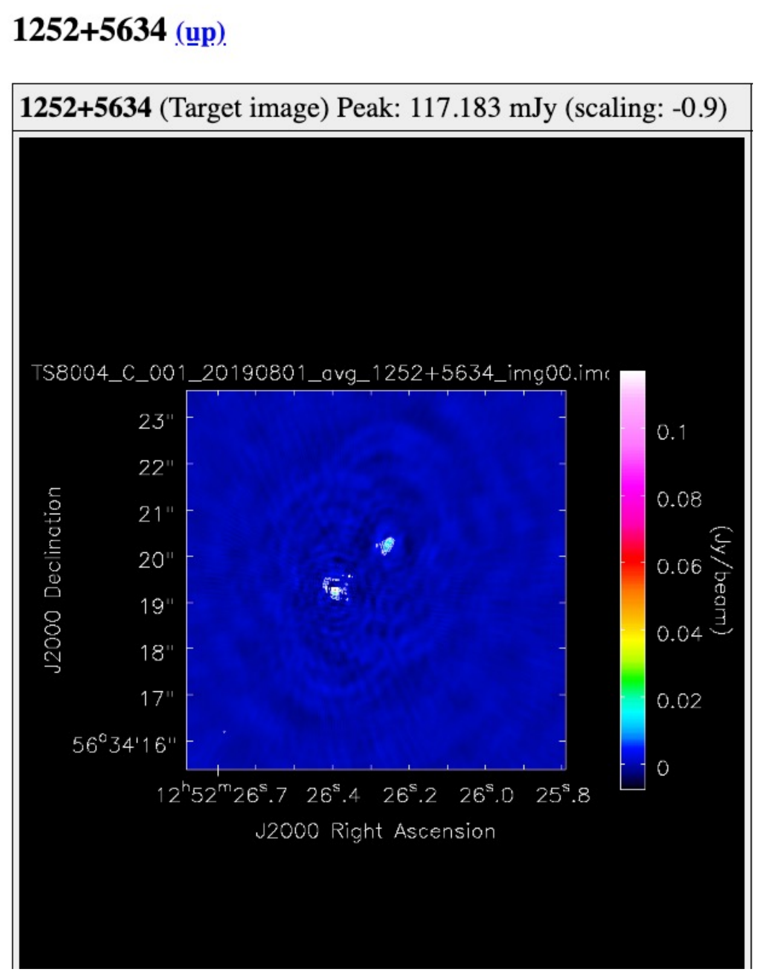
Flag Statistics

- The Flag Statistics tab shows the amount of flagging at different steps of the pipeline, in four different ways:
  - By scan
  - By field
  - By spw
  - By antenna



Images

- The Images tab shows the preliminary images of the target and phase calibrator, including residual images
- Another set of zoom images are also made for each source



Download Data

- The download data tab is as it suggests, the link will download the data from our distribution areas

Download data

This tar file contains the MS and all the plots in the weblog:
TS8004_C_001_20190801: tar

Any Questions?

The eMCP calibration procedures

The eMCP structure

- The eMCP is structured in two sections:
  - pre_processing
  - calibration
- In general, the pre_processing section cannot be re-run after data has been delivered but the calibration section can be re-run as many times as necessary to get well-calibrated data.
- All the steps are outlined on the github page: https://github.com/e-merlin/eMERLIN_CASA_pipeline

```
pre_processing
    run_importfits
    flag_aoflagger
    flag_apriori
    flag_manual
    average
    plot_data
    save_flags

calibration
    restore_flags
    flag_manual_avg
    init_models
    bandpass
    initial_gaincal
    fluxscale
    bandpass_final
    gaincal_final
    applycal_all
    flag_target
    plot_corrected
    first_images
    split_fields
```

pre_processing section

- The "pre_processing" section includes the following steps:
  - run_importfits
  - flag_aoflagger
  - flag_apriori
  - flag_manual
  - average
  - plot_data
  - save_flags
- We will go through these steps first so you know what they do.

```
pre_processing
    run_importfits
    flag_aoflagger
    flag_apriori
    flag_manual
    average
    plot_data
    save_flags

calibration
    restore_flags
    flag_manual_avg
    init_models
    bandpass
    initial_gaincal
    fluxscale
    bandpass_final
    gaincal_final
    applycal_all
    flag_target
    plot_corrected
    first_images
    split_fields
```

start_pipeline

- The start_pipeline step is not listed as a step as it is run every time you run the pipeline
- It will output information on the terminal about what steps are being run and read the measurement set to find all necessary information

```
pre_processing
    run_importfits
    flag_aoflagger
    flag_apriori
    flag_manual
    average
    plot_data
    save_flags

calibration
    restore_flags
    flag_manual_avg
    init_models
    bandpass
    initial_gaincal
    fluxscale
    bandpass_final
    gaincal_final
    applycal_all
    flag_target
    plot_corrected
    first_images
    split_fields
```

run_importfits

- This will load in the raw fits data and turn it into CASA measurement set format
- It averages the data in time, but not in frequency as it hasn't been flagged for RFI yet
- It will run hanning smoothing on the data if it is L band
- It will also split the data into continuum and narrow zooms spws, if the data is in spectral line mode

constobsid = true

scanreindexgap_s = 15.0

antenna=""

field=""

timeaverage = true

timebin = "4s"

chanaverage = false

chanbin = 1

usewtspectrum = false

run_hanning = "auto"

ms2mms = false

spw_separation = [","]

spwmap_sp = []

fix_repeated_sources = false

flag_aoflagger

- This will run the aoflagger software on the data. Note that the importfits step did *not* average in frequency, so we can use the full resolution data to remove the band RFI on a channel by channel basis.
- This step will *only* run on L band data where the RFI environment is challenging. It will not run on C band or K band data.

run = “auto”

fields = “all”

separate_bands = “false”

flag_apriori

- This step will make *a priori* flags of areas of known bad data, including reading in the observatory flags file.
- If an observatory flags file is not available, then standard "quacking" of the data is performed to remove time when the telescope was not on target
- The far edge channels are also flagged from the data at this stage.

```
border_chan_perc = 5.0
observatory_flags = true
do_estimated_quack = “auto”
all_quack = 4.0
std_cal_quack = 120.0
flag_Lo-Mk2 = true
spwmap_sp = []
observatory_flags = true
```

flag_manual

- This step performs additional flagging, but usually made by the e-MERLIN support scientist.
- It flags the data before it is averaged down or calibrated, so if there is something in the data that should not be used, then it should be flagged here
- This step has no default parameters

average

- This step will average the flagged dataset to the standard 4s and by 4 in frequency.
- It has a shift_phasecenter option which can be useful if you have a slight positional offset in the data, but care must be taken when running this and should be run in two parts

field = “”

timebin = “4s”

chanbin = 4

datacolumn = “data”

timerange = “”

scan = “”

antenna = “”

shift_phasecenter = false

plot_data + save_flags

- The plot_data step makes initial plots of the data and puts them in the plots tab of the weblog.
- The one default parameter for this section should always be left alone

- The save_flags step will save all the previous flags into a flagversions table which will be read in during the next part of the pipeline
- There are no default parameters

save_flags

- This step will save all the previous flags into a flagversions table which will be read in during the next part of the pipeline
- There are no default parameters

Any Questions?

calibration section

- The "calibration" section includes the following steps:
  - restore_flags
  - flag_manual_avg
  - init_models
  - bandpass
  - initial_gaincal
  - fluxscale
  - bandpass_final
  - gaincal_final
  - applycal_all
  - flag_target
  - plot_corrected
  - first_images
  - split_fields

```
pre_processing
    run_importfits
    flag_aoflagger
    flag_apriori
    flag_manual
    average
    plot_data
    save_flags

calibration
    restore_flags
    flag_manual_avg
    init_models
    bandpass
    initial_gaincal
    fluxscale
    bandpass_final
    gaincal_final
    applycal_all
    flag_target
    plot_corrected
    first_images
    split_fields
```

restore_flags + flag_manual_avg

- The restore_flags step will restore the flagging tables from the pre_processing part of the pipeline, i.e. from the save_flags step.
- flag_manual_avg will read in your flags from the manual_avg.flags file
- The flag_manual_avg part of the pipeline will also look at Lovell data and calculate where Lovell dropout scans are – Lovell stays on source for every other phase calibrator scan due to its slower slew speed.
- This step will also work out a reference antenna based on the calibrator data

Lo_dropout = “”

Lo_datacolumn = “data”

Lo_useflags = true

Lo_spws = [“3”]

Lo_threshold = 0.5

Lo_min_scans = “”

Examples of a manual_avg.flags file

- Use only ONE white space to separate the parameters (no commas). Each key should only appear once on a given command line/string.
- There is an implicit mode for each command, with the default being 'manual' if not given.
- Comment lines can start with '#' and will be ignored. The parser used in flagdata will check each parameter name and type and exit with an error if the parameter is not a valid flagdata parameter or of a wrong type.

Example of a manual_avg.flags file

```
mode='manual' timerange='2019/08/02/15:43:00~2019/08/02/15:47:00'

mode='manual' antenna='Da' spw='1' corr='LL'
```

You can also choose to clip the data although this is generally not recommended as it usually means that there is bad data below that clip level that would be better removed with a timerange flag.

Similarly, you can flag the data with a quack, if you wish, but the pipeline should have taken care of the worst of this in the pre_processing steps.

init_models

- This step initializes the model column in the measurement set for 3c286, using the model images in the e-MERLIN CASA Pipeline folder.
- This is important as 3c286 is slightly resolved on e-MERLIN baselines and this must be taken into account to get accurate flux recovery

calibrator_models =

“calibrator_models/”

manual_fluxcal = false

fluxcal_flux = [-1]

fluxcal_spix = 0.0

fluxcal_reffreq = “0GHz”

wtmode = “nyq”

dowtsp = false

bandpass

- The bandpass stage calibrates the bandpass calibrator (OQ208/1407+2827) first to estimate the bandpass response of the instrument.
- It will estimate the delay, then perform phase, amplitude+phase corrections, and finally compute the bandpass table
- The step will then flag the bandpass calibrator using tfcrop, delete the tables and re-run the above in its entirety
- This ensures that a bright RFI missed by aoflagger does not affect the bandpass calibration

Table specific parameters to be described in next slides

*_minblperant = 3

*_minsnr = 2

bp_uvrange = ""

bp_fillgaps = 8

bp_solnorm = true

apply_calibrators = ["bpcal.BP0"]

apply_targets = []

run_flag = true

bpcal_d.K0

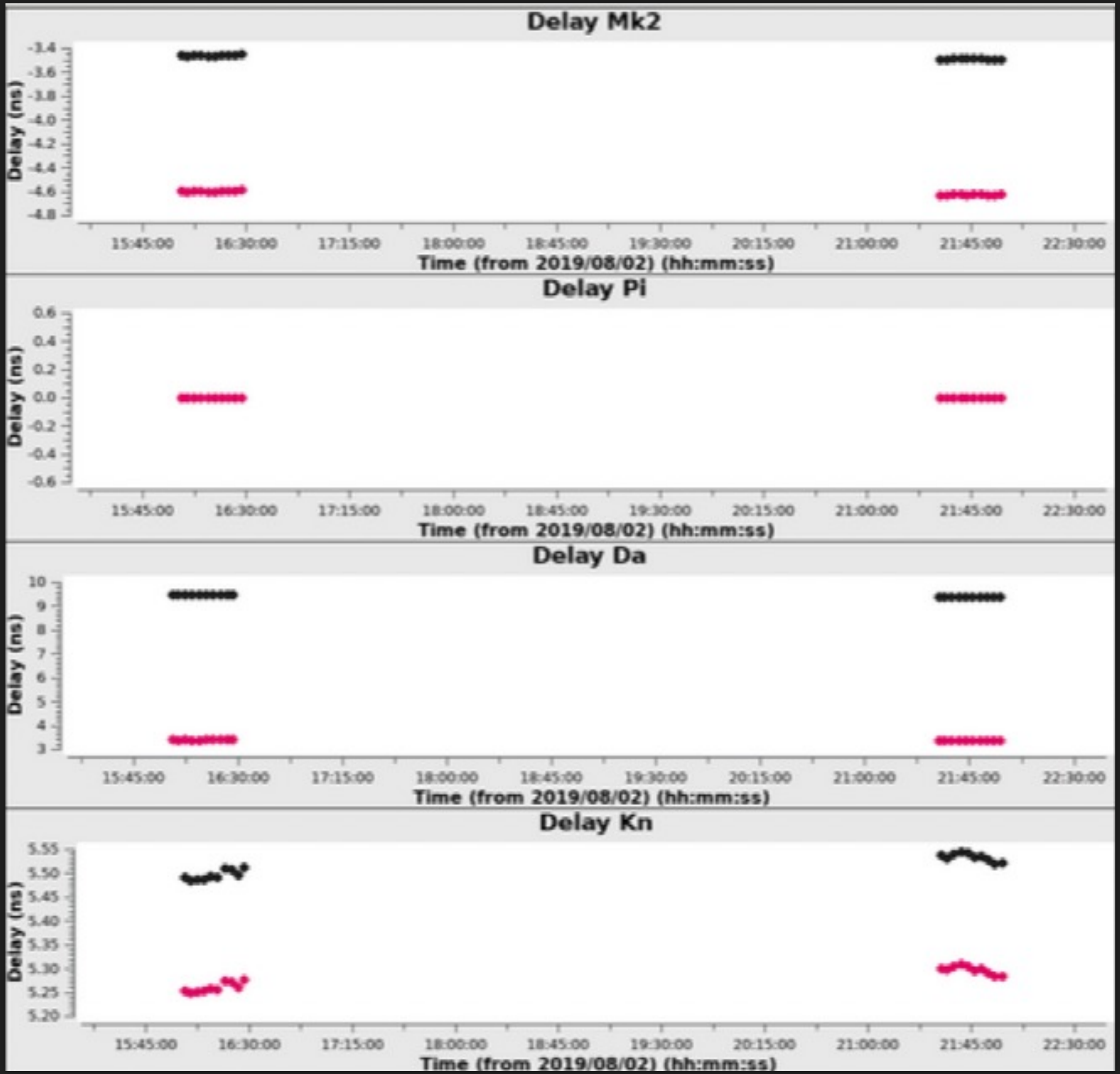


Table specific parameters

delay_tablename = “bpcal_d.K0”

delay_solint = “180s”

delay_combine = “spw”

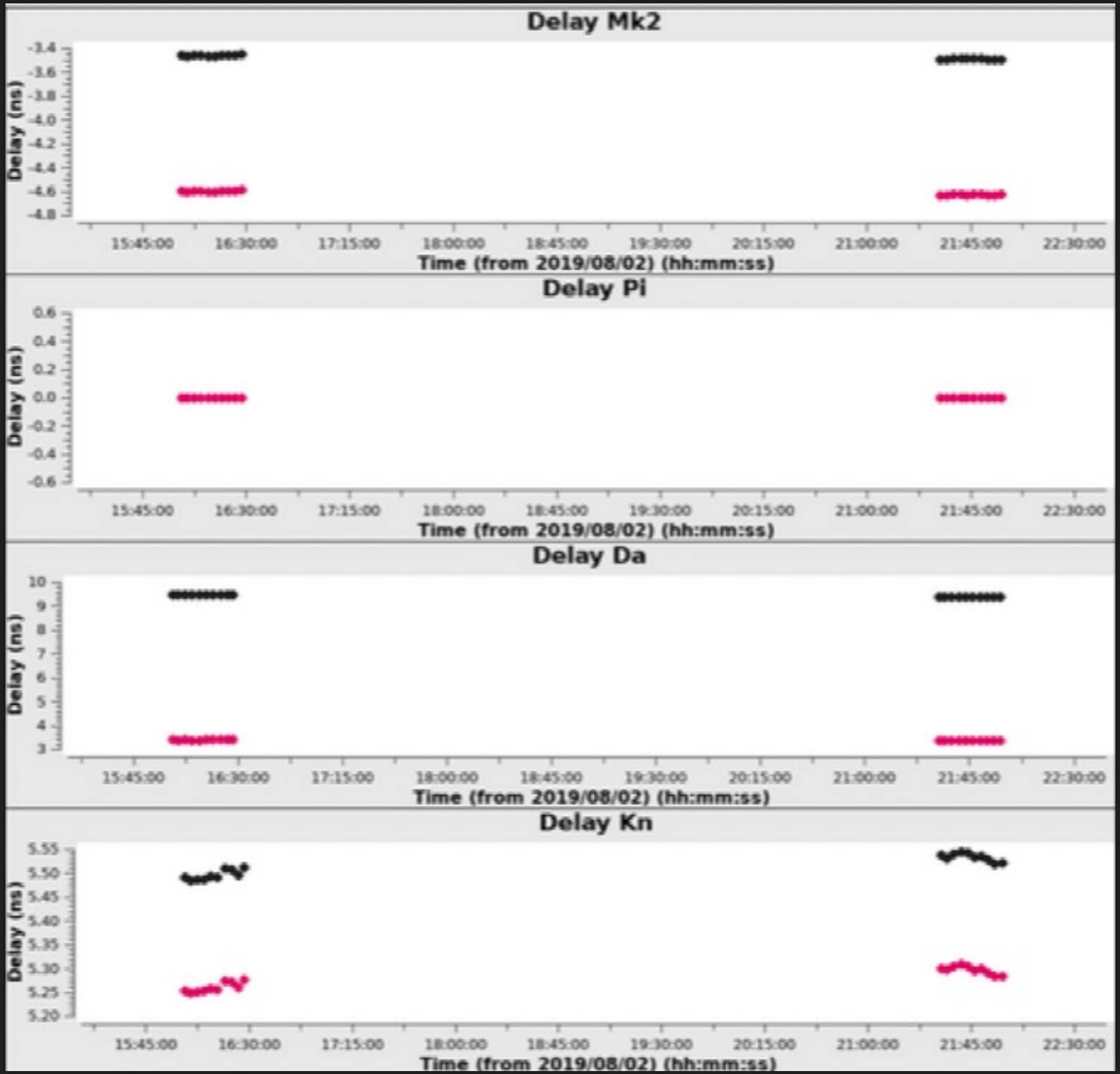
delay_prev_cal = []

delay_interp = “linear”

delay_spw = [“*”,”innerchan”]

bpcal_d.K0

- Refant will have a flat delay
- Check that other antennas flat relative to the refant
- They can be offset by several ns, and the polarisations can also be offset from each other
- Delay jumps are fine
- Variable delay rates are not fine



bpcal_p.G0

Table specific parameters

phase_tablename = "bpcal_p.G0"

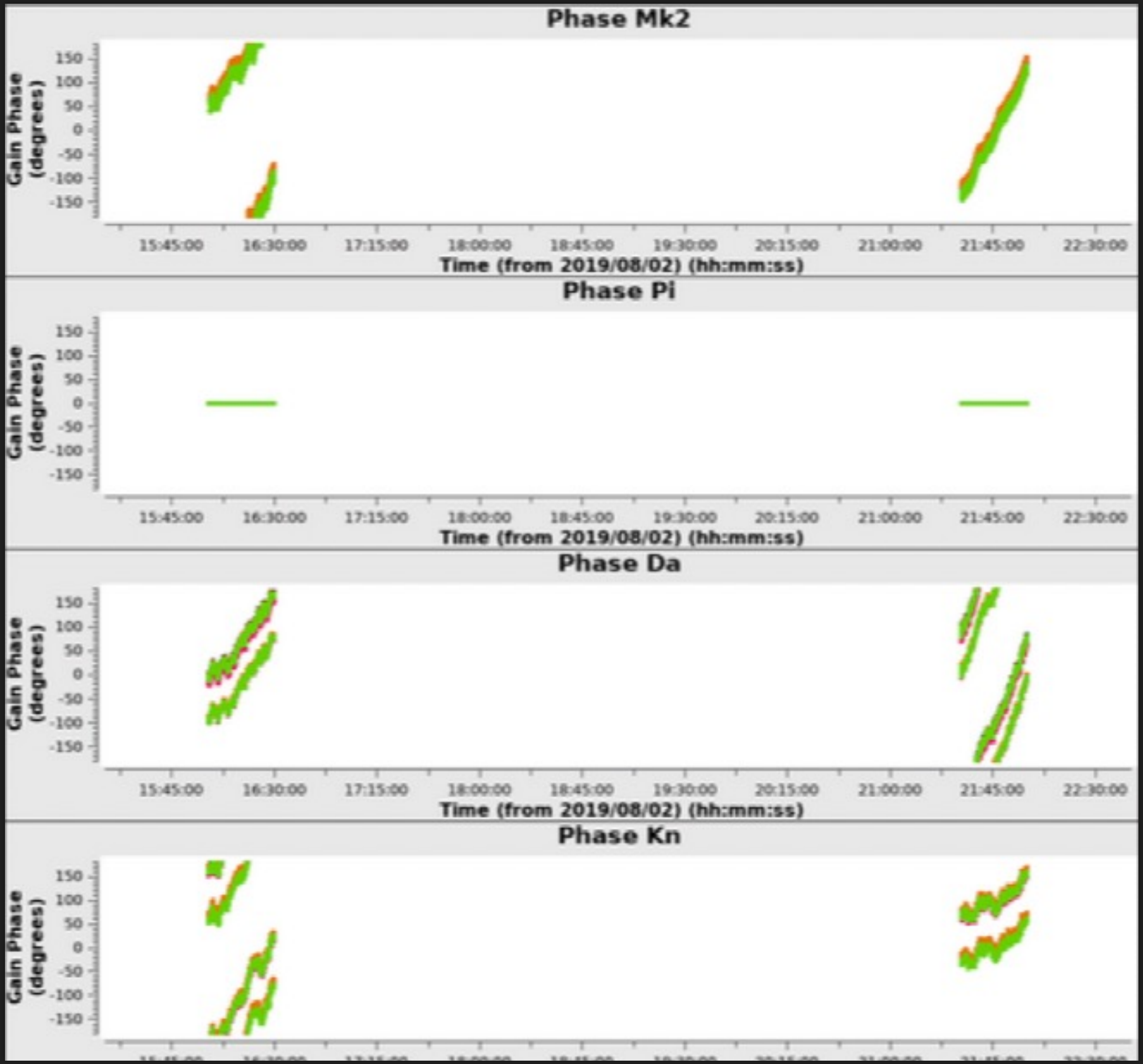
phase_solint = "int"

phase_combine = ""

phase_prev_cal = ["bpcal_d.K0"]

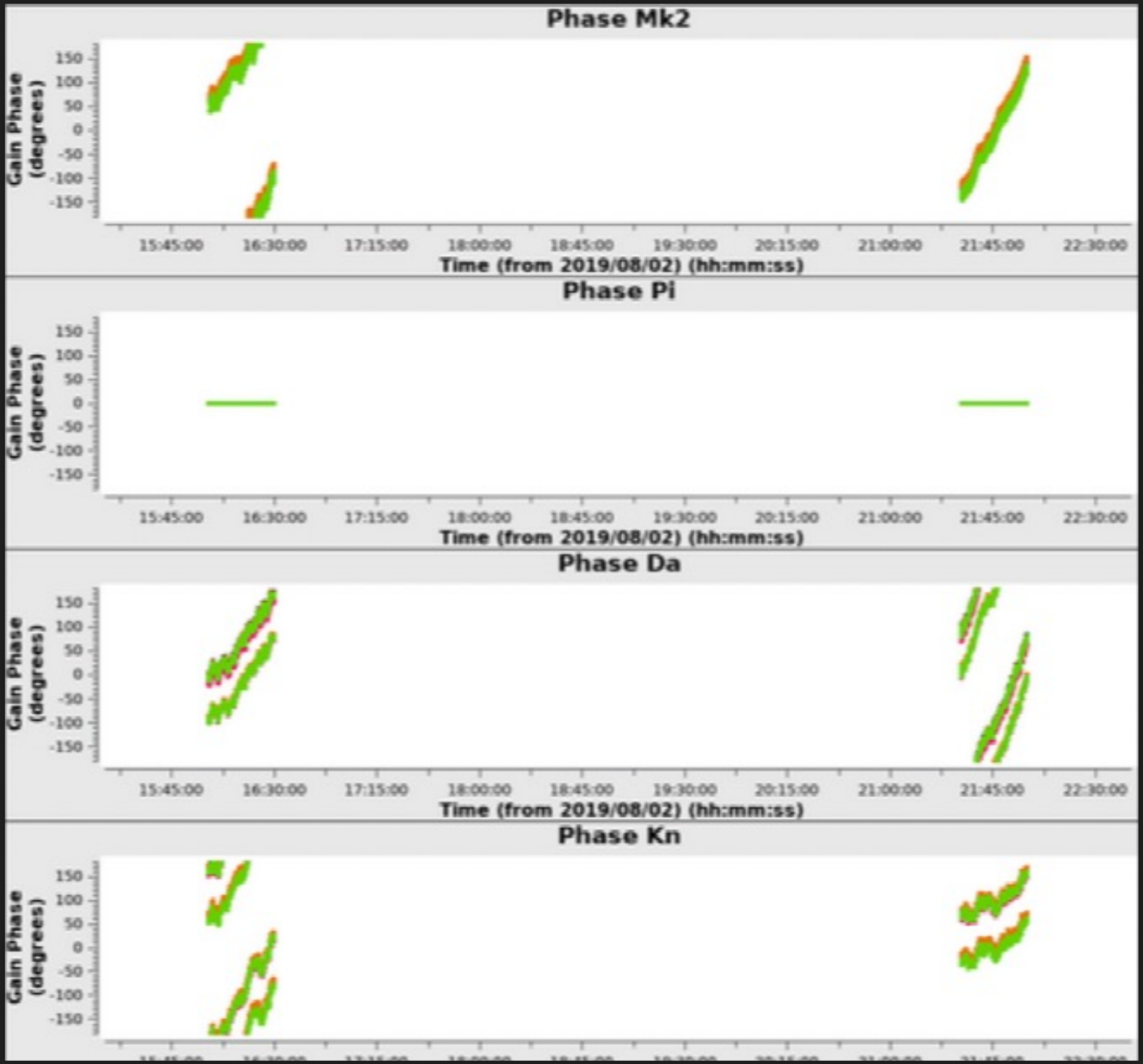
phase_interp = "linear"

phase_spw = ["*","innerchan"]



bpcal_p.G0

- Refant will have a flat phase
- Phase should evolve slowly over time
- Phase wrapping (over 360 degrees) is possible but shouldn't be occurring too quickly
- Areas with a vertical line suggest phase errors which should be checked/flagged



bpcal_ap.G0

Table specific parameters

ap_tablename = “bpcal_ap.G0”

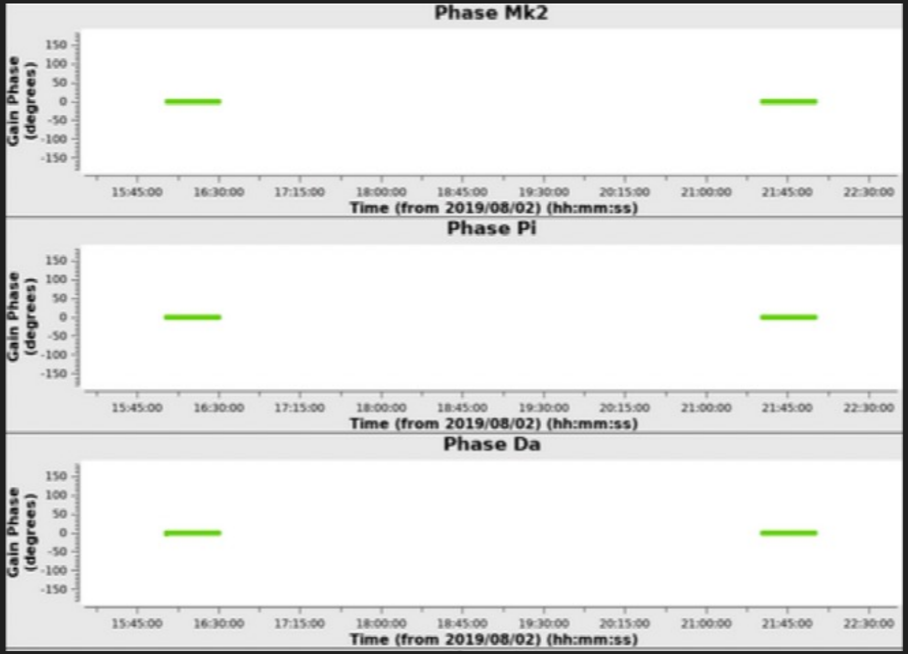
ap_solint = “32s”

ap_combine = “”

ap_prev_cal = [“bpcal_d.K0”, “bpcal_p.G0”]

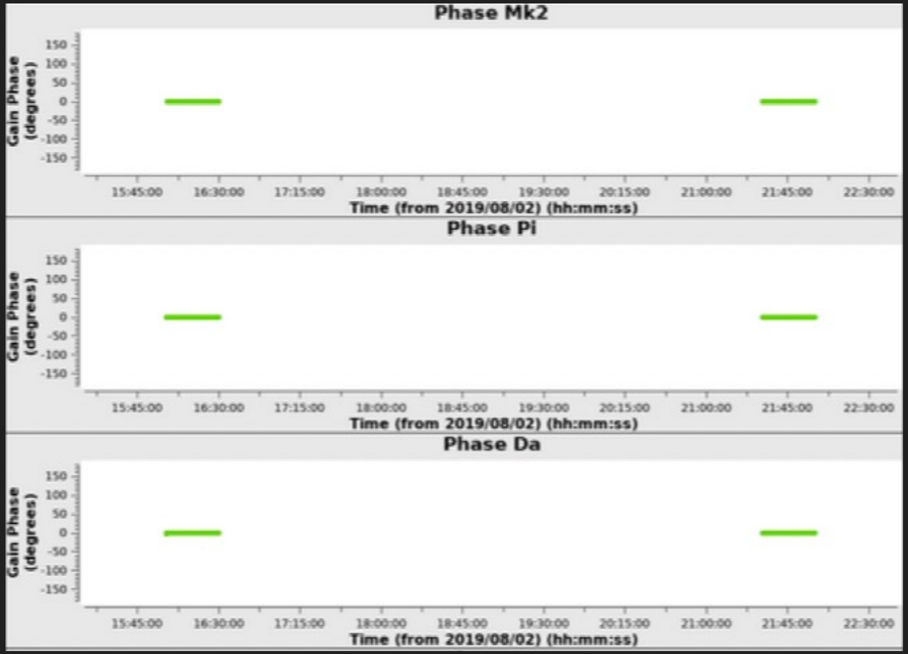
ap_interp = “linear”

ap_spw = [“*”,”innerchan”]



bpcal_ap.G0

- Two plots created: amplitude (top) and phase (bottom)
- Phase corrections applied from previous table mean the phases should be all zero here
- Amplitude drop outs (see Darnhall) should be noted for flagging later
- Look out for variable amplitudes or jumps – something may have gone wrong



bpcal.BP0

Table specific parameters

bp_tablename = “bpcal.BP0”

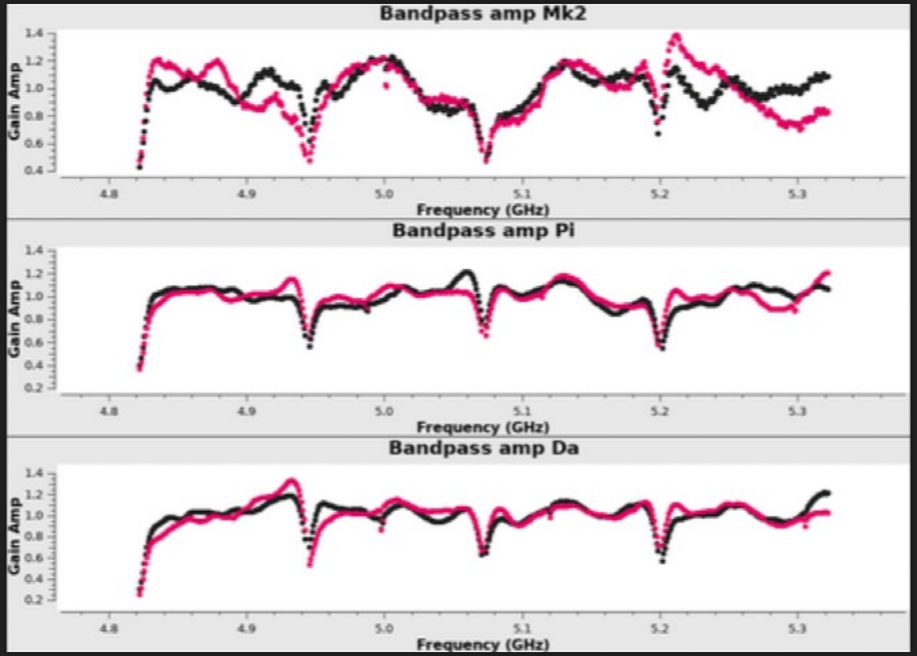
bp_solint = “inf”

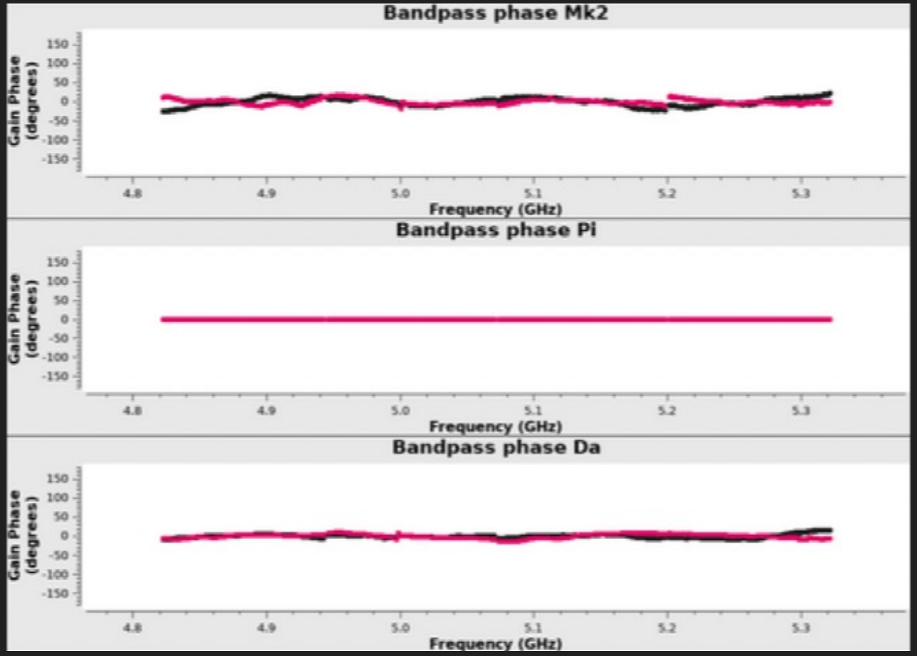
bp_combine = “field,scan”

bp_prev_cal = [“bpcal_d.K0”, “bpcal_p.G0” ”, “bpcal_ap.G0”]

bp_interp = “nearest,cubicflag”

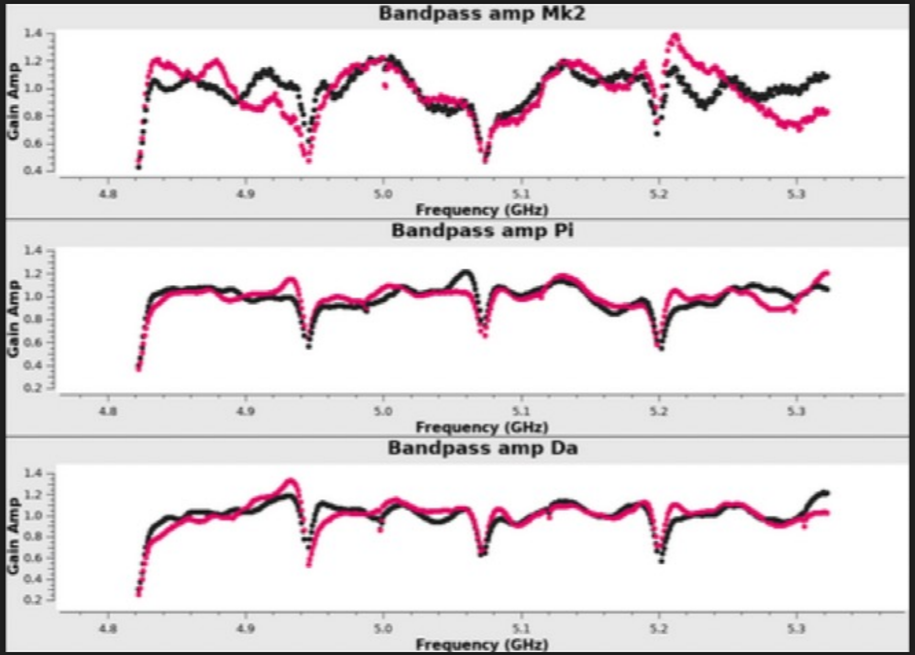
bp_spw = [“*”,””]

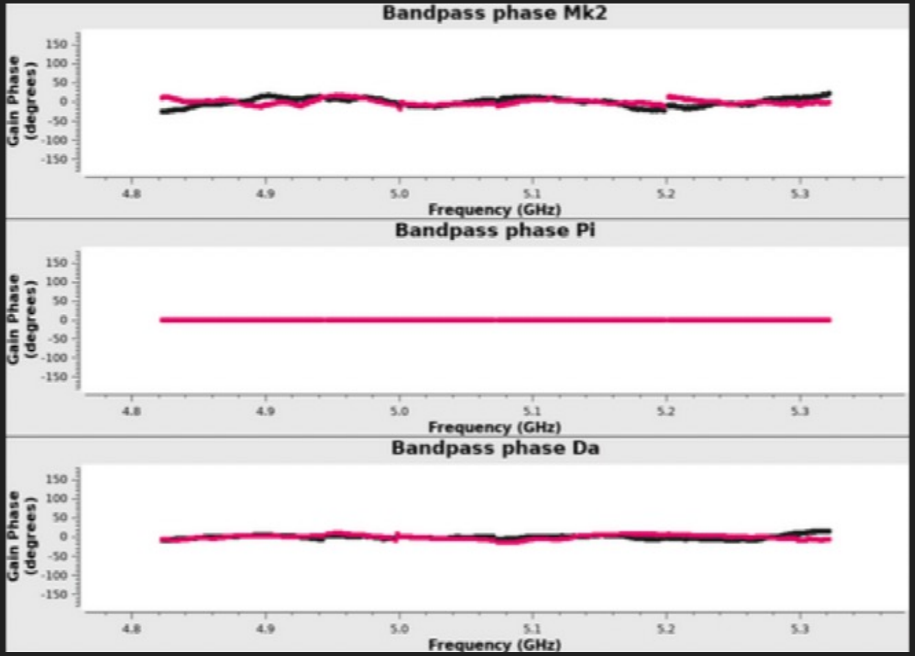




bpcal.BP0

- Two plots created: amplitude (top) and phase (bottom)
- Amplitude + phase plots should show agreement between both polarisations
- Amplitude should show band shape, including spws and band edge roll over and be roughly ~1
- Phase plot should be flat with the occasional discontinuity due to spw edges and should be roughly ~ 0





Troubleshooting the bandpass step

- The bandpass step is usually pretty robust, as OQ208 is bright and compact.
- But if you don't have enough data (more than ~10 mins on source) you may not get good solutions
- Try moving minsnr parameters to lower values, or minblperant to 2, in the case where you have poor data or have lost a few baselines for the bandpass calibrator
- Make a note of any regions where the bandpass calibrator phase or amplitude seems unreasonable, for example in this dataset we may want to consider the amplitude drop on Darnhall during the bpcal first scan
- At the end of this step, the solution tables are applied to the calibrators so that we can perform calibration procedures with this preliminary bandpass taken into account

initial_gaincal

- The initial_gaincal stage calibrates all the calibrators using the bandpass table derived in the previous step
- It will estimate the delay, then perform phase, amplitude+phase corrections,
- The step will then flag the calibrators, delete the tables and re-run the above in its entirety
- This ensures that a bright RFI missed by aoflagger does not affect the phase calibrator

Table specific parameters to be described in next slides

use_fringefit = false

delay_cal = “default”

zerorates = true

*_minblperant = 3

*_minsnr = 2

apply_calibrators = [”bpcal.BP0”, “allcal_d.K1”, “allcal_p.G1”, “allcal_ap.G1”]

apply_targets = []

flagmode = “tfcrop”

allcal_d.K1

Table specific parameters

tablename = “allcal_d.K1”

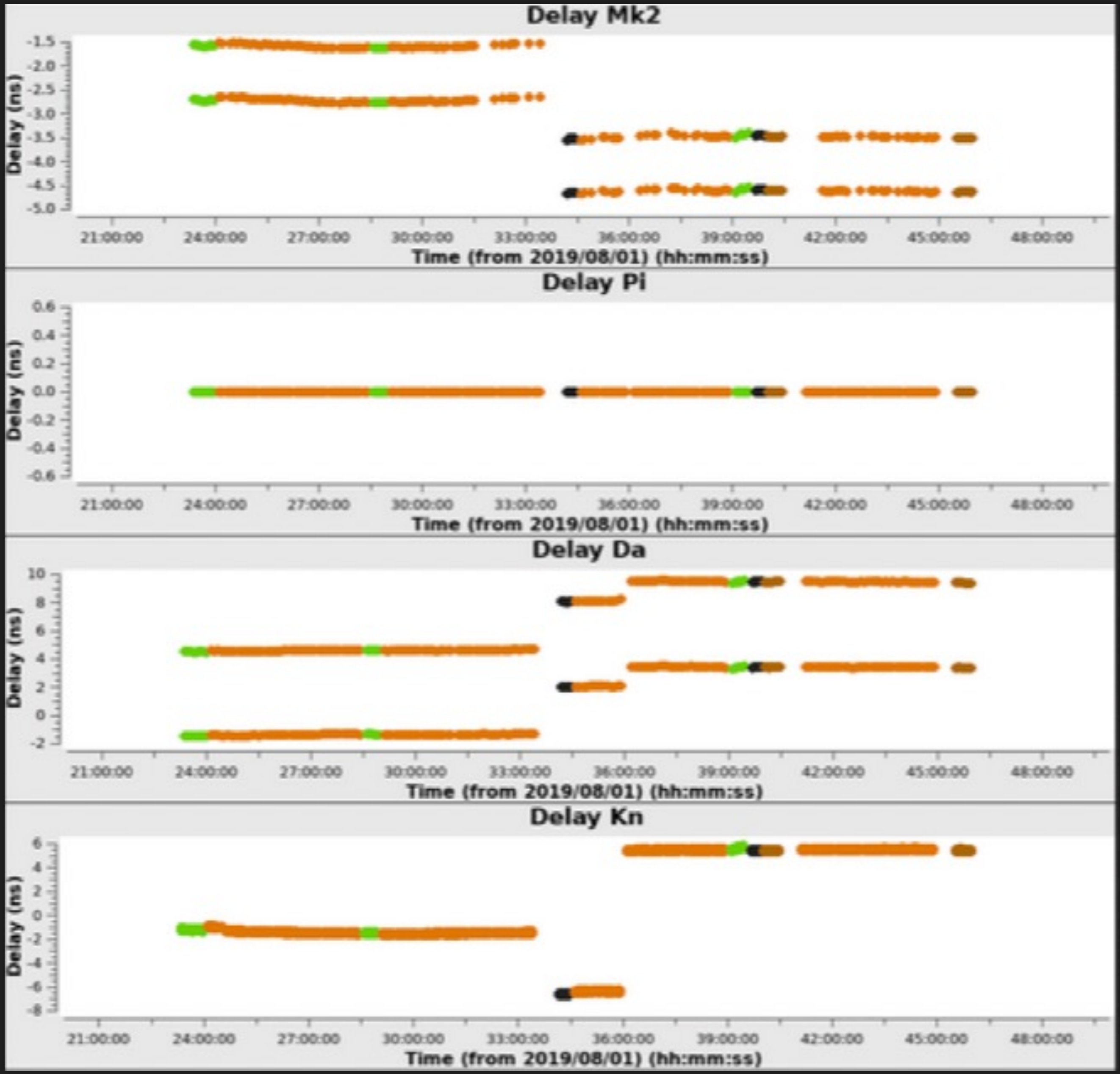
solint = “180s”

combine = “spw”

prev_cal = [“bpcal.BP0”]

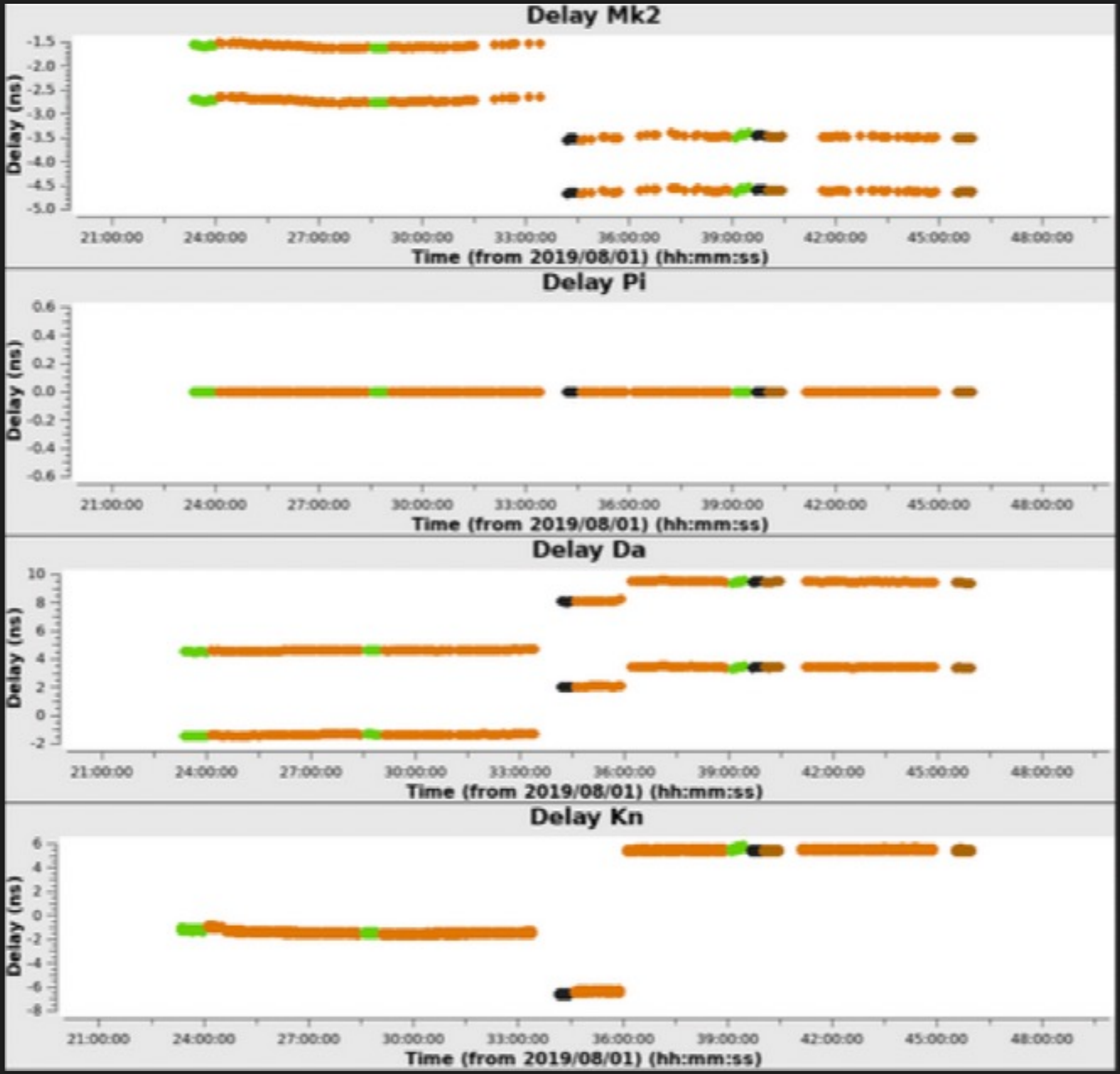
interp = “linear”

spw = [“*”,”innerchan”]



allcal_d.K1

- Refant will have a flat delay
- Check that other antennas flat relative to the refant
- They can be offset by several ns, and the polarisations can also be offset from each other
- Delay jumps are fine
- Variable delay rates are not fine



allcal_p.G1

Table specific parameters

p_tablename = “allcal_p.G1”

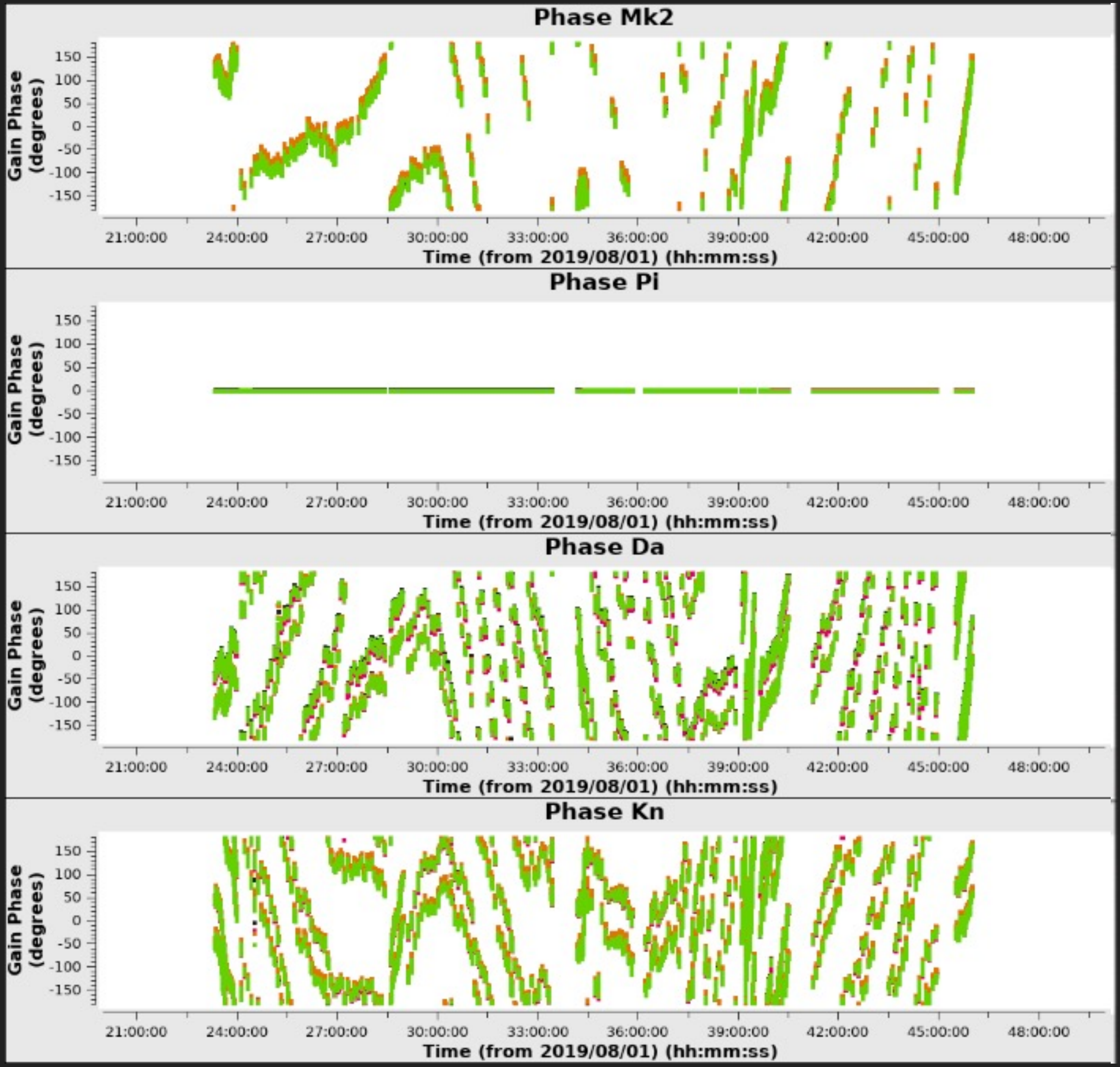
phase_solint = “int”

phase_combine = “”

phase_prev_cal = [“bpcal.BP0”,”allcal_d.K1”]

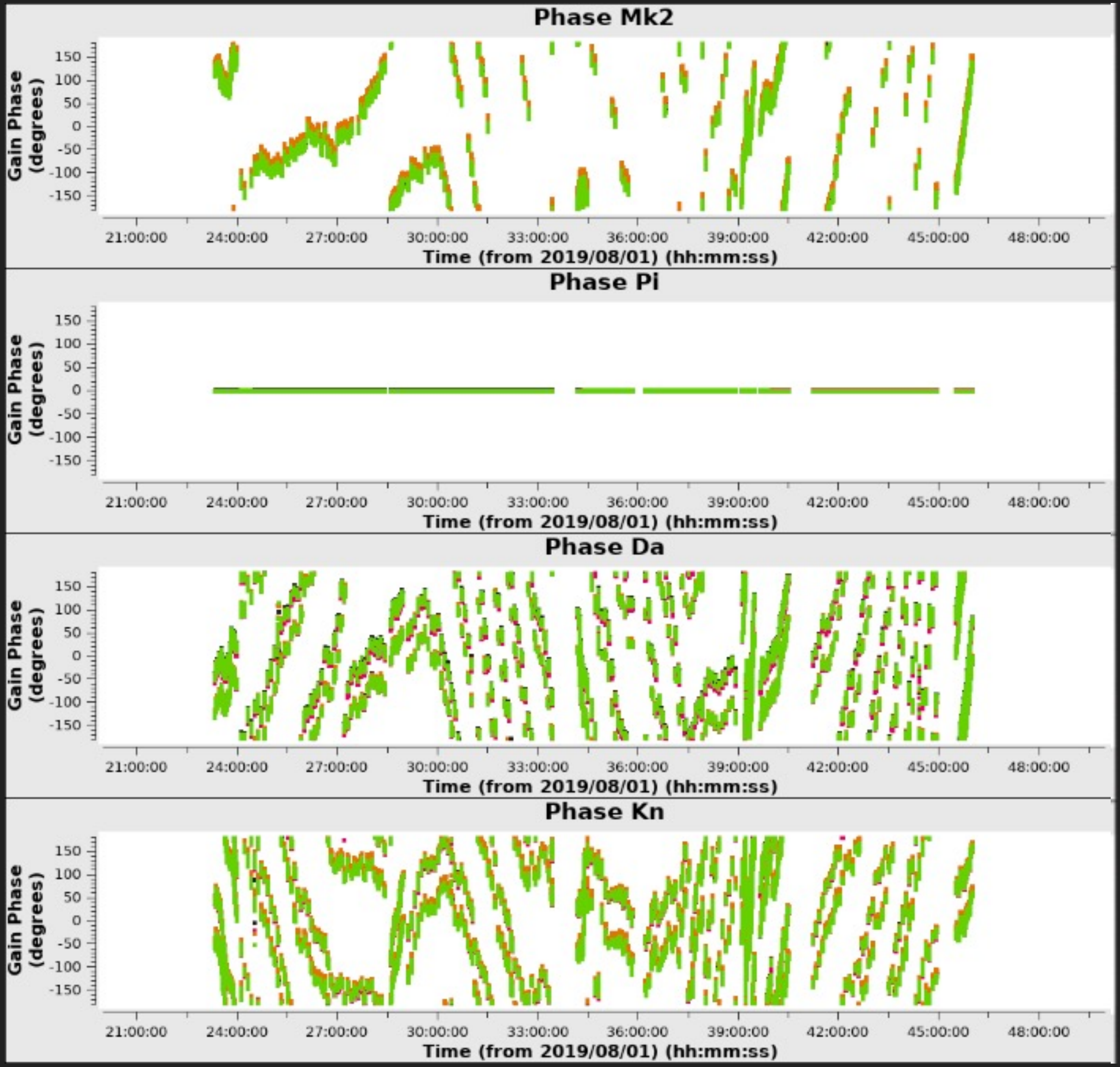
phase_interp = “linear”

phase_spw = [“*”,”innerchan”]



allcal_p.G1

- Refant will have a flat phase
- Phase should evolve slowly over time
- Phase wrapping (over 360 degrees) is *very likely* but you should be able to see a slowly evolving phase signal
- Areas with a vertical line suggest phase errors which should be checked/flagged



allcal_ap.G1

Table specific parameters

ap_tablename = “allcal_ap.G1”

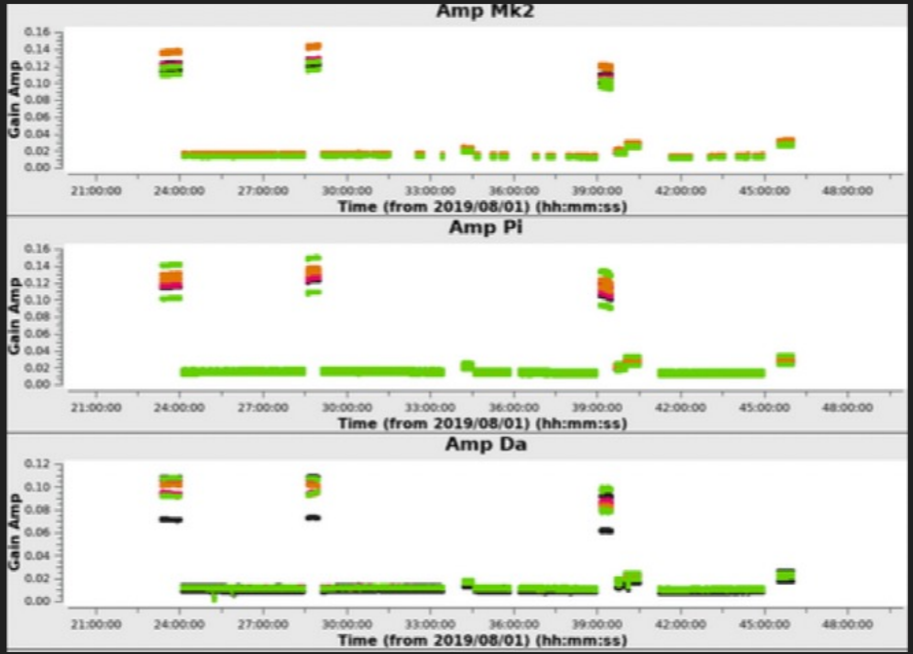
ap_solint = “32s”

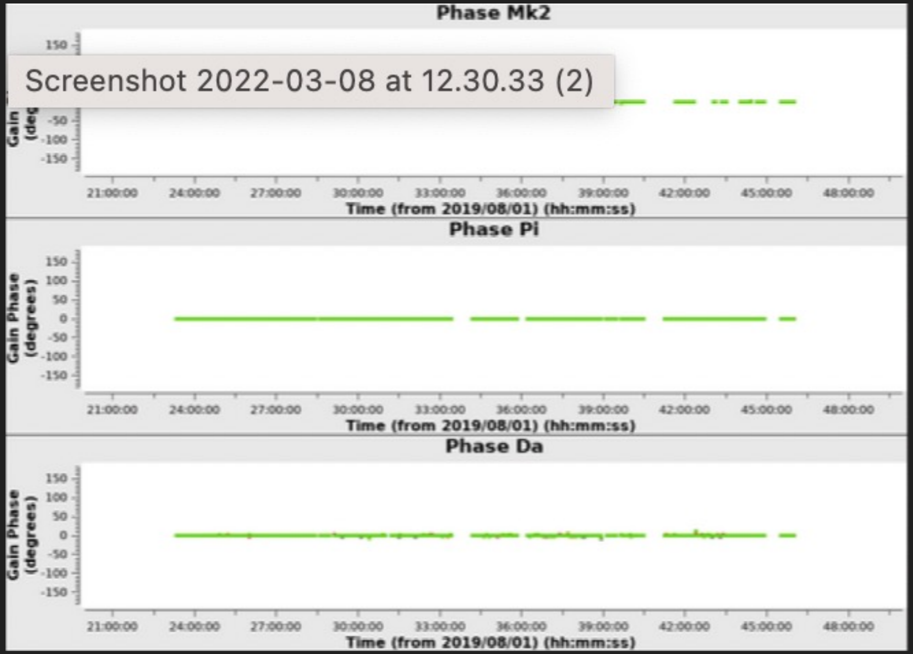
ap_combine = “”

ap_prev_cal = [“bpcal.BP0”, “allcal_d.K1”,”allcal_p.G1”]

ap_interp = “linear”

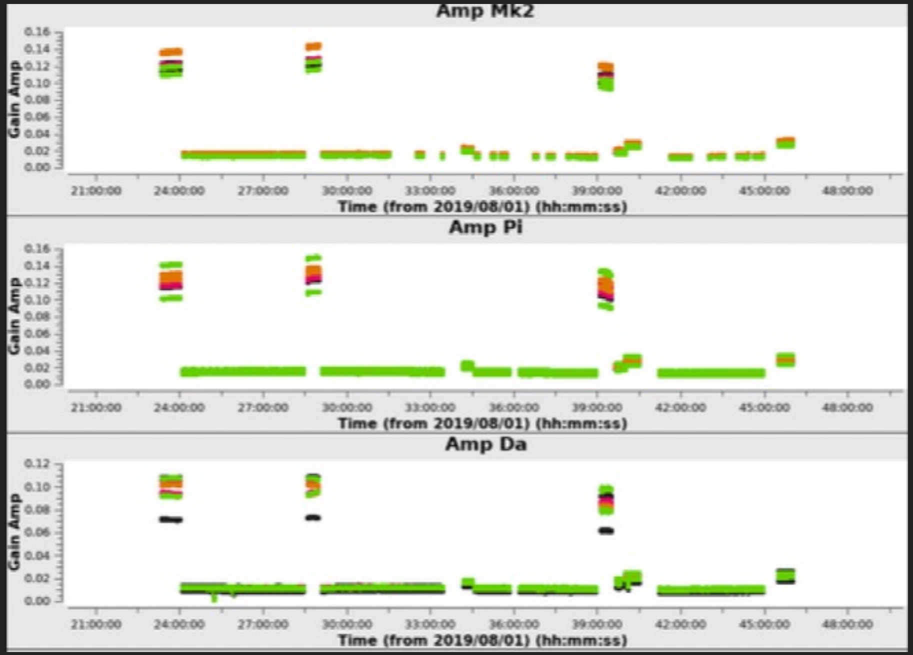
ap_spw = [“*”,”innerchan”]

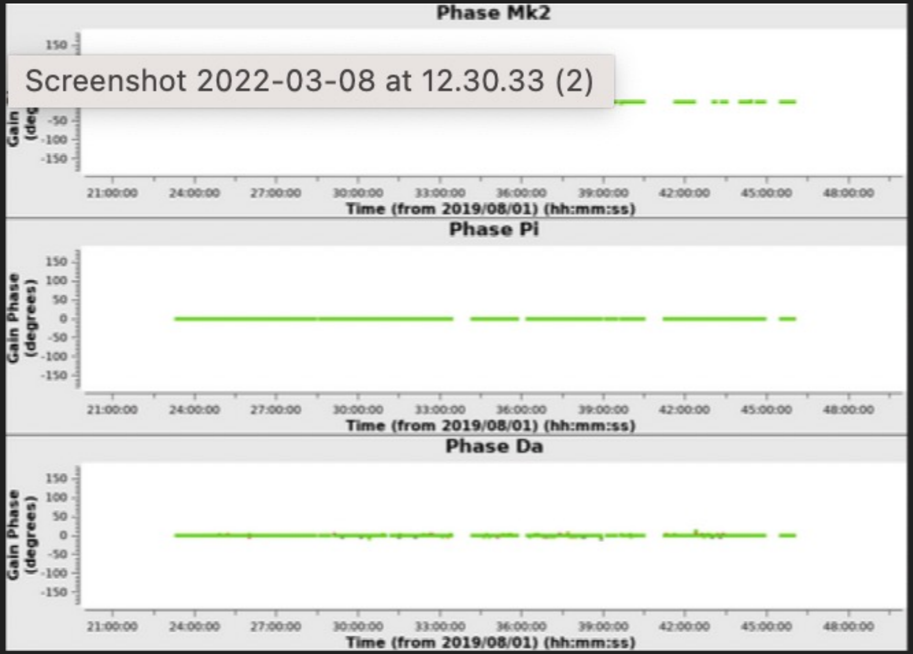




allcal_ap.G1

- Two plots created: amplitude (top) and phase (bottom)
- Phase corrections applied from previous table mean the phases should be all zero here
- Amplitude will appear to jump but this is fine – it reflects the different calibrator source signals
- Look out for variable amplitudes or jumps *in the same calibrator source*– something may have gone wrong





Troubleshooting the initial_gaincal step

- The initial_gaincal step can go awry due to over-flagging of solutions by CASA for the phase cal.
- If this appears to be the case, then try increasing the solution intervals in the p and ap tables
- Try moving minsnr parameters to lower values, or minblperant to 2, in the case where you have poor data or have lost a few baselines for the phase calibrator
- You can also try combining some of the data, like by spw, but this will reduce what you can do later on in the pipeline.

fluxscale

- The fluxscale stage will take the previous solution tables and set the fluxes for all calibrators using the model for 3c286
- It will return a plot with a per spw flux for each calibrator, as well as a fit, including a spectral index and flux density for the sources

tablename = "allcal_ap.G1_fluscaled"

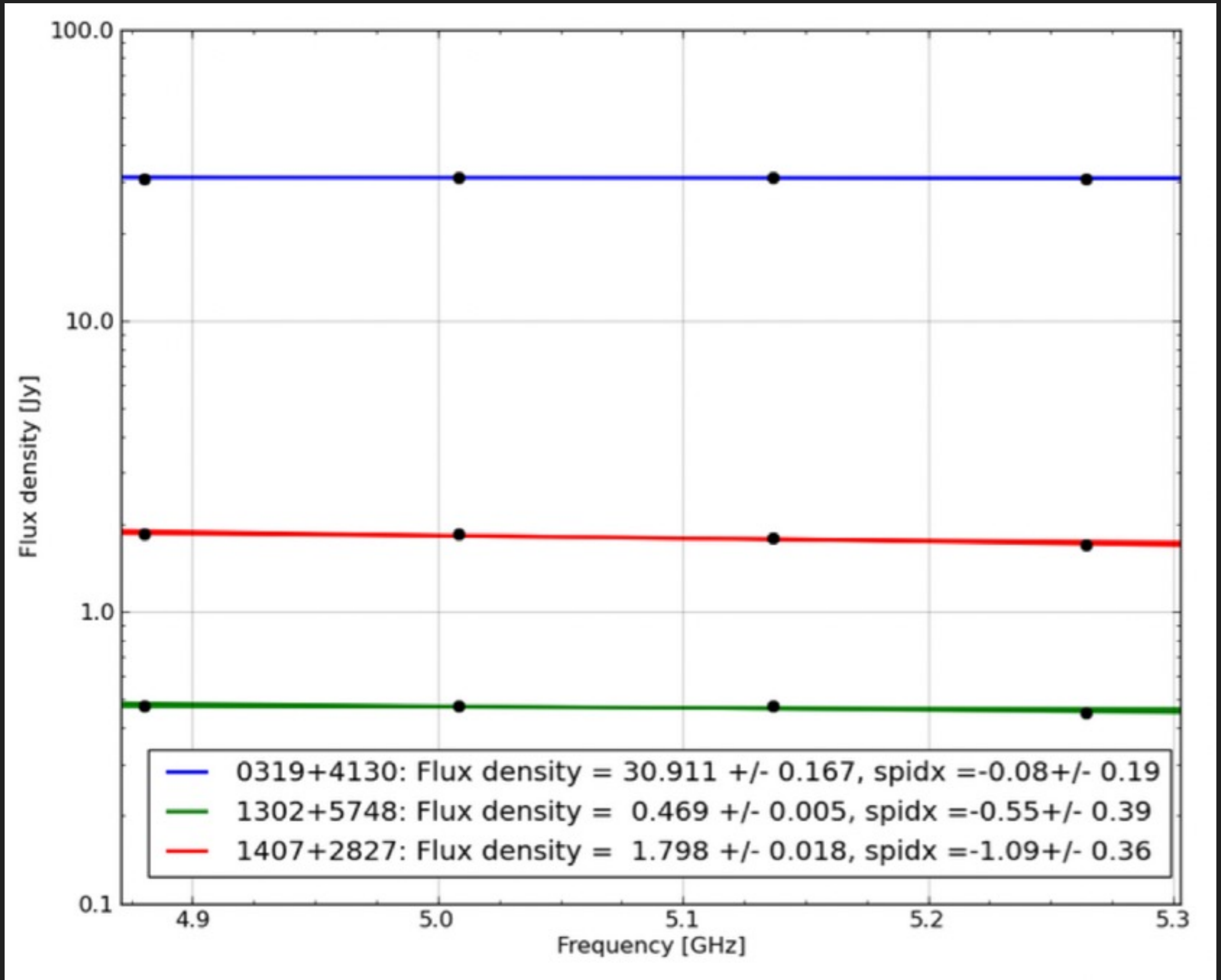
ampcal_table = "allcal_ap.G1"

apply_calibrators = ["bpcal.BP0", "allcal_d.K1", "allcal_p.G1", "allcal_ap.G1_fluscaled"]

apply_targets = []

allcal_ap.G1_fluxscaled

- The plot on the right shows the fits and fluxes of the calibrator fields in the measurement set, bootstrapped from the flux calibrator 3c286.



allcal_ap.G1_fluxscaled

- The text on the right is listed in the weblog and in the CASA logs. It is the full set of flux values calculated by the pipeline including the eMfactor. Note that I have only shown the phase calibrator information here.

CASA fluxscale output (not corrected by eMfactor):

```
# Flux density for 1302+5748 in SpW=0 (freq=4.8805e+09
Hz) is: 0.477934 +/- 0.0480495 (SNR = 9.9467, N = 12)

# Flux density for 1302+5748 in SpW=1 (freq=5.0085e+09
Hz) is: 0.480012 +/- 0.0514218 (SNR = 9.3348, N = 12)

# Flux density for 1302+5748 in SpW=2 (freq=5.1365e+09
Hz) is: 0.479019 +/- 0.0537081 (SNR = 8.91894, N = 12)

# Flux density for 1302+5748 in SpW=3 (freq=5.2645e+09
Hz) is: 0.453396 +/- 0.0575073 (SNR = 7.88416, N = 12)


# Fitted spectrum for 1302+5748 with fitorder=1: Flux
density = 0.472813 +/- 0.00518869 (freq=5.07048 GHz)
spidx: a_1 (spectral index) =-0.546973 +/- 0.390837
covariance matrix for the fit:  covar(0,0)=0.00313254
covar(0,1)=0.0414962 covar(1,0)=0.0414962
covar(1,1)=21.066

# WARNING: All flux densities in this file need to be
multiplied by eMfactor=0.9924 to match the corrections
that have been applied to the data.
```

Troubleshooting the fluxscale step

- The fluxscale step requires enough good data on the phase calibrator and calibrator fields to succeed – it can therefore be quite brittle.
- Look out for clearly erroneous fluxes or spectral indices for the calibrators
- If this part fails, try re-running previous steps to allow more data to pass, i.e. by reducing minsnr, minblperant, or, changing the global parameters from ”calflagstrict” to “calflag”
- If 3c286 is the problem, then you may need to use one of your other bright calibrators, like 3c84 as a manual flux calibrator instead

Bandpass_final

- The bandpass_final stage takes the delay, phase and ap solutions from the initial bandpass, and re-calculates them using the spectral index derived from running fluxscale.

Table specific parameters to be described in next slides

bp_tablename = “bpcal.BP2”

bp_prev_cal = ["bpcal_d.K0", "bpcal_p.G0", "bpcal_ap.G0"]

bp_solint = “inf”

bp_spw = [“*”,””]

bp_combine = “nearest,cubicflag”

bp_uvrange = “”

bp_fillgaps = 8

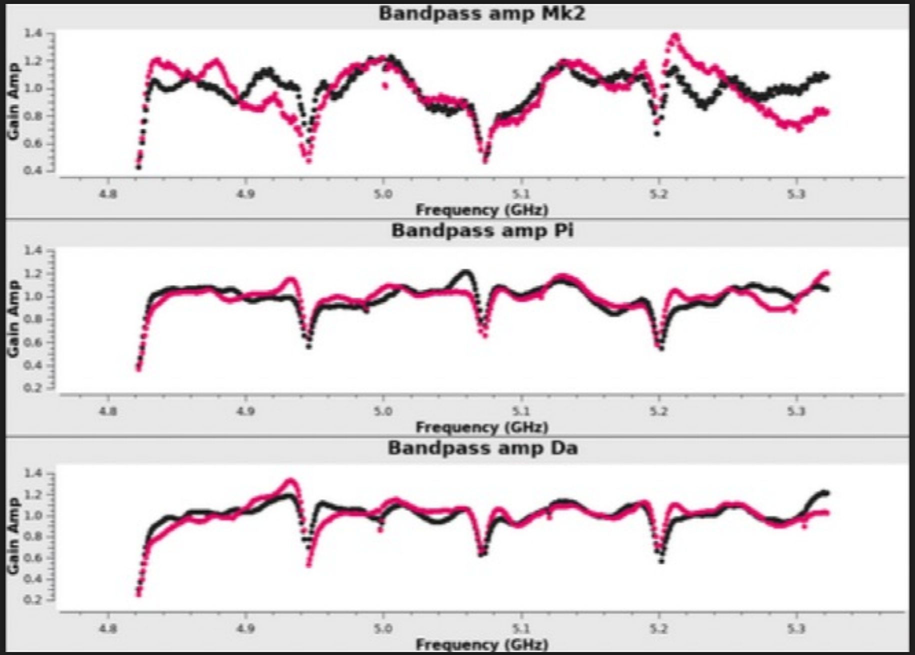
bp_solnorm = true

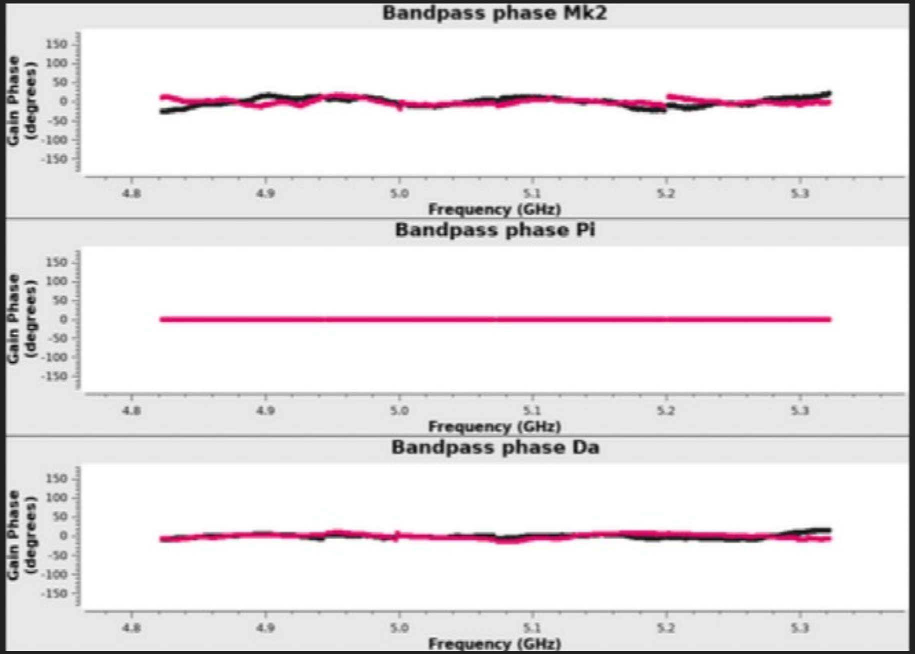
apply_calibrators = [“allcal_d.K0”,”bpcal_p.G0”,” bpcal_ap.G0”,”bpcal.BP2”]

apply_targets = []

bpcal.BP2

- Two plots created: amplitude (top) and phase (bottom)
- Amplitude + phase plots should show agreement between both polarisations
- Amplitude should show band shape, including spws and band edge roll over
- Phase plot should be flat with the occasional discontinuity due to spw edges





gaincal_final

- The gaincal_final stage will take our spectral index dependent bandpass table (BP2) and re-derive the phase and ap solutions for all calibrators, using the delay solutions found earlier.
- This step will also produce a per scan solution table in both phase and ap for the phase calibrator. These "scan" tables will be applied to the target field in the next step.
- If you have a spectral line observation, it will also compute offset and a narrow band pass table for each zoom spectral window.

Table specific parameters to be described in next slides

*_minblperant = 3

*_minsnr = 2

ap_calibrator = “default”

ap_scan_calibrator = “phscals”

apply_calibrators = [“allcal_d.K1", "bpcal.BP2", "allcal_p.G3", "allcal_ap.G3”]

apply_targets = ["allcal_d.K1", "bpcal.BP2", "phscal_p_scan.G3", "phscal_ap_scan.G3"]

allcal_p.G3

Table specific parameters

p_tablename = “allcal_p.G1”

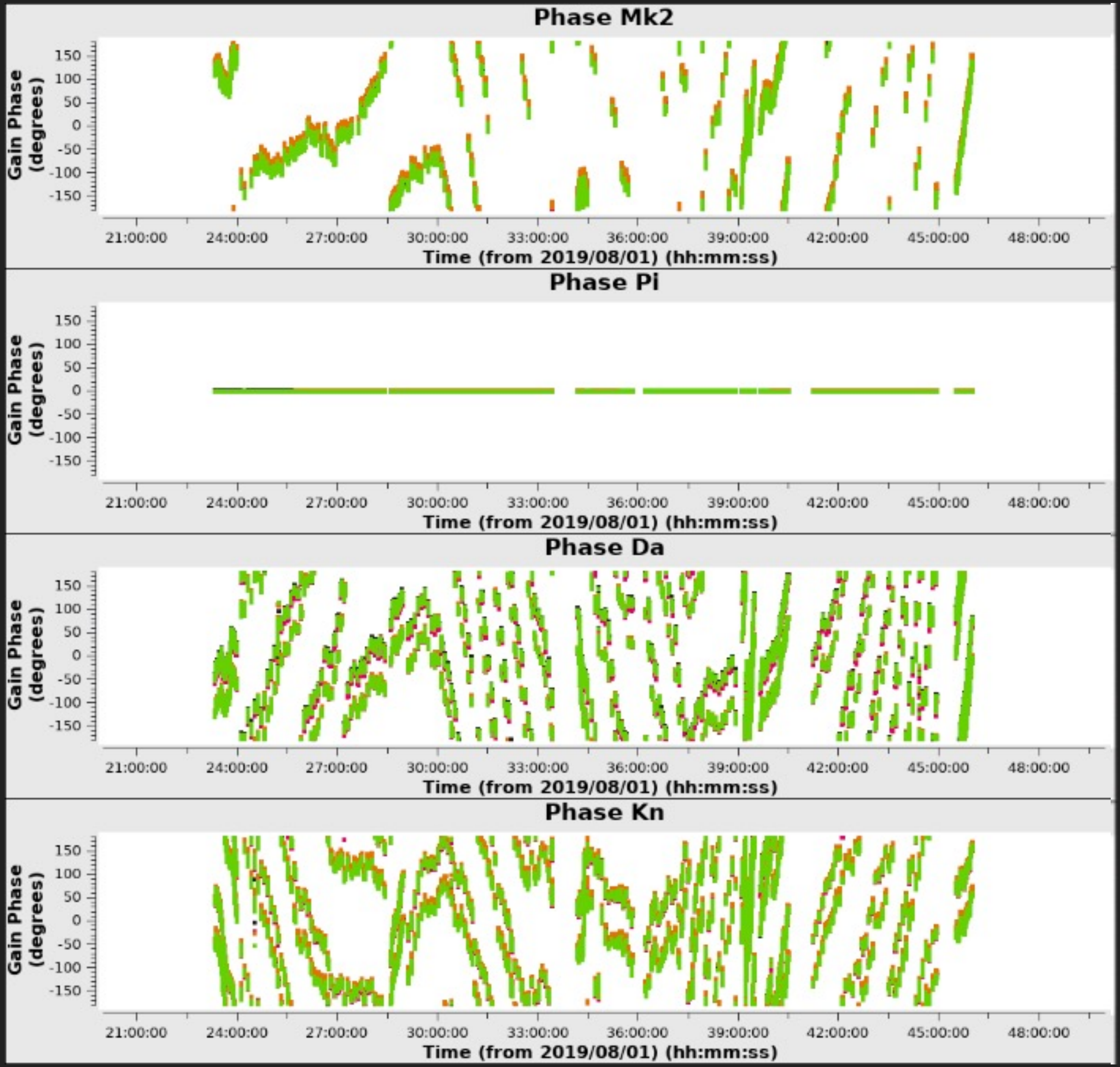
p_prev_cal = [“bpcal.BP2”,”allcal_d.K1”]

p_solint = “int”

p_spw = [“*”,”innerchan”]

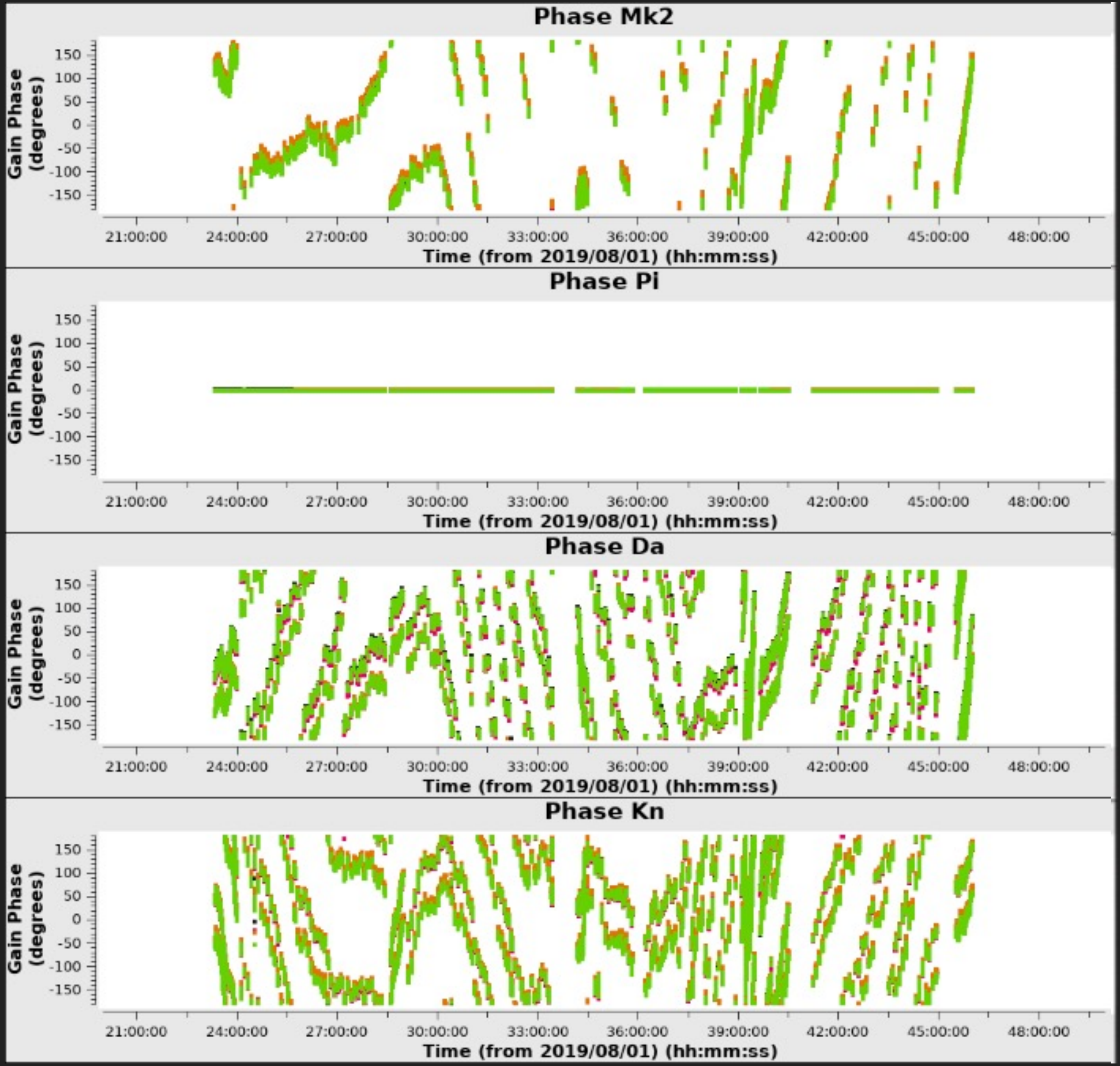
p_combine = “”

phase_interp = “linear”



allcal_p.G3

- Refant will have a flat phase
- Phase should evolve slowly over time
- Phase wrapping (over 360 degrees) is *very likely* but you should be able to see a slowly evolving phase signal
- Areas with a vertical line suggest phase errors which should be checked/flagged



allcal_ap.G3

Table specific parameters

ap_tablename = “allcal_ap.G3”

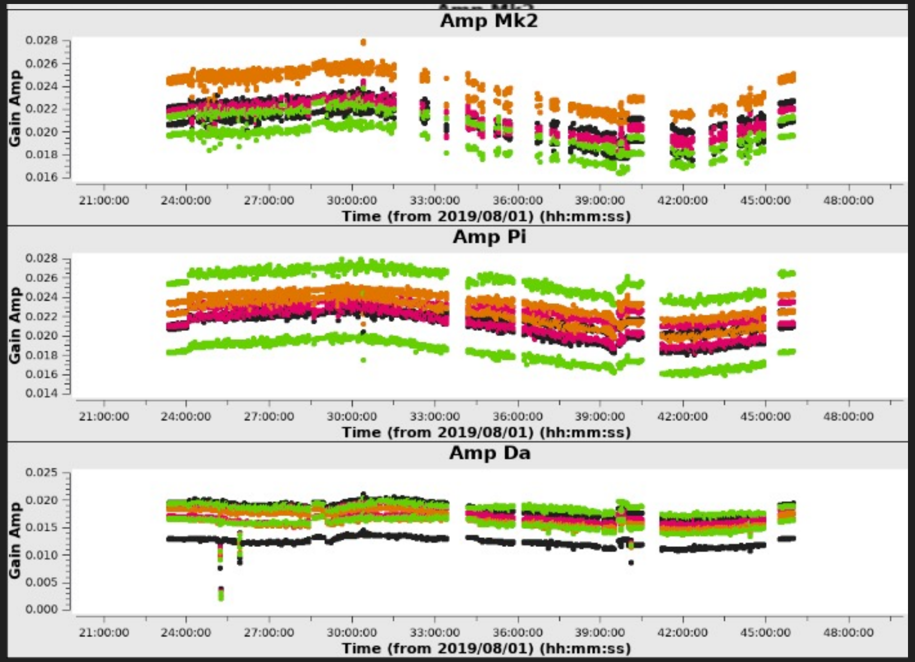
ap_prev_cal = [“bpcal.BP2”, “allcal_d.K1”,”allcal_p.G3”]

ap_solint = “32s”

ap_spw = [“*”,”innerchan”]

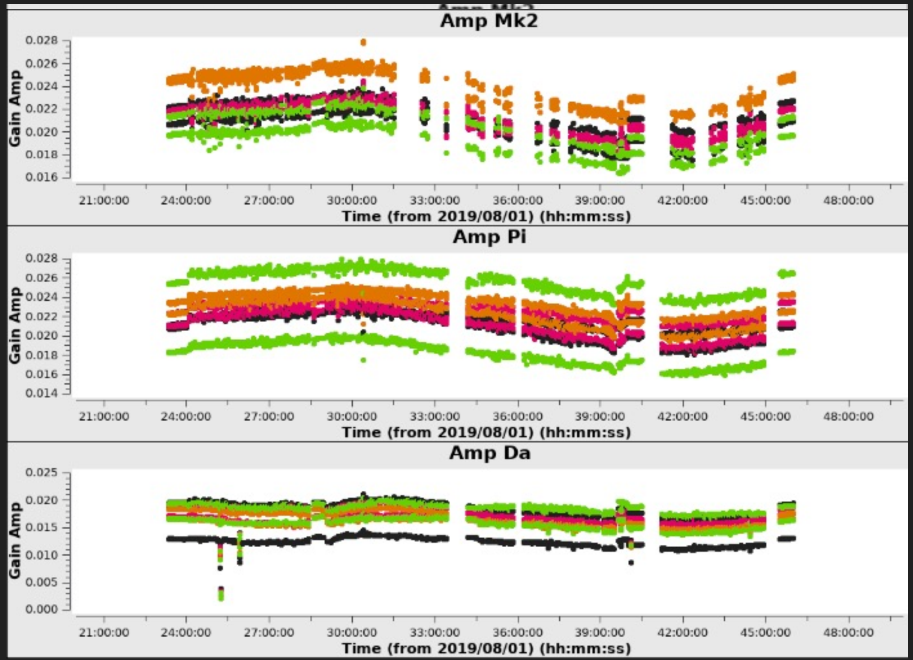
ap_combine = “”

ap_interp = “linear”



allcal_ap.G3

- Two plots created: amplitude (top) and phase (bottom)
- Phase corrections applied from previous table mean the phases should be all zero here
- Amplitude will appear to jump but this is fine – it reflects the different calibrator source signals
- Look out for variable amplitudes or jumps *in the same calibrator source*– something may have gone wrong



phscal_p_scan.G3

Table specific parameters

p_scan_tablename = phscal_p_scan.G3"

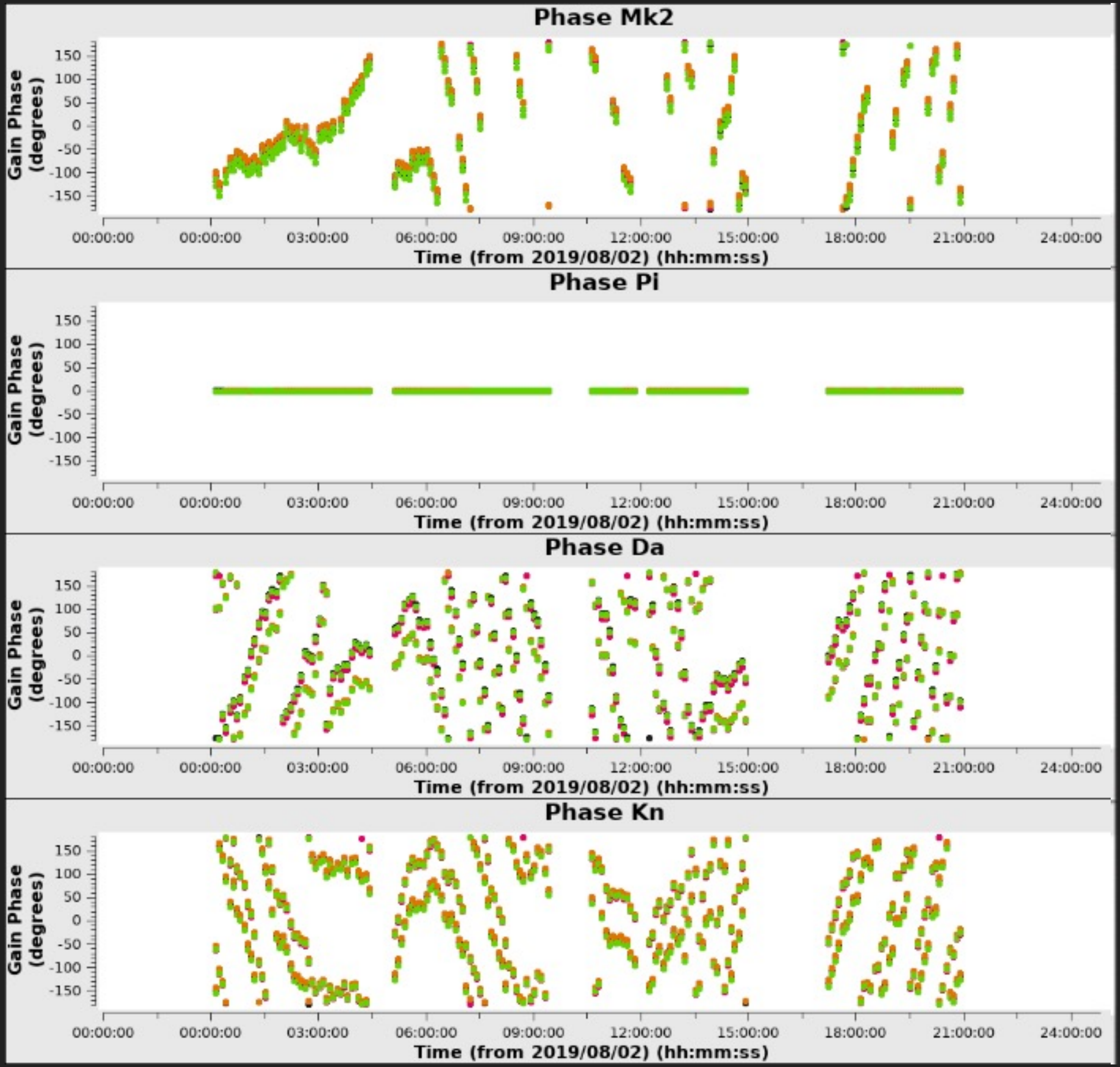
p_scan_prev_cal = ["bpcal.BP2","allcal_d.K1"]

p_scan_spw = ["*","innerchan"]

p_scan_solint = "int"

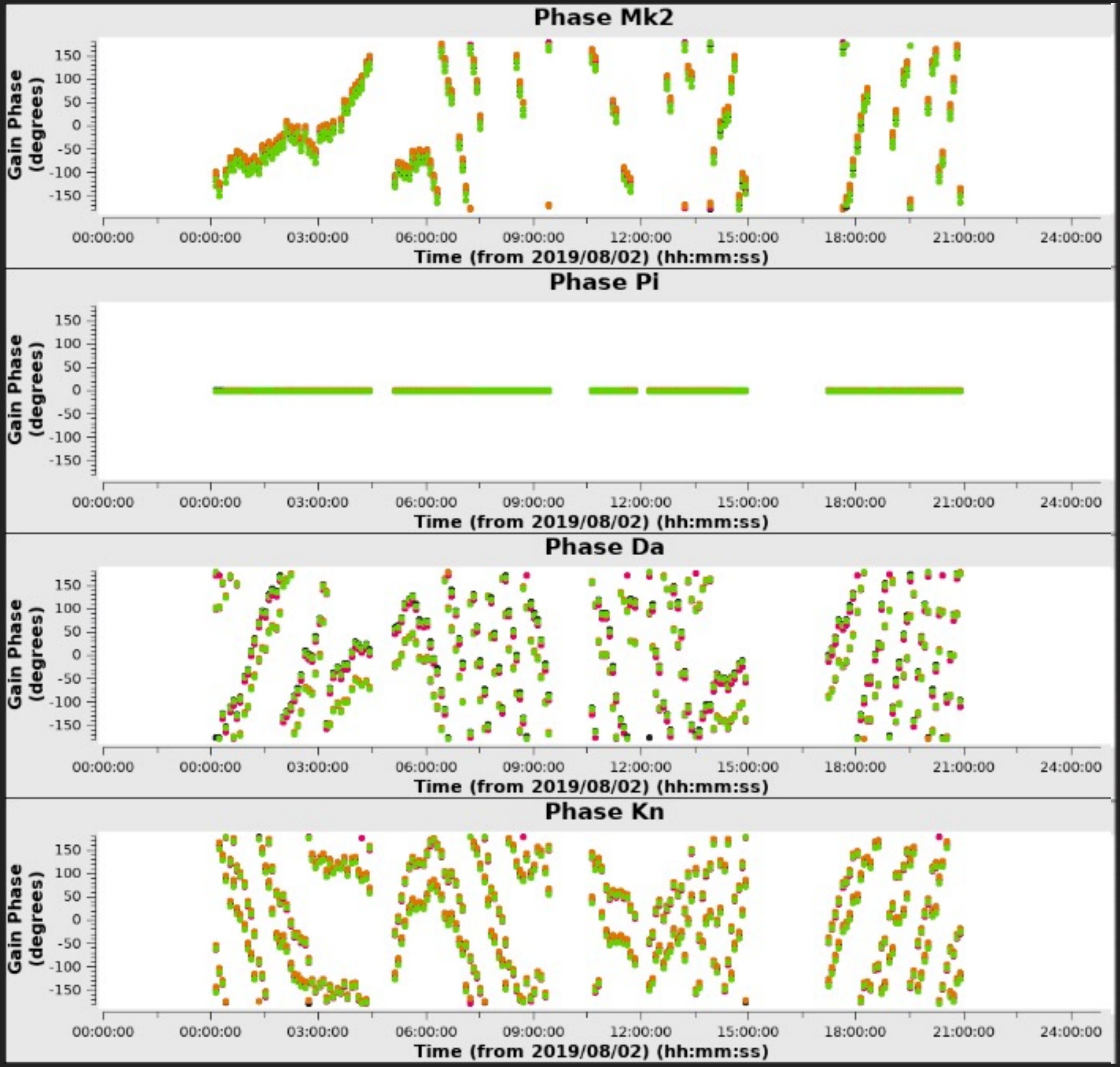
p_scan_combine = ""

p_scan_interp = "linear"



phscal_p_scan.G3

- Refant will have a flat phase
- Phase should evolve slowly over time
- Phase wrapping (over 360 degrees) is *very likely* but you should be able to see a slowly evolving phase signal
- Areas with a vertical line suggest phase errors which should be checked/flagged



phscal_ap_scan.G3

Table specific parameters

ap_scan_tablename = "phscal_ap_scan.G3"

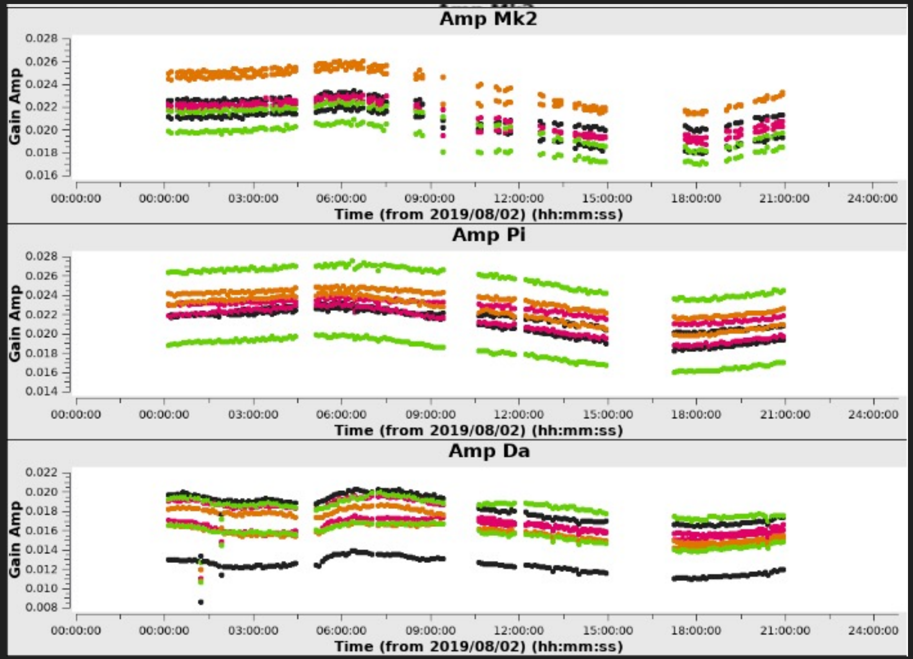
ap__scan_prev_cal = ["bpcal.BP2", "allcal_d.K1","allcal_p.G3"]

ap_scan_solint = "inf"

ap__scan_spw = ["*","innerchan"]

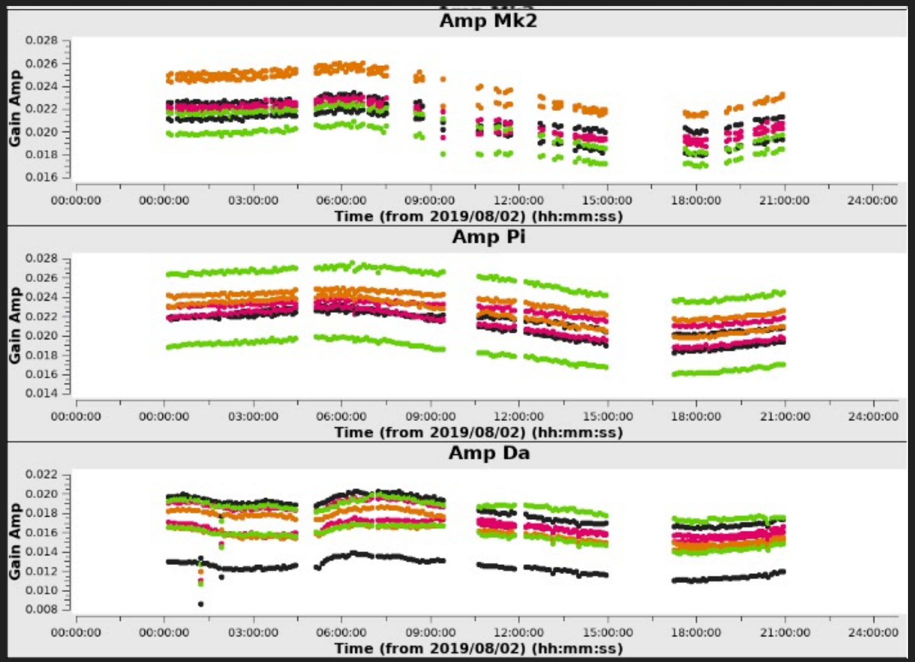
ap_scan_combine = ""

ap_scan_interp = "linear"



phscal_ap_scan.G3

- Two plots created: amplitude (top) and phase (bottom)
- Phase corrections applied from previous table mean the phases should be all zero here
- Amplitude should not follow smoothly across the observation
- Look out for variable amplitudes or jumps *in the same calibrator source*– something may have gone wrong



Troubleshooting the gaincal_final step

- Like the initial_gaincal step, this step can go awry due to over-flagging of solutions by CASA for the phase cal.
- If this appears to be the case, then try increasing the solution intervals in the p and ap tables
- Try moving minsnr parameters to lower values, or minblperant to 2, in the case where you have poor data or have lost a few baselines for the phase calibrator
- You can try combining solutions here too, but again it's not ideal unless absolutely necessary

Additional spectral line info for gaincal_final

- While not applicable for the 3c277.1 data, if your observations include a spectral zoom mode, then the eMCP will also derive the narrow bandpass solutions per each narrow spectral window, and, calculate the offset in phases between the narrow and continuum spws

*_minblperant = 3

*_minsnr = 2

narrow_bp_uvrange = “”

narrow_bp_fillgaps = 8

narrow_bp_solnorm = true

narrow_apply_calibrators = ["allcal_d.K1", "narrow_bpcal.BP2", "allcal_p.G3", "allcal_ap.G3", "narrow_p_offset.G3"],

narrow_apply_targets = ["allcal_d.K1", "narrow_bpcal.BP2", "phscal_p_scan.G3", "phscal_ap_scan.G3", "narrow_p_offset.G3"]

narrow_p_offset.G3

Table specific parameters

p_offset_tablename = "narrow_p_offset.G3"

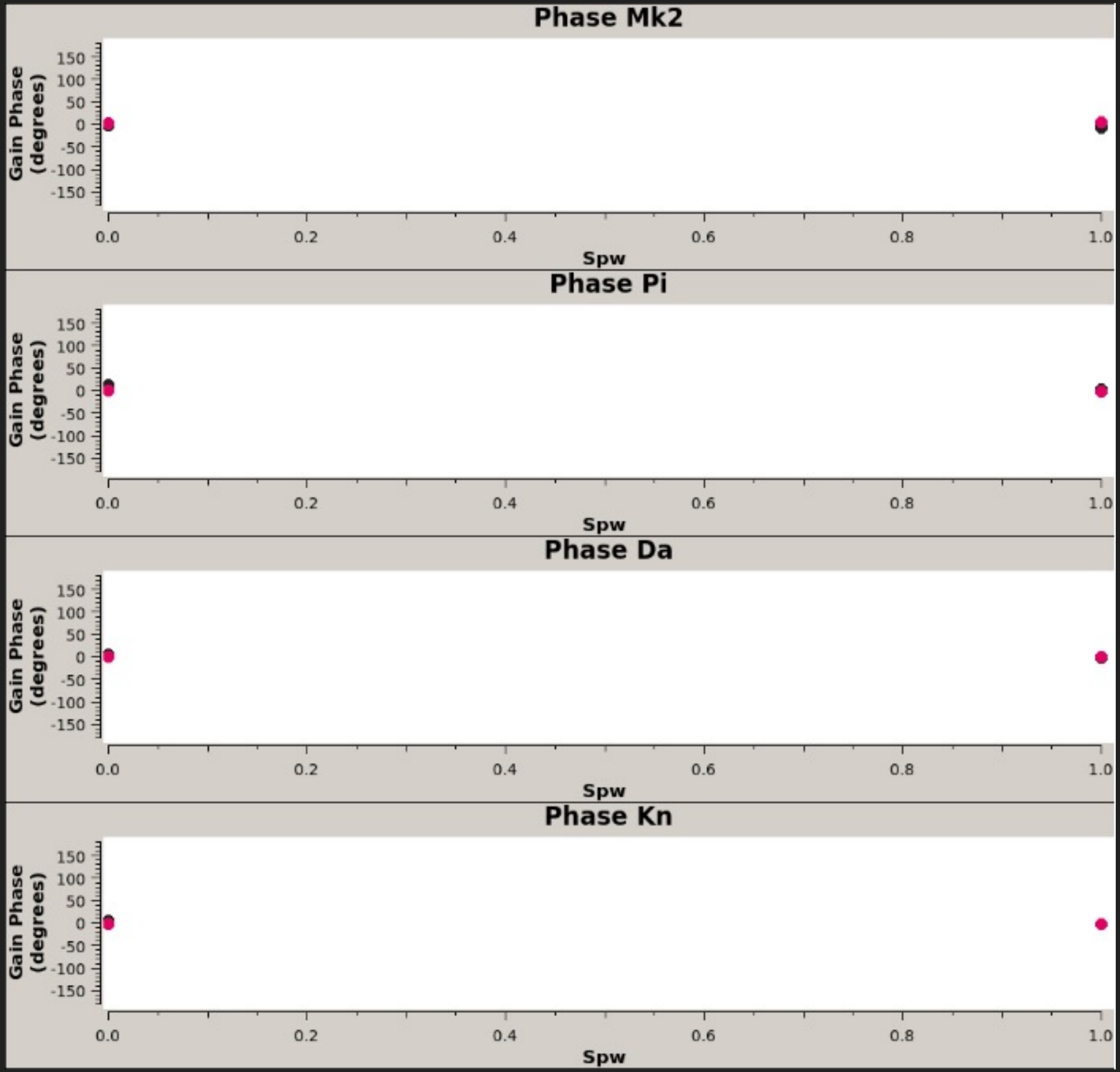
p_offset_prev_cal = ["allcal_d.K1","allcal_p.G3"]

p_offset_solint = "inf"

p_offset_spw = ["*","innerchan"]

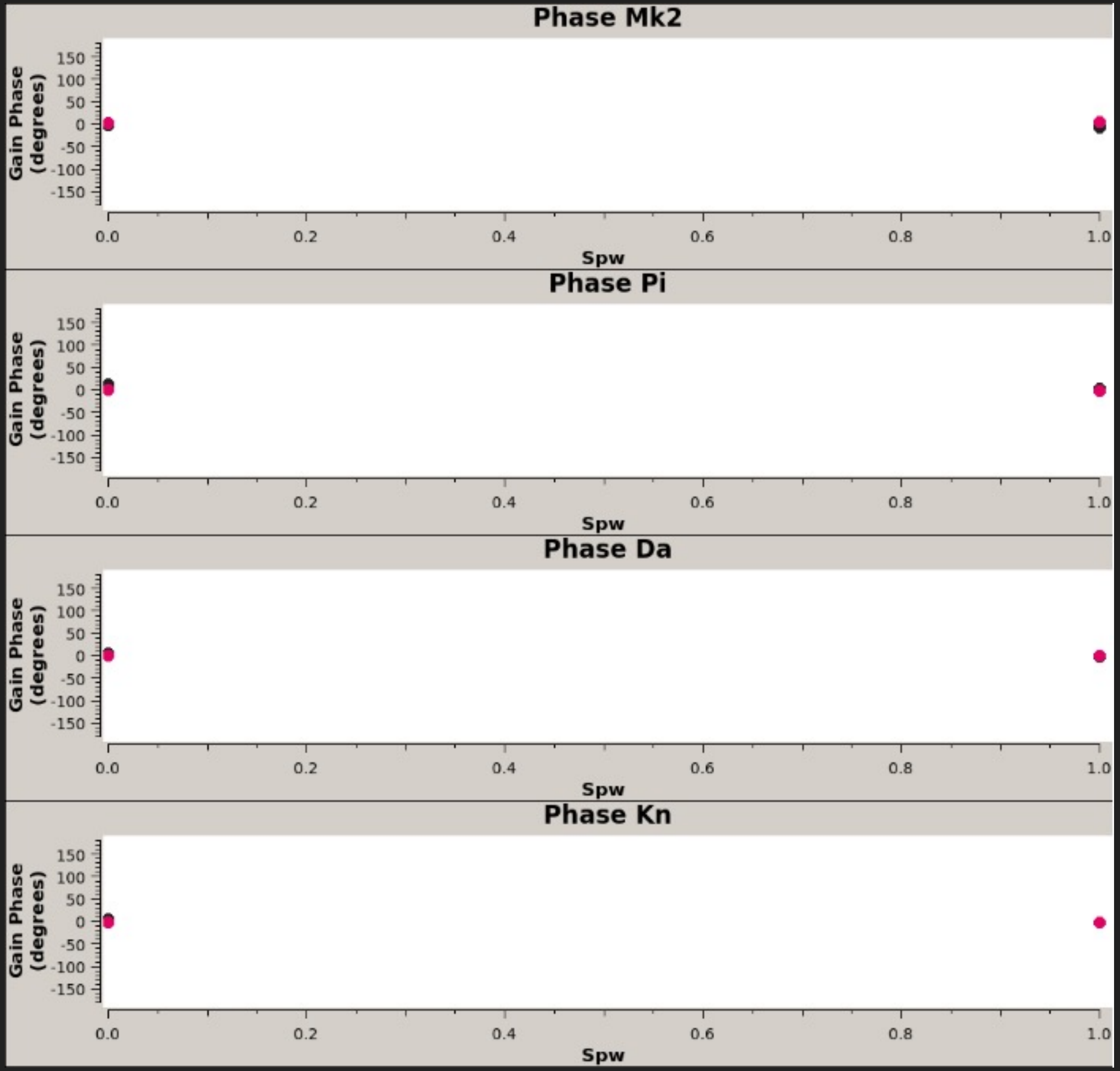
p_offset_combine = ""

p_offset_interp = "linear"



narrow_p_offset.G3

- This is a table to compute the phase offset of the narrow to continuum spectral windows.
- In this case the offset is close to but *crucially* not zero for both of the narrow spectral windows.



narrow_p_offset.G3

Table specific parameters

narrow_bp_tablename = "narrow_bpcal.BP2"

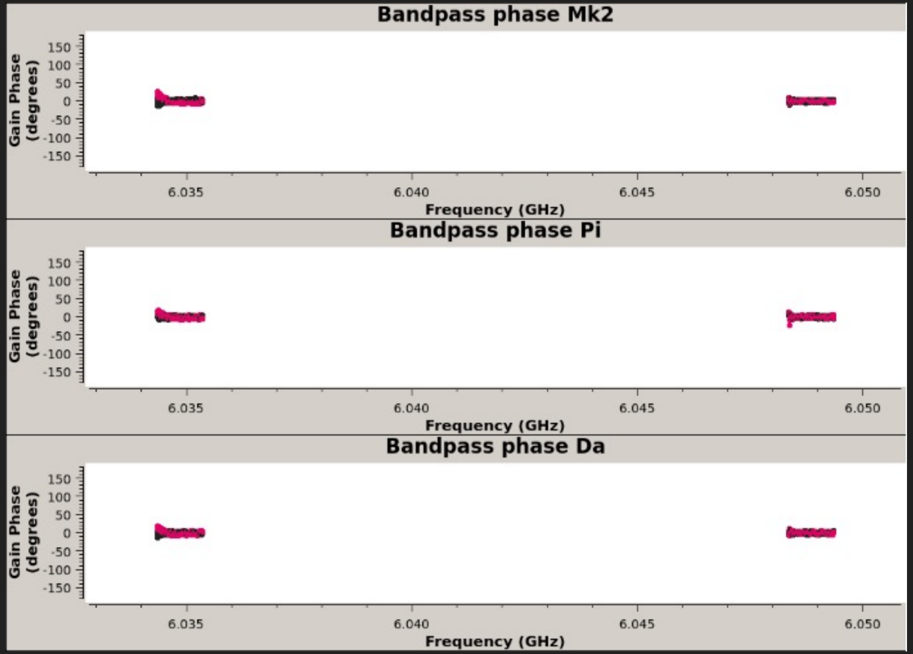
narrow_bp_prev_cal = ["allcal_d.K1", "allcal_p.G3", "allcal_ap.G3", "narrow_p_offset.G3"]

narrow_bp_solint = "inf"

narrow_bp_spw = ["*","innerchan"]

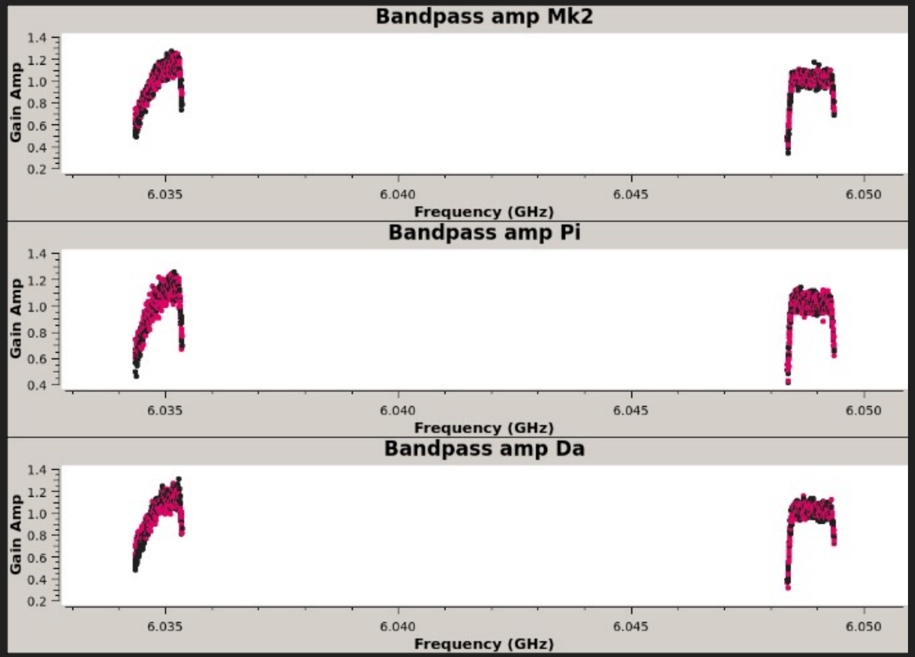
narrow_bp_combine = ""

narrow_bp_interp = "linear"



narrow_bpcal.BP2

- This is producing a band pass for each of the narrow spectral windows, so similar to the continuum band pass tables you should see a flat phase and a band structure for gains



applycal_all

- This is the final calibration stage of the pipeline (hurray!). It will take all of your solution tables and apply them to your data
- It will also re-weight the data using statwt.

apply_calibrators = ["allcal_d.K1", "bpcal.BP2", "allcal_p.G3", "allcal_ap.G3"],

apply_targets = ["allcal_d.K1", "bpcal.BP2", "phscal_p_scan.G3", "phscal_ap_scan.G3"],

apply_narrow_calibrators = ["allcal_d.K1", "narrow_bpcal.BP2", "allcal_p.G3", "allcal_ap.G3", "narrow_p_offset.G3"]

apply_narrow_targets = ["allcal_d.K1", "narrow_bpcal.BP2", "phscal_p_scan.G3", "phscal_ap_scan.G3", "narrow_p_offset.G3"]

run_statwt = true

statwt_timebin = 0.001s

Troubleshooting the applycal_all step

- This step is similar to the gaincal steps in that it could fail if there have been a lot of failed solutions previously. It is sometimes useful to revisit your previous calibration tables and check them if the pipeline fails here
- One additional thing to note that may not be obvious until the imaging stage is that statwt can sometimes cause issues with the data, leading to a green "blank" screen when imaging. This is due to statwt putting NaNs in the data. It is therefore worth running through to the imaging part of the pipeline after running applycal_all to check this

flag_target

- This is a final flagging step to now go and flag the target, having applied all of the calibration solutions from our calibrators.
- You can choose either to run tfcrop (the default) or rflag. Whichever one you choose the default parameters are similar to those in the CASA default parameters for tfcrop or rflag.

```
mode_to_run = ”rflag"
mode = "rflag"
sources = “targets"
antenna = ""
scan = ""
spw = ""
correlation = ""
ntime = "scan"
combinescans = false
datacolumn = "corrected"
timedevscale = 4.5
freqdevscale =4.5
extendflags = true
action = "apply"
display = ""
flagbackup = false
```

flag_target

- This is a final flagging step to now go and flag the target, having applied all of the calibration solutions from our calibrators.
- You can choose either to run tfcrop (the default) or rflag. Whichever one you choose the default parameters are similar to those in the CASA default parameters for tfcrop or rflag.

```
mode_to_run = "tfcrop"
mode = "tfcrop"
sources = "targets"
antenna = ""
scan = ""
spw = ""
correlation = ""
ntime =""
combinescans = false
datacolumn = "corrected"
winsize = 3
timecutoff = 4.5,
freqcutoff = 4.5,
maxnpieces = 7,
uwstats = "none",
halfwin = 1,
extendflags = true,
action = "apply",
display = "",
flagbackup = false
```

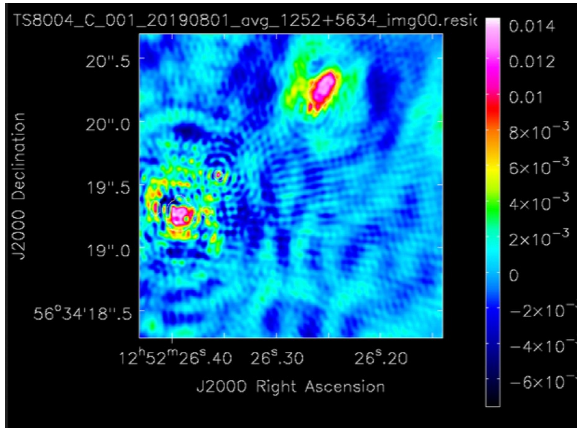
first_images

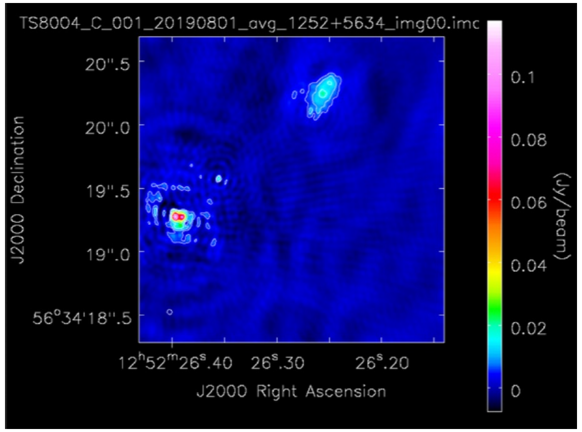
- Now to make some images of your sources and phase calibrator fields, which is what this step does
- It performs the deconvolution automatically and displays both the image and residual maps for the target and phase calibrator sources

Imsize = 1024
niter = 80
deconvolver = hogbom
nterms = 1
scales = [],
weighting = "briggs"
robust = 0.5
gain = 0.1
uvrange = "",
uvtaper = []
restoringbeam = []
nsigma = 5.0
sidelobethreshold = 1.0
noisethreshold = 8.0
lownoisethreshold = 1.5
minbeamfrac" = 0.2
growiterations = 25
parallel = true
level0 = 3.0
zoom_range_pix = 150

first_images

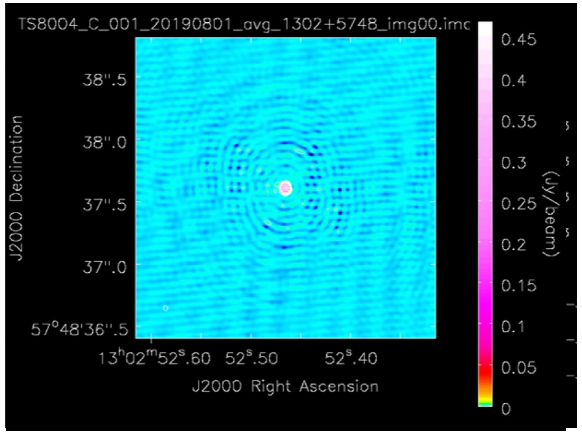
- Top image is the target field intensity map and bottom image is the residual image
- The pipeline automatically chooses an image contrast level and contours based upon the flux and noise in the image
- The peak flux and rms noise are stated in the weblogs
- Look out for ripples across the image (we will discuss this further later) that could be calibration errors
- Important to remember that this is auto-thresholded image with few iterations – it may not be a great image!

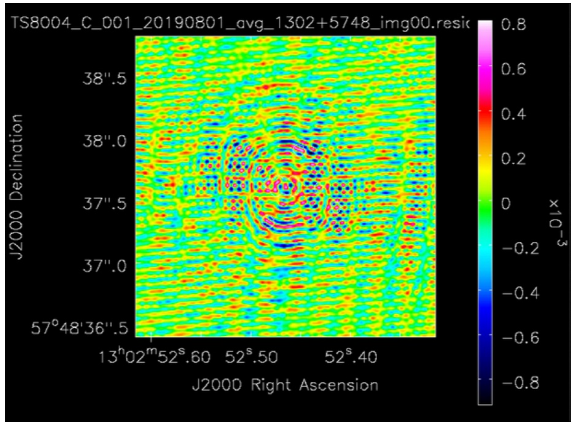




first_images

- Top image is the phase cal field intensity map and bottom image is the residual image
- The pipeline automatically chooses an image contrast level and contours based upon the flux and noise in the image
- The peak flux and rms noise are stated in the weblogs
- Look out for ripples across the image (we will discuss this further later) that could be calibration errors
- Important to remember that this is auto-thresholded image with few iterations – it may not be a great image!





split_fields

- After imaging your data, the eMCP will split out the target fields by default into their own measurement sets
- It does some averaging on these split datasets, so that you can quickly inspect and re-image them later
- All of the split measurement sets are placed in the “splits” directory.

```
fields = “targets"
timeaverage = true
timebin = "8s"
chanaverage = true,
chanbin = 2,
datacolumn = "corrected",
createmms = false,
output_dir = "./splits"
```

Any Questions?

Inspecting the weblogs and pipeline outputs

What to do after running the pipeline

- Generally, it’s a good idea to look at the weblogs straight away, before inspecting the data directly.
- As a support scientist, I usually start at the images and work my way backwards through the tabs, trying to find bad data
- The Calibrated UV Plots are incredibly useful and overlooked part of the pipeline weblog outputs which can highlight issues with data quickly
- We will now go through the weblog outputs and inspect the data with a live demonstration of what to look for

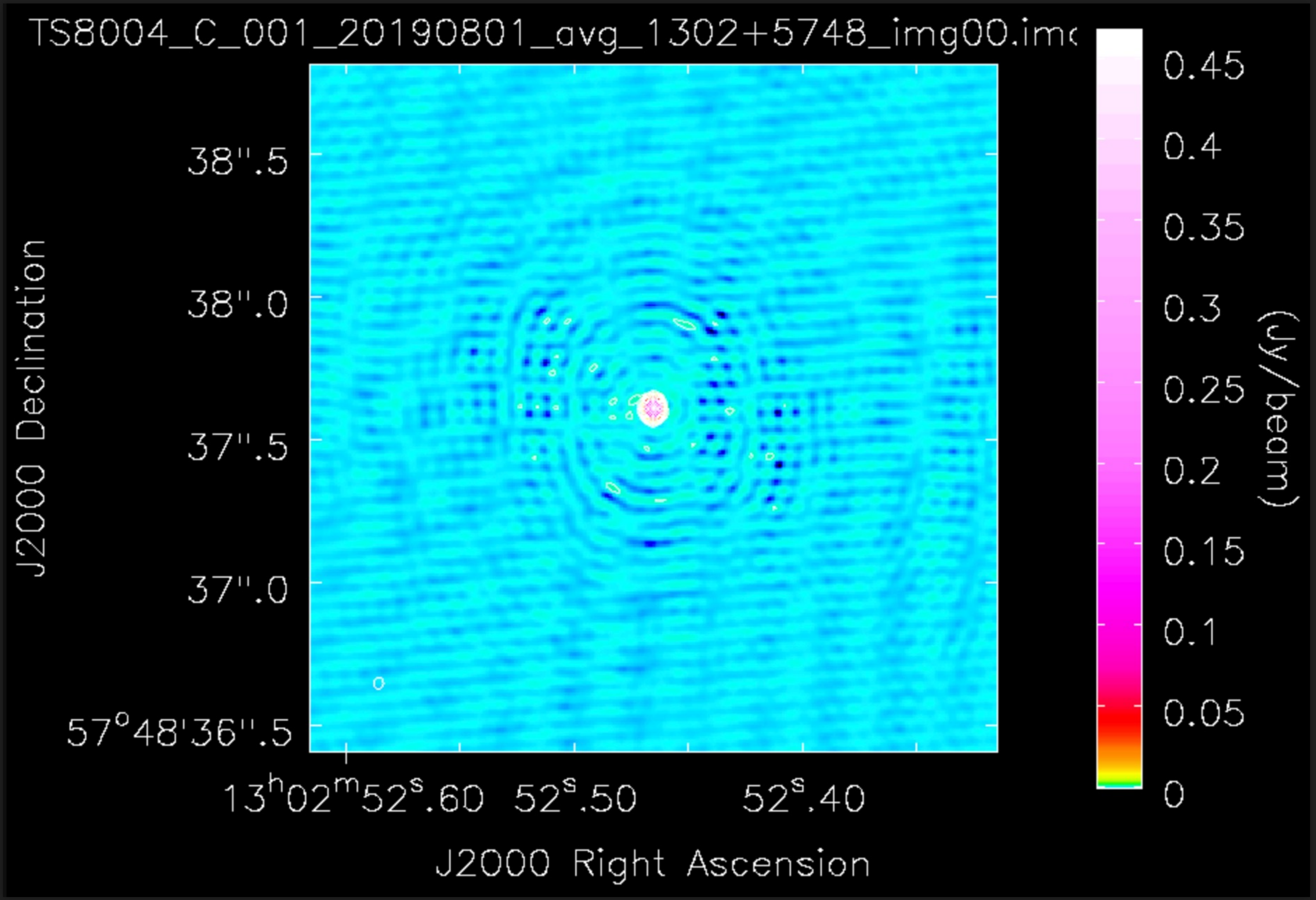
Inspecting the data – images first

Start with the phase calibrator image. Does it look like a point source?

If so, are there any calibration errors?

Are there any large scale ripples across the intensity or residuals map that could point to problems?

Pictured – intensity image of phase calibrator



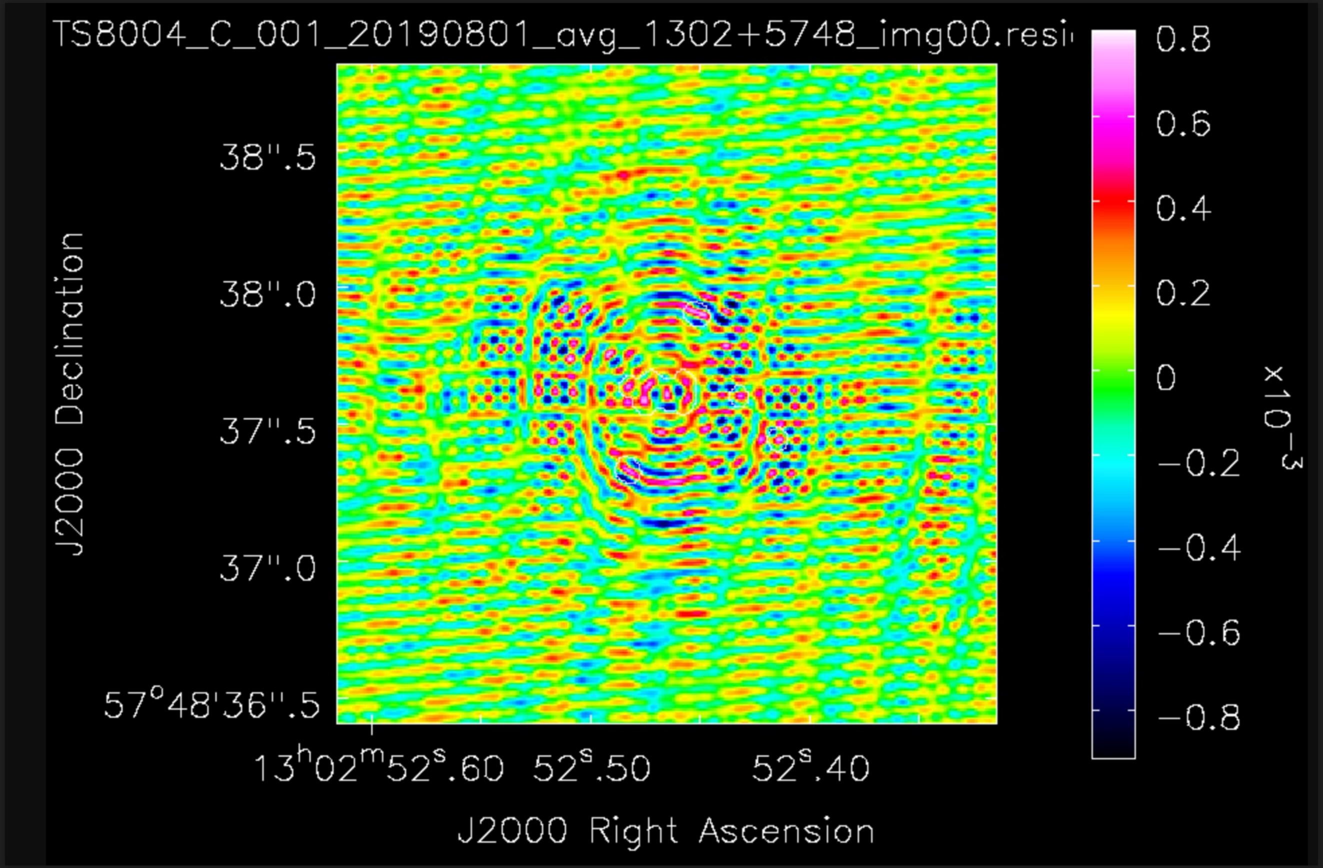
Inspecting the data – images first

Start with the phase calibrator image. Does it look like a point source?

If so, are there any calibration errors?

Are there any large scale ripples across the intensity or residuals map that could point to problems?

Pictured – residual image of phase calibrator



Inspecting the data – images first

- The target image is nice to see, especially if you know roughly what the structure of the source should be, but it doesn't tell you much about the calibration as any calibration errors will have been folded through previously
- Next up, go to the Plots tab in the weblog

Inspecting the data – the Plots tab of the weblog

Do your calibrators look like point sources?

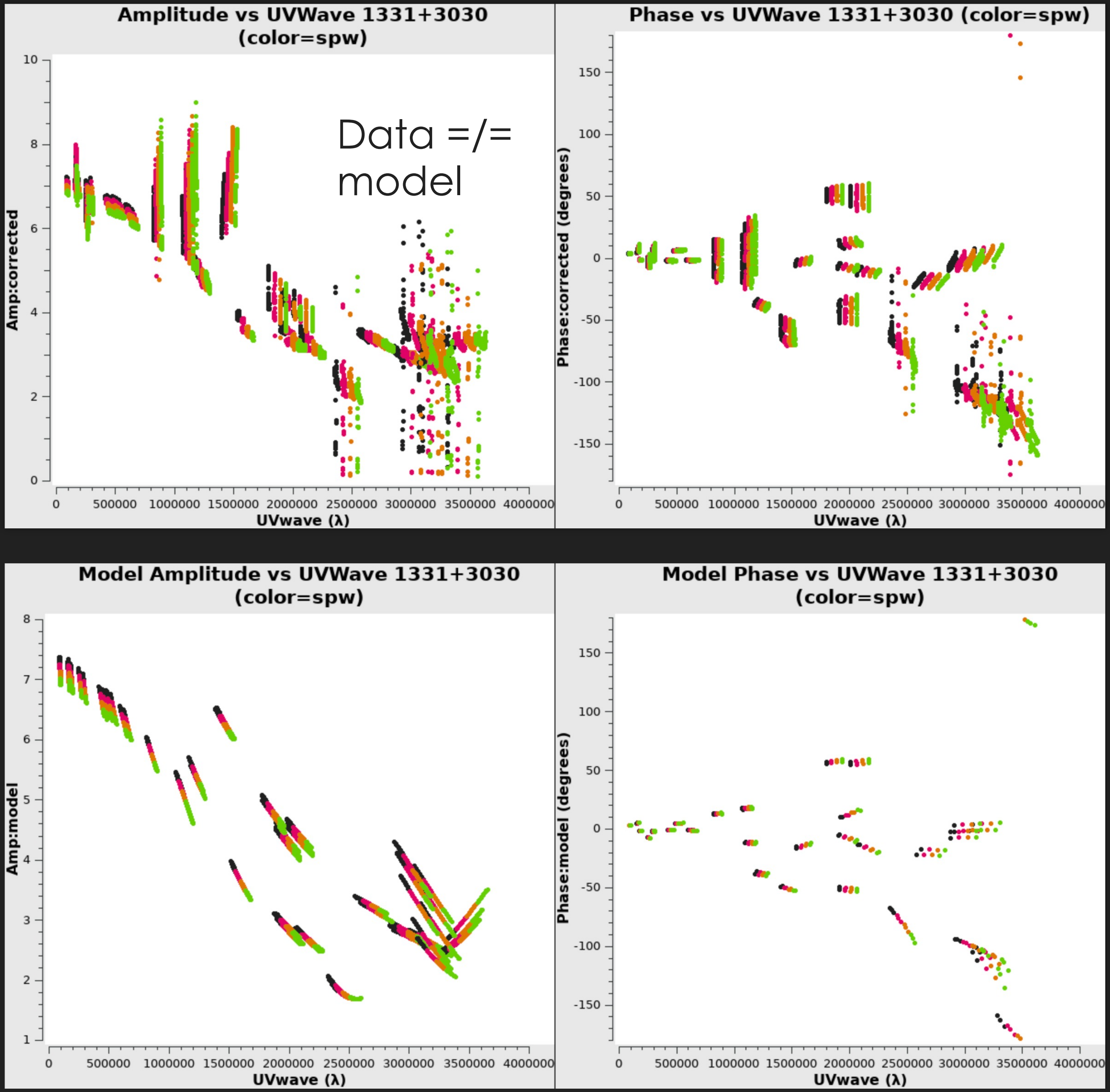
How noisy is the data?

Are there any amplitude dropouts?

Are there any phase discontinuities?

Do the calibrator models look like the calibrated data?

Pictured - 3c286 calibrated data and model



Inspecting the data – the Plots tab of the weblog

Do your calibrators look like point sources?

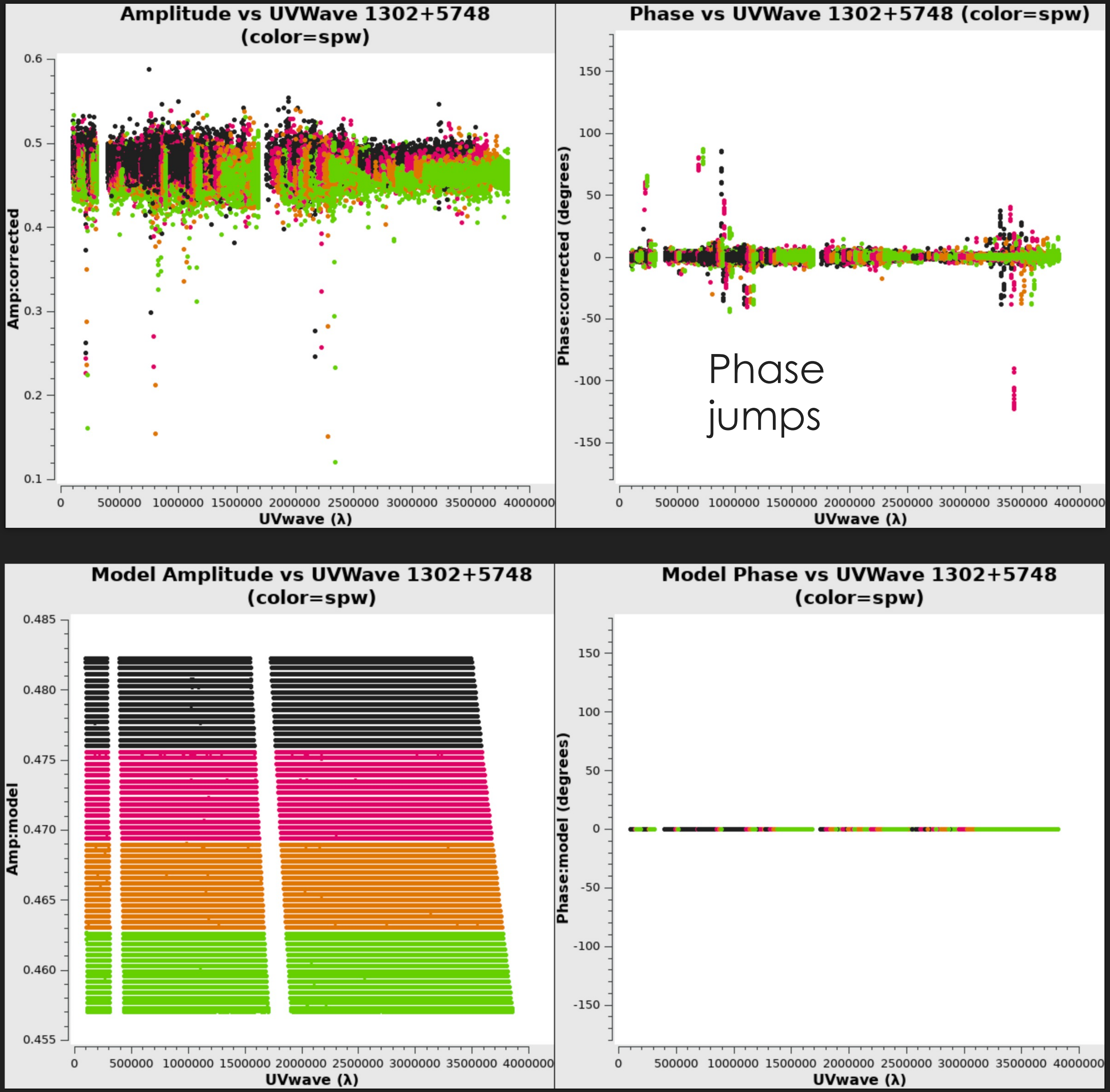
How noisy is the data?

Are there any amplitude dropouts?

Are there any phase discontinuities?

Do the calibrator models look like the calibrated data?

Pictured - phase calibrator calibrated data and model



Inspecting the data – the Plots tab of the weblog

Do your calibrators look like point sources?

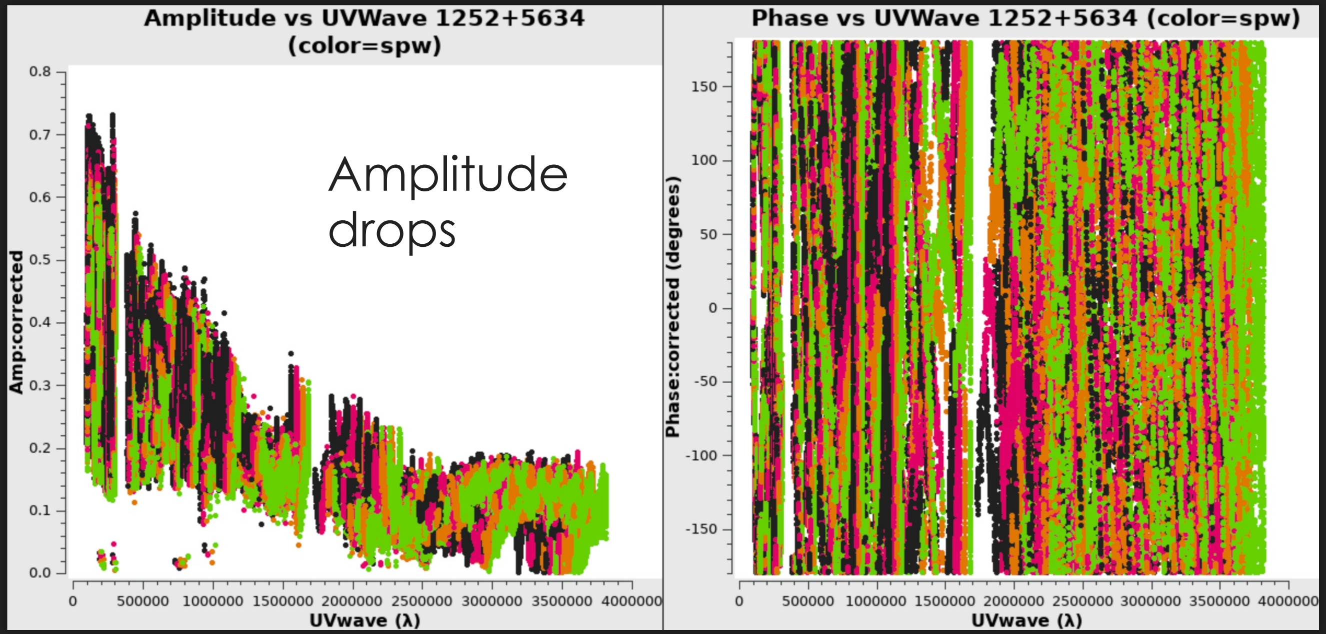
How noisy is the data?

Are there any amplitude dropouts?

Are there any phase discontinuities?

Do the calibrator models look like the calibrated data?

Pictured - target calibrated data



Inspecting the data – use the calibrated UV plots

Do your calibrators look like point sources?

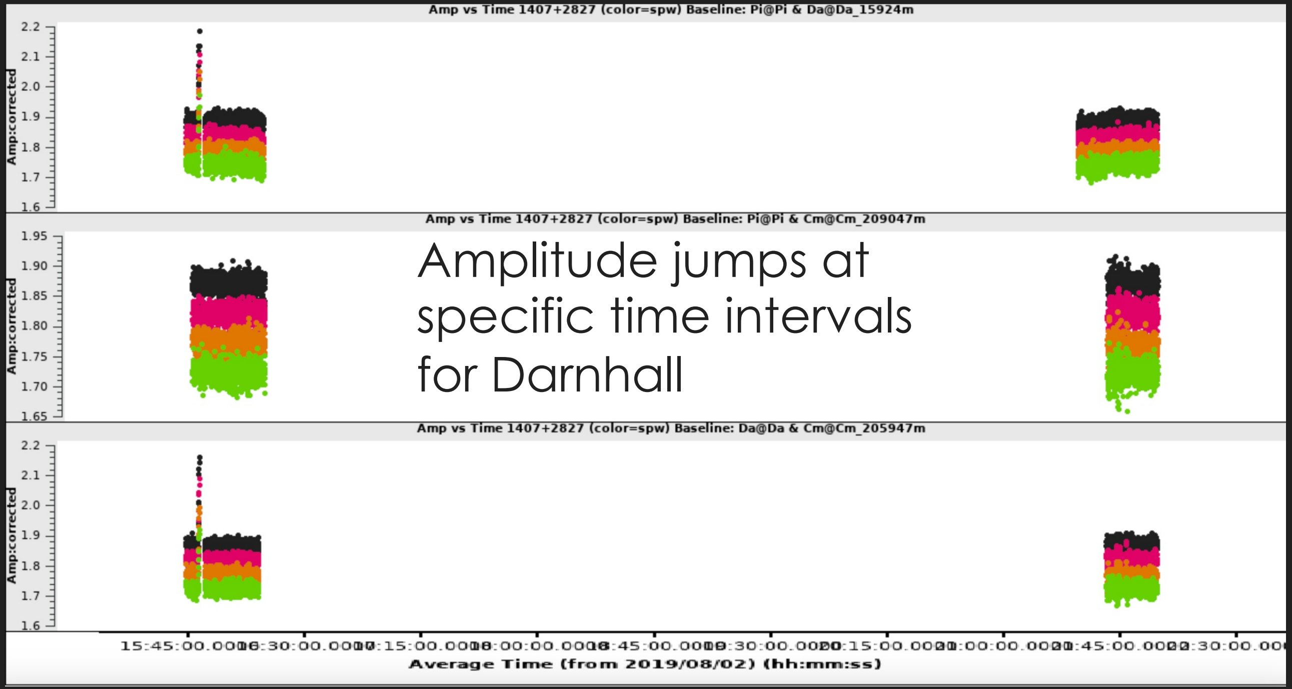
How noisy is the data?

Are there any amplitude dropouts?

Are there any phase discontinuities?

Do the calibrator models look like the calibrated data?

Pictured - band pass calibrated data, Amp vs time



Inspecting the data – the Plots tab of the weblog

- The plots tab is an under utilised part of the weblog
- It shows you all the issues with the data in a handful of easy-to-read plots
- The main question to ask yourself is: do my calibrators look like calibrators? Does my target look like a target?

- These may look correct and fine, but we should also check the flux scaling and band pass tables – look to the Calibration tab