e-MERLIN

MANCHESTER 1824
The University of Manchester

UKRI Science and Technology Facilities Council

Links to Radio Interferometry Introduction

This document does not intend to go over radio interferometry as there are many resources online which cover these aspects in more detail. For example, the "Essential Radio Astronomy" textbook by James Condon and Scott Ransom is available online and gives a good overview of interferometry at a level for Masters/PhD students. You can also look at the lectures and tutorials for the European Radio Interferometry School (ERIS) which can also be found online. It is recommended that you go through some of these parts if you are not sure of some of the fundamentals of radio astronomy.

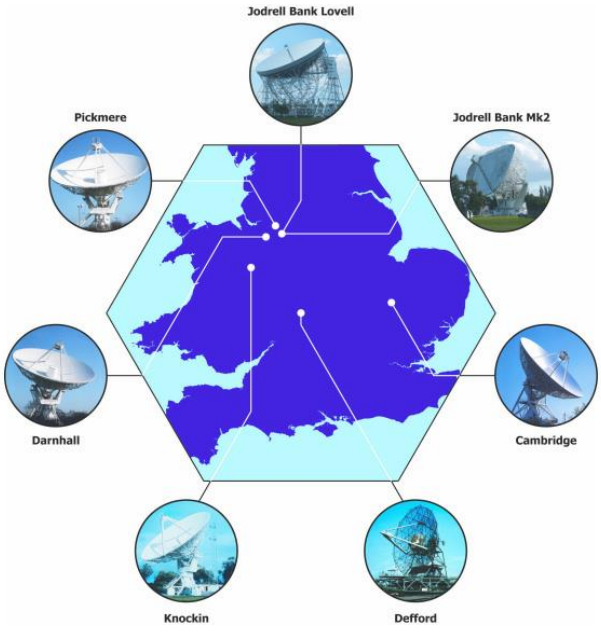
The e-MERLIN interferometer

e-MERLIN is a UK-based interferometer consisting of 7 25m+ antennas (see old image below of where they are situated in the UK). They are (in measurement set order):

- Lovell (Lv) : on site at Jodrell Bank
- Mark II (Mk2) : on site at Jodrell Bank
- Knockin (Kn) : near Oswestry on the Wales border
- Defford (De) : south of Birmingham
- Pickmere (Pi) : near to Jodrell Bank
- Darnhall (Da) : near to Jodrell Bank
- Cambridge (Cm) : should be self-explanatory!

e-MERLIN operates at 3 frequencies, L band (1.25-1.75 GHz), C band (4.3-7.5 GHz) and K band (19-25 GHz). The longest baseline length is ~210 km, so e-MERLIN is capable of reaching resolutions of 150 (L band), 40 (C band) or 10 (K band) milli arcseconds. A full description of e-MERLIN observing capabilities can be found online.

e-MERLIN observations are made year-round, with two calls for proposals in a year (one in ~November and one in ~May) with proposals graded on a science merit by an independent time allocation group (TAG). Observations are made dynamically throughout the cycle. The data are stored in a long-term archive at Jodrell Bank Observatory (JBO) and are calibrated and assessed for quality by internal operations team members before being sent out to the primary investigator. The data are calibrated with the e-MERLIN CASA Pipeline (eMCP) which can be downloaded from github and cited via the ASCL. This document goes through the e-MERLIN test data on 3C277.1 as a tutorial. The data can be downloaded online from it's distribution area.



The e-MERLIN CASA Pipeline (eMCP)

The e-MERLIN CASA Pipeline (eMCP) is the observatory-developed set of scripts to perform end-to-end calibration of e-MERLIN data, including flagging, calibration and first-pass imaging. It uses CASA: a python-based software that that is used by several other facilities to calibrate and image radio data, including the VLA (Very Large Array) and ALMA (Atacama Large Millimeter Array). Currently, the eMCP has been tested with CASA version 5.6 and version 5.8. A CASA 6 version is being worked on and is available to be downloaded at the eMCP github page. This document goes over the specific guidance for running the eMCP and improving your calibration to make the most of your e-MERLIN data sets.

Basics of the eMCP

The eMCP is designed to perform all the standard calibration steps for continuum data, so that you can get straight on with imaging the data or perform post-processing analysis such as spectral line or polarisation analysis. The purpose is to enable the user to make minor adjustments to improve the overall calibration and data reduction, either by improving the flagging of bad data via providing manual flag masks, or, but tweaking default parameters to improve calibration solutions. In theory, once you have run the pipeline several times making these changes, you should have a final calibrated dataset that can be used for imaging procedures. The eMCP is not designed to provide science ready images, and those provided in the pipeline are useful first pass images.

The eMCP is split into two sections: the “pre_processing” and “calibration” sections. Data from e-MERLIN will be passed through both of these sections, a quality assessment will be performed by the e-MERLIN operations team before sending the data out to PIs (Principal Investigators). The "pre_processing" part of the pipeline will import the raw fitsidi files into CASA measurement set (MS) format, perform some a priori flagging, average the data to a more manageable size and then save all the flagging information to enable the calibration part of the pipeline to run. The “pre_processing” part of the pipeline takes longer than the “calibration” section, and has more computing requirements, so the data sent to PIs is the averaged data and you will not be able to re-run the “pre_processing” part of the pipeline. The “calibration” section performs all the standard radio astronomy techniques to produce calibrated data, such as phase and amplitude calibration, flux scaling, band pass corrections, additional flagging, and preliminary imaging. These are the important sections that can require additional flagging or tweaking of pipeline parameters. Most of this document will go over the “calibration” section, and how to re-run the pipeline to get the most out of your data.

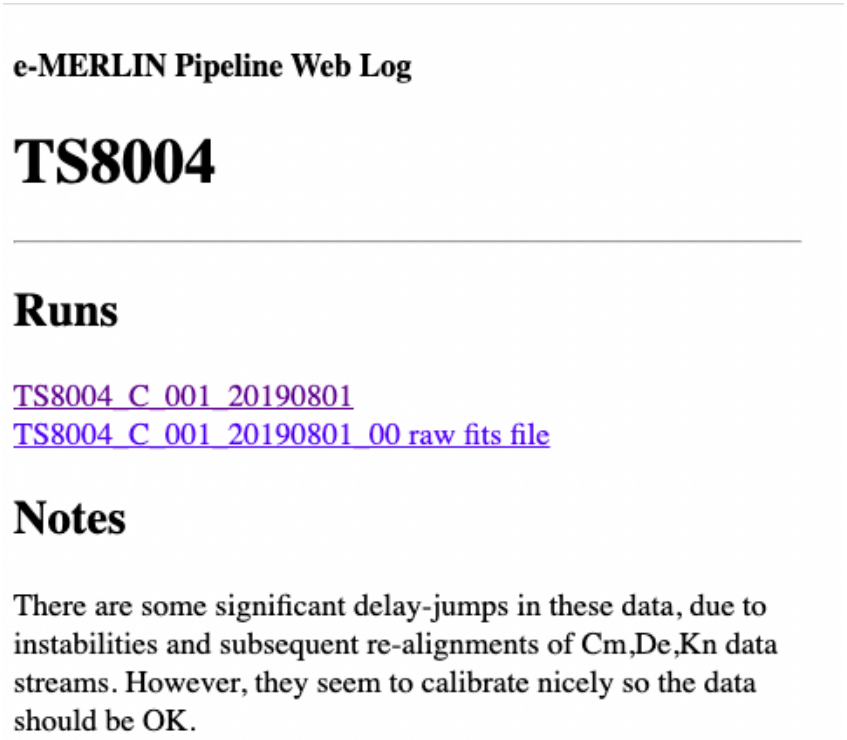
For every observation, the eMCP will also produce a weblog, which includes all the information about the observation in an easy-to-read format. The weblogs can be viewed on any internet browser and include a link to the calibrated data which you can download and produce science-quality images or do post processing on. The next section will go into detail about the weblog, and what each page of the weblog shows.

e-MERLIN data

Distribution page

Once your observations have been made, an *e*-MERLIN support scientist will contact you with information on the data quality as well as a link to a distribution area where you can view the weblogs for each individual observation and download the data. If you have your own dataset, then go to the distribution area link, else use the 3C277.1 test data at this link. The 3C277.1 data has been used to test the *e*-MERLIN pipeline and will be used for demonstration purposes for this tutorial.

You will see a link to "Runs" and some "Notes" on the project if they have been included by your support scientist. Further down the page you will find standard information on the pipeline and basic syntax on how to re-run the pipeline. The front page looks like



e-MERLIN Pipeline Web Log

TS8004

Runs

TS8004_C_001_20190801
TS8004_C_001_20190801_00 raw fits file

Notes

There are some significant delay-jumps in these data, due to instabilities and subsequent re-alignments of Cm,De,Kn data streams. However, they seem to calibrate nicely so the data should be OK.

Clicking on the top link (TS8004_C_001_20190801) will take you to the weblog for that observation. The second link (TS8004_C_001_20190801_00_raw_fits_file) downloads the raw unaveraged and uncalibrated *e*-MERLIN data for this project. This second link can be ignored for now.

The Notes section is optional and depends on the support scientist and programme as to whether it is included in the distribution area. As the 3C277.1 data is freely available as a test dataset, we include the Notes section in the distribution area. Most datasets will not have this "Notes" section. Instead, these notes will be sent to you in the email you receive when the data have been observed and passed quality assessment.

To continue, click on the top link to take you to the weblog.

The eMCP weblog

When clicking on the link to an observation weblog, a new tab will open on the "Home" page (see next section) of that observation. The links at the top of the weblog are listed in the sections below and will provide you with different information on the data quality. These are: Home, Observation summary, Pipeline Info, Calibration,

Visibility Plots, Flag statistics, Images and Download Data. The idea of the weblog is to enable quick-look inspection of the data for quality assessment.

Home

The “Home” page (screen shot shown below for the 3C277.1 data) gives an overview of the important pieces of information relevant to that observation.

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
Total bandwidth	512 MHz
Polarizations	R, L

Whilst most of the variables on the Home page are self-explanatory, an exhaustive guide is given below:

Project: In this case TS8004, but this will be your project’s unique identifier, e.g. CY12 prefix will be a cycle 12 project, and the three numbers at the end represent your project ID in CY12.
Run: In this case, TS8004_C_001_20190801. In most cases this will be in the same format: <*ProjectID*>_<*Band*>_<*Run number*>_<*YearMonthDay*>. This should separate all your runs in a straightforward and simple way.
MS File: The measurement set file name to be used when viewing this weblog, if you want to look at the different plots interactively. This is usually the Run name with the suffix “_avg.ms” in the top-level folder when you download the data.
Start: Start time and date (inclusive of all calibrators)
End: End time and date (inclusive of all calibrators)
Band: Observing band, either L, C, or K.
Antennas: This lists the antennas present in the *correlator configuration* and not necessarily the ones present in the measurement set. For example, a normal run will include all stations, bar the Lovell, so this line would normally have

all of them in here. However, if one of them wasn't working or stopped working, then this will be reflected in the calibration plots but will still list the antenna here.
Number of sources: Total number of sources in the measurement set, usually 5: Target, phase calibrator, 3C286 (1331+3030, flux cal), OQ208 (1407+2827, band pass) and 3C84 (0319+4130, pol cal).
Integration time: This refers to the integration time of the *MS file* listed above. All e-MERLIN data are recorded at 1s integration time, but averaged to 4s in the pre_processing stage of the pipeline.
Frequency: The frequency range of this observation, in this case from 4.82-5.33 GHz.
Num. spw: Number of spectral windows (spws). In the case of L band, it is usually 8 spws, and for C band and K band, it is usually 4 spws.
Channels/spw*: The number of channels per spw, given as the number in the *MS file* listed above. The e-MERLIN data is correlated with 512 channels per spw, but only the 128 channels per spw data are sent to the PI (Principal Investigator) to save space.
Channel width: Usually 1 MHz at C/K band or 0.5 MHz at L band, due to the number of channels left after averaging, I.e., 128 per spw, and depending on the number of spws in an observation.
spw bandwidth: The bandwidth in a single spw, I.e., the Total bandwidth divided by the number of spws.
Total bandwidth: The full bandwidth of *e*-MERLIN, which will always be 512 MHz, unless you have a non-standard project and this value is less than 512 MHz.
Polarizations: Usually listed as "R, L", by default we observe with both right and left polarisations and the data includes both parallel (RR, LL) and cross (RL, LR) hand data. However, the pipeline only operates on the parallel hand data.

Observation summary

The observation summary gives specific information on the sources in the observation, uv plots of those sources and elevation plots of the observation.

The first part gives a summary of the sources, including the listobs file made at two points in the pipeline. The "current observation" listobs txt file shows the "_avg.ms" listobs file from CASA, whereas the other listobs files will be from the initial unaveraged measurement set, and/or spectral line zoom mode listobs files, if available in the data. A table of sources is shown, with a separation between the phase calibrator and target – we aim to schedule unresolved phase calibrators that are within 5 degrees of the target field. Listed are the calibrator sources for this project: 1331+3030 (flux calibrator), 1407+2827 (band pass cal), 0319+4130 (point/pol cal) and the phase calibrator, 1302+5748.

Next, the list of antennas present in the observation is included. They are initially sorted in their radial order from the nominal array centre, I.e. home stations such as the Lovell (Lv), Mark 2 (Mk2), Pickmere (Pi), Darnall (Da) are at the top and outstations like Knockin (Kn), Defford (De) and Cambridge (Cm) at the bottom of the list. The reference antenna(s) is chosen during the pre_processing section of the pipeline after averaging has been performed, by by looking at the signal-to-noise ratio (SNR) on the reference fields and then choosing the antenna with the highest SNR. Due to the slightly larger diameter of the Cambridge telescope compared to the others, it is often automatically picked by the pipeline, but it is usually better to choose an antenna closer to the centre of the array, such as Mk2, Pi

Summary:

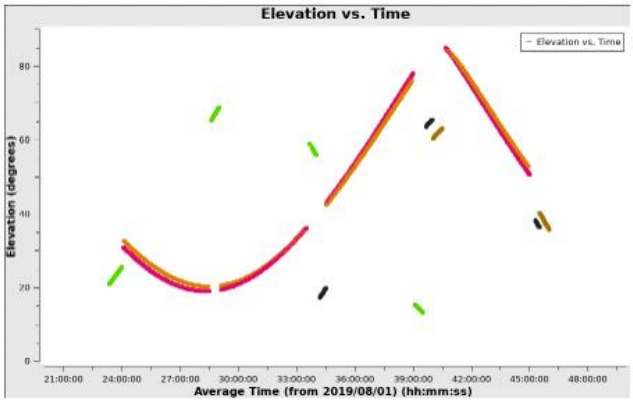
Summary of current observation (listobs): txt

Antennas:

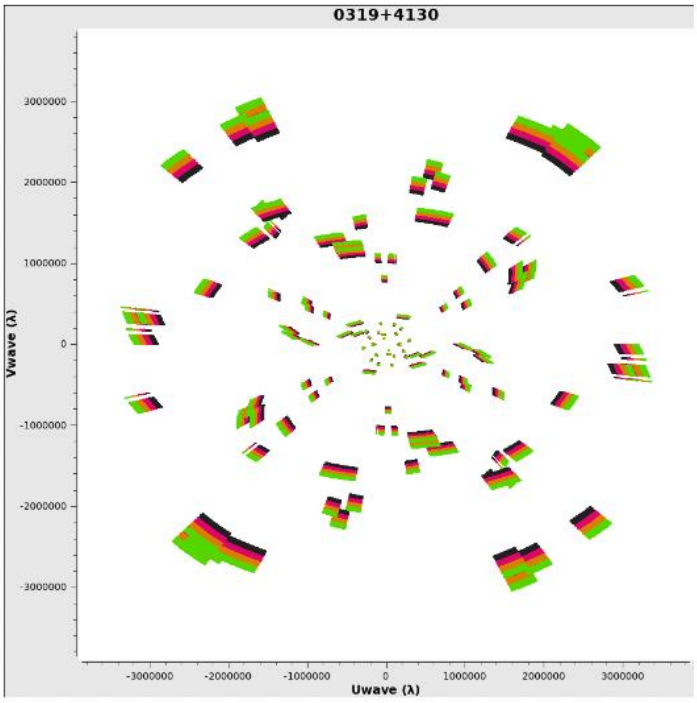
```
Mk2
Pi
Da
Kn
De
Cm
```

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

Source elevation:



UV coverage:



or Da. Note also that Mk2 will not be chosen if the Lovell is in the array for the observation.

A plot of the source elevation over time for the duration of the observation is included, colourised by the source. While a colour key is not shown, *e*-MERLIN requires long tracks on target/phase calibrator pairs to build up sufficient uv coverage. In the plot below this can be seen clearly as two sources are observed in long tracks, with occasional excursions to the bright calibrator sources.

Finally, on this page are uv coverage plots of all the sources in the measurement set. The uv plots are made at the start of the pipeline procedures so they do not reflect any flagging performed later in the pipeline. Below is the uv coverage plot for 3C84 (0319+4130) in the 3C277.1 dataset.

Pipeline info

The pipeline info page on the weblog is a table with all the pipeline steps executed and whether they have been executed to completion (in green) or not (in red). It also has notes on the execution, including the time taken, and the key parameters used at each stage.

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, scanreindexgap_s=15.0. Hanning=False, createmms=False, timebin=4s
flag_aoflagger		0	-	
flag_apriori	OK	2022-07-21 10:59:30	11 min	Clip zeros. Subband edges *:0~3;508~511. channels 0:0~26 3:485~511. Observatory 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, datacolumn=data

At the bottom of the page are links to the log files and parameter files. The eMCP.log file is a shortened version of of the CASA_eMCP.log file, which shows what comes out of the CASA logger when the pipeline is run. It is often useful to use both files to work out what sections of the pipeline may need changing to improve the pipeline solutions, or, if the pipeline has failed, specifically at which stage and which task it has failed on.

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

The relevant parameter files include the eMCP_info.txt log, which shows all the parameters used for each step of the pipeline, in case it's ambiguous or if you want to restore a calibration to the observatory pipelined dataset. The caltables.txt file shows which parameters each calibration table used when it was created.

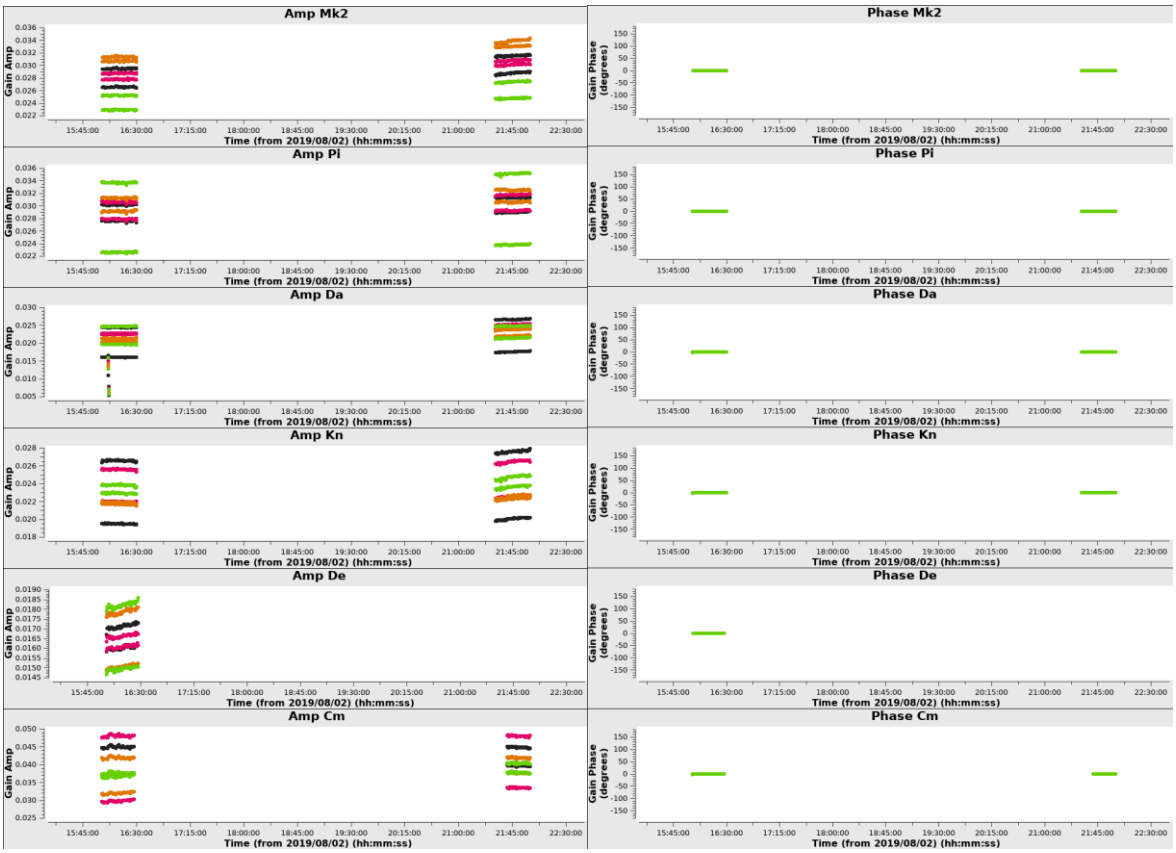
Calibration

The Calibration tab is arguably the most important tab in the eMCP. A full discussion of every step undertaken as part of the pipeline is left to a later section of this document. Here, we show the main types of plots and how they are set up on the page for interpretation.

Delay/Phase/Amplitude plots

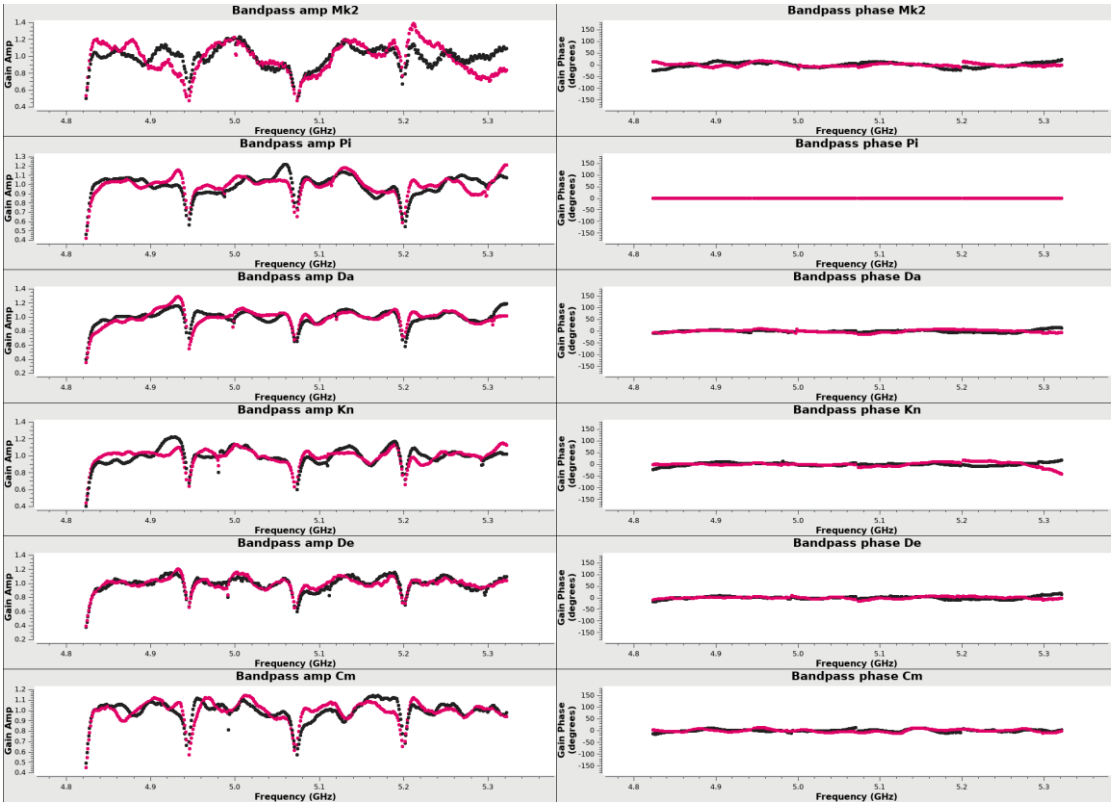
These plots are shown as the delay/phase/amplitude plotted against time. This includes many tables in the

calibration tab, but broadly, these show the variation of the solved value as it evolves over time. Any discontinuities should be flagged. When an “ap” solution has been performed, the phase plot will be on the left and the amplitude plot on the right. If two colours are shown (e.g., delay plots), then you are looking at the two parallel polarisations, LL and RR. If more than two colours are shown, then the plots are colourised by spectral window. The only difference to this colours rule is for the allcal_d.K1 plot, which colourised by field. Below is an example of an “ap” solution plot.



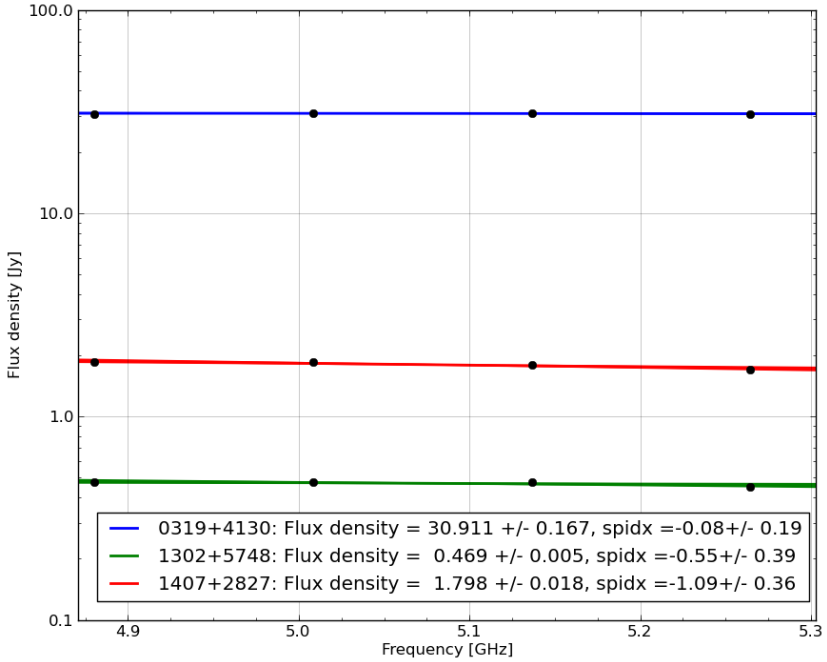
Bandpass plots

The plots showing the bandpass solution describe the gain amplitude (or phase) over the frequency band and are colourised by the correlation, I.e. LL or RR. Edges of the bands should be flagged, and the edges of the spectral windows will be visible but *not* flagged. This is ok, as they are not included when computing the bandpass. The bandpass plots should look similar to the one below.



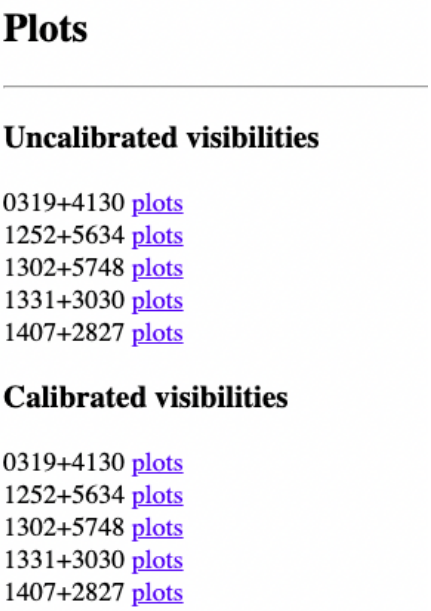
Fluxscale plot

To do the flux-scaling properly, we use 3c286 as a flux calibrator with a known e-MERLIN model of the source, to estimate the fluxes on the other sources. The plot created is below, showing the flux density as a function of frequency, with one data point per spectral window. The lines represent the fit and errors on that fit. It should be noted also that the read out of fluxes is included just below the plot too.



Plots

The plots tab is a useful though often overlooked part of the pipeline. It provides visibility plots of the data for all sources, to enable you to inspect the data quality easily. It is in two main sections: Uncalibrated/Calibrated visibility plots, and Calibrated UVplots. The former will show you each source amplitude and phase, plotted against time or frequency. The latter will show you the amplitude/phase plotted against uvdistance, and comparing to the model version obtained from the pipeline flux scaling procedure or 3c286 model, in the case of 3c286.



Visibility Plots

The visibility plots are split into uncalibrated and calibrated sections. You can click on a link per source, an example for calibrated 1407+2827 data is shown below. It is often useful to compare the uncalibrated visibilities with the calibrated visibilities to see what the pipeline has flagged during the pipeline processing steps. You will usually find that areas in the data where there is no signal or poor phase will be flagged and therefore not be present in the calibrated visibility plots. However, if you notice data in the uncalibrated visibility plots that look ok and significant amounts of data are flagged by the pipeline, then it is worth looking at where the pipeline is flagging the data, either in the log files or by tweaking the default parameters and re-running the pipeline.

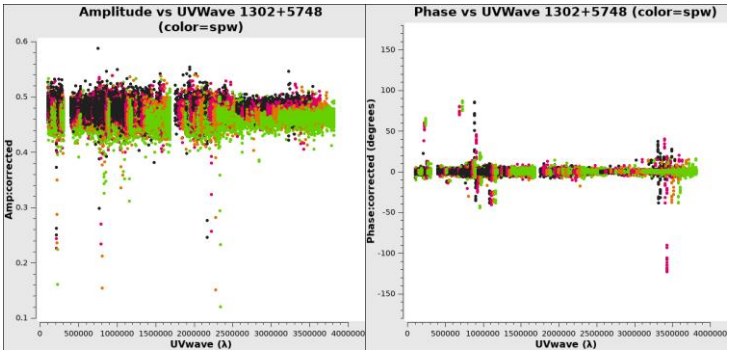
Calibrated uv Plots

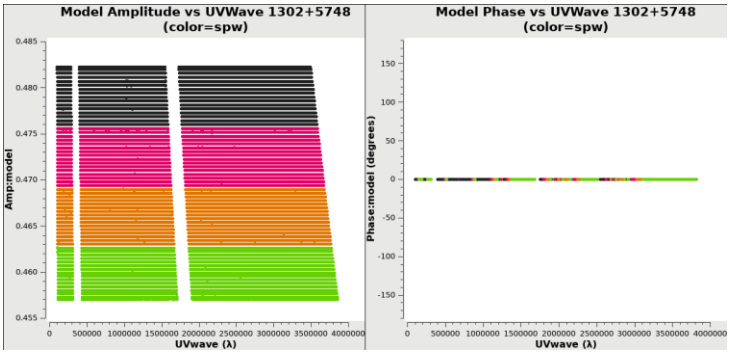
The calibrated uv plots show the amplitude and phase of a source plotted against the uv distance. For calibrators,

the model is also plotted next to it, so you can compare what the pipeline thinks the source should look like in uv space, and what the data are showing you. This can be quite useful as a diagnostic tool, as often there will be small excursions in amplitude or phase which show small errors in the calibration.

Phase calibrator

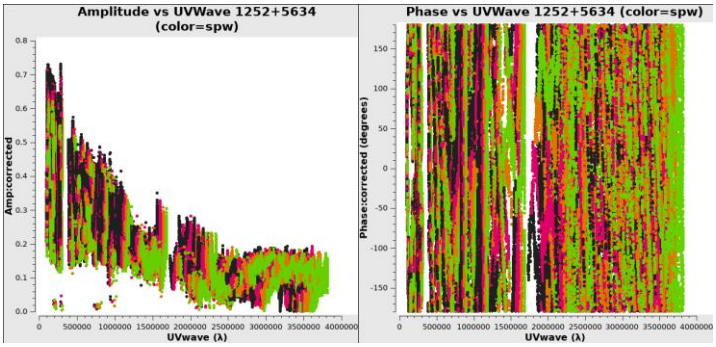
The phase calibrator below shows the model is a point source (flat phases and amplitude is constant at all uv distances for a given frequency), but there are some noticeable differences when compared to the model, which has several phase excursions from a flat zero phase.





Target

Here, you can see small dropouts in the amplitude, but you cannot see anything in the phase. This is because the source has a complicated morphology, so the phases wrap a lot, but the amplitude gradually decreases as the source becomes more resolved on longer baselines.



Flag statistics

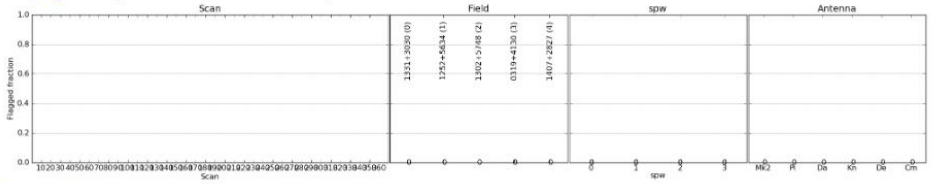
The flag statistics page shows how much flagging has been performed at certain points in the pipeline and is split

into four parts: flags by scan, flags by field, flags by spw and flags by antenna. Often, if there is a lot of data being flagged in your dataset, it is useful to look here to find out where it is happening, both in terms of where in the pipeline it may be happening, but also what part of the data, I.e., antenna or spw it is happening on.

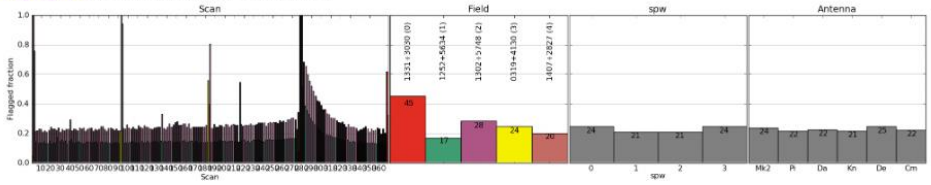
Flag statistics

High-resolution plots in step title.

run_importfits (Total: 0.0%. Increase: 0.0%)



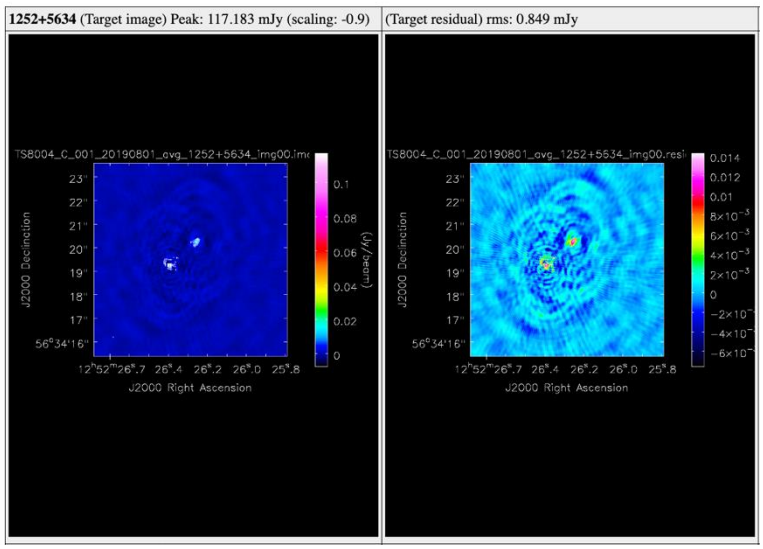
flag_apriori (Total: 22.6%. Increase: 22.6%)



Images

The images tab shows both the target and phase calibrator field preliminary images made at the end of the pipeline procedures. Below is the target field, showing the image (left) and residual map (right). A second set of images to the right-hand side of the page shows zoom images of these maps. The peak of the map and rms of the residual map are also shown so you can get an idea of the signal-to-noise ratio in the image.

1252+5634 (up)



Download Data

This tab allows you to download the weblogs and calibrated avg.ms file data so that you can look through it yourself and re-reduce it if necessary.

Download data

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

Data reduction and calibration procedures in eMCP

The eMCP is an end-to-end pipeline that will ingest your dataset, produce automated flags, perform standard calibration tasks, and make preliminary images of the data, whilst making useful plots that can be used for inspection of the data quality. The philosophy of the pipeline is therefore to provide a calibrated dataset that the user can re-calibrate at their leisure, making minor changes to the default parameters and adding their own manual flag files, to improve the calibration. The eMCP runs in two stages and not all stages are made available to the PI, unless requested explicitly. The two sections are the "pre_processing" and "calibration" stages, the former of which is run at JBO but cannot be re-run by the user, and the latter is editable by the user. Therefore, the user should only ever need to touch a handful of files in the pipeline, and only run the calibration steps.

This section goes over each of the steps outlined in the "Pipeline info" page of the weblog in detail going over what each step does, what outputs are provided in each step, what plots are produced and what the pipeline default parameters are with some hints as to modifications to help with different data sets. The pipeline files are outlined below, I.e. what you get when you download your data from the weblogs.

Pipeline files

When you download the data from the TS8004 page (or your own data), you should be presented with several files and folders. They are:

- **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 the eMCP materials and scripts

To load up the weblogs locally, you can load the index.html file in the weblog folder using a browser, e.g., firefox.

inputs.ini file

An example of the inputs.ini file is given below, I.e., the one for this tutorial dataset.

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

fits_path = /scratch/raw_data/TS8004/TS8004_C_001_20190801/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]

# 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
# More details in https://github.com/e-merlin/eMERLIN_CASA_pipeline
```

The main parts of this file are the fits-path, which should point to where your raw uncalibrated data live. In the case of the eMCP, you can usually leave this blank as you will only be running from the calibration part of the pipeline. The “inbase” parameter is the run name given in the weblog. When the pipeline starts it will look for measurement sets with the “inbase” prefix, so it’s important to make sure this is set correctly without typos. The rest of the inputs.ini file includes the calibrator or target information. You can add multiple sources to the target list as follows:

```
targets   = target1,target2,target3
phscals   = phasecal1,phasecal2,phasecal3
```

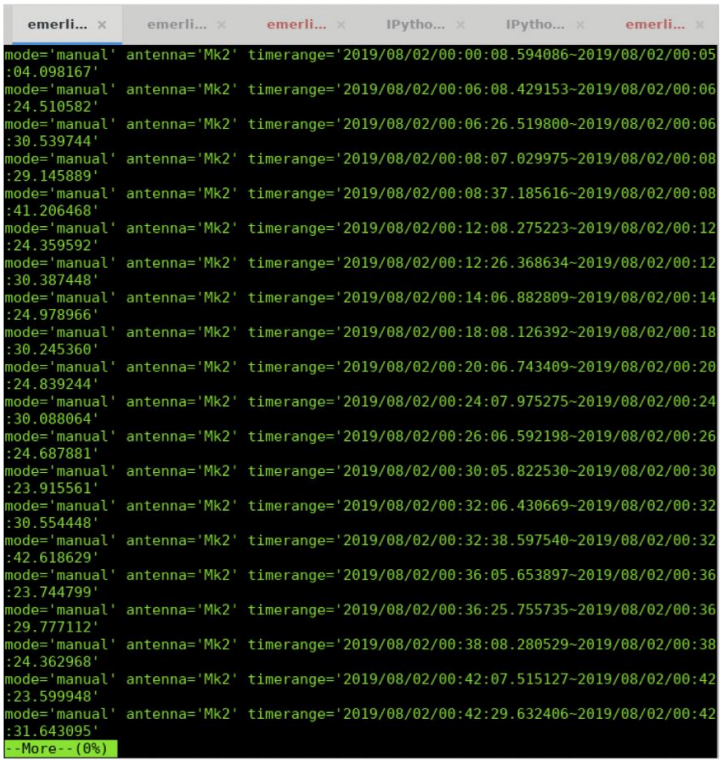
If target 1, 2 and 3 share the same phase calibrator, then set them to the same phase cal. ``

default_params.json file

This is a long file but includes all the parameters that are used as part of the pipeline to calibrate the data. We will go through this in detail later in this document. It is a json file, so you can open it with a text editor and make changes to it, as necessary.

observatory.flags file

This is a list of automatically generated flags from the e-MERLIN correlator that notes when the antennas are on source or not. It gets read into the eMCP near the start to apply these flag commands to the data and make it easier for PIs to have a well-calibrated dataset. It is recommended not to edit this file. An example of the observatory flags file is on the right and is useful for seeing how all flag files are written.



The eMCP log files

The files eMCP.log and casa_eMCP.log show the pipeline and CASA logs respectively, from the runs of the pipeline made during the quality assessment process. Everytime you re-run the pipeline these will be appended to, so you can look back and see what was run previously.

The manual_avg.flags file

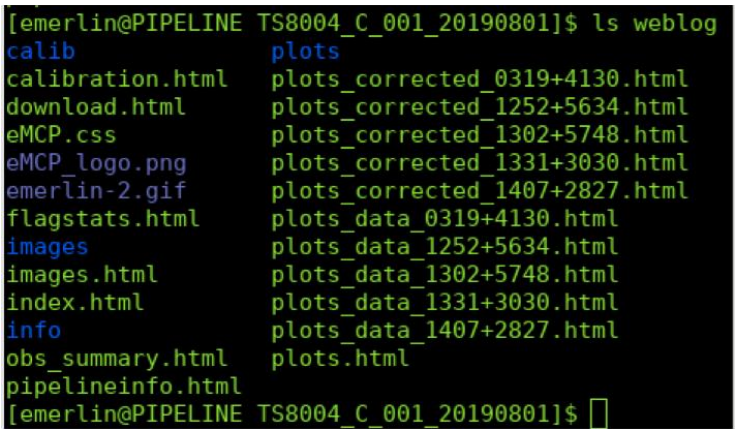
This file will include all the manually added flags from your support scientist. If this file does not exist, then you can generate it yourself and add your own sets of flag commands. We will do this later for the tutorial dataset.

The _avg.ms and _avg.ms.flagversions folders

These folders are the calibrated and averaged data in measurement set format, and the associated flagversions tables generated throughout the pipeline, respectively. Unless requested, you won't have access to uncalibrated raw fits files or unaveraged measurement set data, which precludes you from running the "pre_processing" steps of the pipeline (see later discussions).

Other folders

All other folders refer to the calibration tables, weblog, logs and plots needed by the pipeline or weblogs to display/calibrate the data. You can plot or display these images via CASA or by image viewers, which will help you decide on good/bad data. The image on the right shows all the information included in the weblog folder for the tutorial dataset. All of this is generated during the pipeline runs.



(Re-)Running the eMCP

To run the pipeline, you need to be in the data directory for the dataset you want to run the pipeline on. Open a terminal window in this directory. To rerun the pipeline, type the following command:

```
/path/to/casa -c ../../eMERLIN_CASA_Pipeline/eMERLIN_CASA_pipeline.py -r
calibration
```

This will re-run all the calibration steps of the pipeline. In case you want to run a specific step, instead of 'calibration' in the previous command, replace this with the relevant step, e.g., 'first_images':

```
/path/to/casa -c ../../eMERLIN_CASA_Pipeline/eMERLIN_CASA_pipeline.py -r
first_images
```

You can also run the pipeline for some calibration steps and skip others. For example, if you want to run all steps in the calibration phase, except first_images, you can write the following;

```
/path/to/casa -c ../../eMERLIN_CASA_Pipeline/eMERLIN_CASA_pipeline.py -r
calibration -s first_images
```

Please note that you should not need to run the first half of the pipeline, e.g., the bit named "pre_processing". That is because all these steps are performed at Jodrell Bank first with the initial quality assessment. You will only need to run the “pre_processing” stage of the pipeline if you have raw fits files which you can download from the TS8004 test dataset area. However, this takes a long time and will take up significantly more space in your directory. Nevertheless, this document reviews these steps first, and will then go into the calibration steps afterwards.

pre_processing

This section will describe the overview of the first half of the pipeline called the "pre_processing" stage. As the pro_processing stages are performed at Jodrell Bank, there is no need to re-run these stages, but below is an overview of what each stage does, for completeness.

The stages are as follows:

1. run_importfits
2. flag_aoflagger
3. flag_apriori
4. flag_manual
5. average
6. plot_data
7. save_flags

start_pipeline

This step is run by default every time you run the pipeline but note that it is not listed in the steps to be performed above. Therefore, you cannot “run” this task as you would all the others. This is because this step only performs the most basic of tasks to print back at you what the pipeline is going to do and reads the global parameters from the default_params.json file.

What does it produce?

When this step is run, I.e., at the start of *any* step run in the pipeline, then you will get a terminal print out of what the pipeline will do. For example, below is the printout of the start of a pipeline run if only the “run_importfits” step is run. One thing to note is that if you are running this step from the start, I.e., you only have a fits file and no weblogs, then this step does not generate the weblogs, that is done in the next steps.

```
2023-10-06 09:42:56 | INFO | Starting pipeline
2023-10-06 09:42:56 | INFO | Running pipeline from:
2023-10-06 09:42:56 | INFO |
/pipeline1/processing/TS8004/TS8004_C_001_20190801/eMERLIN_CASA_pipeline/
2023-10-06 09:42:56 | INFO | CASA version: 5.8.0
2023-10-06 09:42:56 | INFO | Pipeline version: v1.1.19
2023-10-06 09:42:56 | INFO | Using github branch: master
2023-10-06 09:42:56 | INFO | github last commit: f2b6efa
2023-10-06 09:42:56 | INFO | This log uses UTC times
2023-10-06 09:42:56 | INFO | Loading default parameters from
./default_params.json:
2023-10-06 09:42:56 | INFO | fits_path :
/scratch/raw_data/TS8004/TS8004_C_001_20190801/DATA/
2023-10-06 09:42:56 | INFO | inbase    : TS8004_C_001_20190801
2023-10-06 09:42:56 | INFO | targets   : 1252+5634
2023-10-06 09:42:56 | INFO | phscals   : 1302+5748
2023-10-06 09:42:56 | INFO | fluxcal   : 1331+3030
2023-10-06 09:42:56 | INFO | bpcal     : 1407+2827
2023-10-06 09:42:56 | INFO | ptcal     : 0319+4130
2023-10-06 09:42:56 | INFO | Step selection
2023-10-06 09:42:56 | INFO | run_steps : ['run_importfits']
2023-10-06 09:42:56 | INFO | skip_steps: []
```

```
2023-10-06 09:42:56 | INFO | Sorted list of steps to execute:
2023-10-06 09:42:56 | INFO | run_importfits  : 1
2023-10-06 09:42:56 | INFO | flag_aoflagger  : 0
2023-10-06 09:42:56 | INFO | flag_apriori    : 0
2023-10-06 09:42:56 | INFO | flag_manual     : 0
2023-10-06 09:42:56 | INFO | average         : 0
2023-10-06 09:42:56 | INFO | plot_data       : 0
2023-10-06 09:42:56 | INFO | save_flags      : 0
2023-10-06 09:42:56 | INFO | restore_flags   : 0
2023-10-06 09:42:56 | INFO | flag_manual_avg : 0
2023-10-06 09:42:56 | INFO | init_models     : 0
2023-10-06 09:42:56 | INFO | bandpass        : 0
2023-10-06 09:42:56 | INFO | initial_gaincal : 0
2023-10-06 09:42:56 | INFO | fluxscale       : 0
2023-10-06 09:42:56 | INFO | bandpass_final  : 0
2023-10-06 09:42:56 | INFO | gaincal_final   : 0
2023-10-06 09:42:56 | INFO | applycal_all    : 0
2023-10-06 09:42:56 | INFO | flag_target     : 0
2023-10-06 09:42:56 | INFO | plot_corrected  : 0
2023-10-06 09:42:56 | INFO | first_images    : 0
2023-10-06 09:42:56 | INFO | split_fields    : 0
2023-10-06 09:42:56 | INFO | Updating casa-data
```

What are the default parameters?

The default parameters for this step are those at the top of default_params.json file, I.e., the global parameters. *It is particularly important that these are properly set for each pipeline run.* These parameters will be used throughout the calibration steps and can make a significant difference to the outcome of the pipeline.

Parameter	Default	Comments
update_casa-data	true	This is by default set to True as it assumes that you are running your pipeline on a CASA installation that you have kept up to date and have root access to. However, for machines where the CASA install is shared, then this should be set to false. In those cases, make sure that your CASA installation should be updated i.e., run "!update-data" in CASA.
refantmode	"strict"	This parameter will use the same reference antenna throughout the pipeline run. This can optionally be set to "flex" but should be left to "strict" if you want to do polarisation processing at the end of the pipeline. If set to flex, make sure you set the refant parameter below. The pipeline will generate a list of reference antennas in preference order which can be used here.
refant	""	By default, this is set to blank as the pipeline will choose the antenna with the best SNR in the calibrators from the measurement set. However, the pipeline often picks Cambridge as a reference antenna which is not the best choice so have a look at what the best reference antennas are in the pipeline and choose one of Pi/Da/Mk2/Kn. If you have set the refantmode to "flex" then you can set this to a list of the antennas to use in case one antenna fails.
is_mixed_mode	"auto"	You should leave this parameter to auto, as it will work out whether your data is continuum or spectral line mode and then

		perform the necessary additional pipeline steps if it is the latter.
applymode	"calflagstrict"	This parameter is set to "calflagstrict" by default as it will be conservative when looking for bad solutions in the data. However, "calflagstrict" will flag all spws if it finds a bad piece of data on one spw, which will often lead to substantial amounts of flagging of the e-MERLIN dataset. So, you can change this to "calflag" which will save a lot more of your data for calibration, which will hopefully mean a significant improvement in sensitivity. However, keep in mind this may allow more bad pieces of data into the dataset, so more manual flagging may be needed.
run_importfits flag_aoflagger flag_apriori flag_manual average plot_data save_flags 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	1	These are the steps of the pipeline, so the default is set to 1. You can leave this set to 1 and run everything from the command line if you go through line by line.

Troubleshooting at this stage

If you try to run this step on its own, I.e.

```
/path/to/casa -c ../../eMERLIN_CASA_Pipeline/eMERLIN_CASA_pipeline.py -r
start_pipeline
```

Then it will fail and give you a critical error "start_pipeline". In general, if you choose a step which does not exist in the pipeline, it will give you this error. The best way to make this step run again, if you need to, is to choose another step and this will run at the start.

A note on reference antennas

In terms of what telescopes to use for reference antennas, it is best to use one of the "home-stations", I.e., in the central core of Darnhall/Pickmere/Mark II. The Lovell telescope should never be selected as your reference antenna and only participates in specific e-MERLIN observations, so in general it will not be present in your data. You will note that the pipeline often picks Cambridge as a reference antenna, owing to its slightly bigger diameter when

compared to other telescopes in the array. You can amend the refant in the global parameters of the default_params.json file.

run_importfits

This step takes the raw unprocessed data from the telescope and uses the CASA tasks “importfitsidi” and “mstransform” to transform the raw data from a fitsidi file into a CASA measurement set (MS). It will concatenate (merge) all fits files in the input folder, unflag the data so that it can be flagged later on, do some averaging (e.g., to 4s time intervals), generate some of the general parameters for the rest of the pipeline alongside the basic plots for the observation summary in the weblog (and other weblog tabs). This step will also generate the weblogs from scratch.

What does it produce?

This step will produce an MS file with the name Run.ms, in the case of the example dataset, this means TS8004_C_001_20190801.ms. Importantly, this MS will not be the same as the “_avg.ms” that is sent to PIs, as it is averaged in time but *not* frequency at this stage if pipeline default parameters are used. From this measurement set, the pipeline will generate the weblog files, including the folders (weblog and logs), and the casa_eMCP.log and eMCP.log files. The weblog is updated at this step to include all the information from the MS file, so the Home page is filled out, and all the observation summary tab is filled out except for a reference antenna choice. An initial flag statistics plot is made which should have a 0% flag amount at this stage, due to the “unflag” flagdata task run during this step of the pipeline.

What are the default parameters?

The default parameters for this stage can be found in the dictionary labelled “import_eM”. The default parameters are stated below, with comments on what they all do shown.

Parameter	Default	Comments
constobsid	true	This gives the same observation ID for each file input, so leave to true for concatenation of all fitsidi files
scanreindexgap_s	15.0	This should be set to the default of 15.0 for e-MERLIN data. This is to make sure that each source is in a different scan, but short scans on sources are not broken up.
antenna	“”	Choose which antennas to be included, usually left blank as default to include all antennas
field	“”	Choose which fields to be included, usually left blank as default to include all sources
timeaverage	true	Turn on time averaging with true.
timebin	“4s”	This will average the data to this time interval to make an avg.ms dataset, which is the standard e-MERLIN uses for reduction. Can be set to “1s” or timeaverage set to false to stop this from happening.
chanaverage	false	Turn off channel averaging – this is done later
chanbin	1	Default value is kept to 1 for the channel averaging channel bin.
usewtspectrum	false	If true, this will create a WEIGHT_SPECTRUM column in the MS file. This will weight by channel, which can be useful if you have

		flagged data by channel (like L band) and then you average, but we do not do this with e-MERLIN.
run_hanning	“auto”	Will run Hanning smoothing – this can be left as the default parameter as the eMCP will search the data to see if it is L/C/K band and only run this step if the data are at L band, where Hanning smoothing is required.
ms2mms	false	This converts the file to a multi-measurement set (MMS) format – only useful for parallelising processing which the e-MERLIN pipeline does not currently support
spw_separation	[“” ,” ”]	This does the continuum-to-narrow spw mapping for e-MERLIN spectral line data. Usually you will have 4 (or 8) continuum spectral windows, each with the same bandwidth, and 2-4 narrow 'zoom' spectral windows with a different bandwidth to the continuum spws (and they do not have to be the same between narrow spectral windows). For an observation with 4 continuum spws and 2 narrow spws, this would be set to ["0,1,2,3","4,5"]. In most cases, if you have set the “is_mixed_mode” parameter to “auto” in the global parameters, then the pipeline should work this out correctly, but it’s worth double checking!
spwmap_sp	[]	This should be an array of the continuum spws to apply to the narrow spw data, where the continuum spw should be the one that overlaps the narrow spw in frequency. For example, if you have an L band dataset with 4 continuum spws, and 4 narrow spws centred at 1612, 1665, 1667 and 1720 MHz, then this parameter would be set to [2,3,3,3]. Here, the continuum spws would go from 1252-1380, 1380-1508, 1508-1636 and 1636-1764 MHz. Again, the pipeline should work this out automatically, but there have been issues with this in the past so definitely double check this parameter!
fix_repeated_sources	false	In the case where a source with the same name and phase centre is loaded but with a different field ID, this will split each field ID individually, and then concatenate. The concatenation will check names and positions and merge them if they are the same. Usually not needed

Troubleshooting at this stage

Usually, this part of the pipeline does not fail. Whenever it does fail it can usually be traced back to an incorrectly set spw_separation or spwmap_sp in spectral line mode observations. These two parameters (only useful for spectral line observations) need to be carefully checked to make sure they represent the frequencies you expect. Following the notes on default parameters above, you can use the listobs output to decide what the correct values are. You can always use a continuum spw that does not directly overlap your narrow zoom spw, but it is preferable to use the overlapping spws.

Why do we average in time and not in frequency at this stage? We tend to average in time at this stage to reduce the data volumes by 4. The data are not flagged at this point, so all the RFI is still in the dataset. We do not average in frequency as we have not performed any RFI excision yet, so it is important for L band that this remains unaveraged, so we do not over flag, before performing a second averaging stage later in the pipeline.

flag_aoflagger

AOFlagger (Offringa+2013) is a great piece of software that goes through and removes instances of radio frequency interference (RFI) from the data. RFI is caused by many different electrical products like mobile phones, wi-fi and even electric fences! Similarly, you can pick up satellites when the sources are near the horizon, as well as thunderstorms and planes at Manchester airport (depending on the receiving band). The AOFlagger software goes through the data and removes the most egregious of these RFI from the data. It is important to note that *this step is not run on C band or K band data as the RFI environment is much better.*

What does it produce?

For L band, this step will run aoflagger to flag the data, and produces a plot in the flag statistics tab on the weblog. It will state in the eMCP.log what it has done to the data, and the casa_eMCP.log will have a full verbose output from aoflagger. Below is the eMCP.log output for an L band dataset (I.e., not this tutorial dataset):

```
2023-10-06 01:45:04 | INFO | ---------------
2023-10-06 01:45:04 | INFO | Updating weblog
2023-10-06 01:45:04 | INFO | [redacted] mode "auto"
2023-10-06 01:45:04 | INFO | L-band data, aoflagger will be executed
2023-10-06 01:45:04 | INFO | Start run_aoflagger_fields
2023-10-06 01:45:04 | INFO | Bands will be processed all together.
2023-10-06 01:45:04 | INFO | Checking AOflagger version
2023-10-06 01:45:04 | INFO | AOflagger version is 2.9.0
2023-10-06 01:45:04 | INFO | Running AOFLagger for field 0319+4130 (4) using strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis
2023-10-06 01:45:04 | INFO | Processing source 0319+4130, all bands
2023-10-06 01:46:37 | INFO | Last AOFlagger command: time aoflagger -fields 4 -strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis CY15213_L_Combine.ms
2023-10-06 01:46:37 | INFO | Running AOFLagger for field 080019+264205 (2) using strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis
2023-10-06 01:46:37 | INFO | Processing source 080019+264205, all bands
2023-10-06 02:00:04 | INFO | Last AOFlagger command: time aoflagger -fields 2 -strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis CY15213_L_Combine.ms
2023-10-06 02:00:04 | INFO | Running AOFLagger for field 0802+2509 (3) using strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis
2023-10-06 02:00:04 | INFO | Processing source 0802+2509, all bands
2023-10-06 02:08:03 | INFO | Last AOFlagger command: time aoflagger -fields 3 -strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis CY15213_L_Combine.ms
2023-10-06 02:08:03 | INFO | Running AOFLagger for field 1331+3030 (5) using strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis
2023-10-06 02:08:03 | INFO | Processing source 1331+3030, all bands
2023-10-06 02:10:10 | INFO | Last AOFlagger command: time aoflagger -fields 5 -strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis CY15213_L_Combine.ms
2023-10-06 02:10:10 | INFO | Running AOFLagger for field 1359+5544 (6) using strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis
2023-10-06 02:10:10 | INFO | Processing source 1359+5544, all bands
2023-10-06 02:11:25 | INFO | Last AOFlagger command: time aoflagger -fields 6 -strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis CY15213_L_Combine.ms
2023-10-06 02:11:25 | INFO | Running AOFLagger for field 1403+5418 (7) using strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis
2023-10-06 02:11:25 | INFO | Processing source 1403+5418, all bands
2023-10-06 02:14:00 | INFO | Last AOFlagger command: time aoflagger -fields 7 -strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis CY15213_L_Combine.ms
2023-10-06 02:14:00 | INFO | Running AOFLagger for field 1407+2827 (0) using strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis
2023-10-06 02:14:00 | INFO | Processing source 1407+2827, all bands
2023-10-06 02:15:25 | INFO | Last AOFlagger command: time aoflagger -fields 0 -strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis CY15213_L_Combine.ms
2023-10-06 02:15:25 | INFO | Running AOFLagger for field 1415+1320 (1) using strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis
2023-10-06 02:15:25 | INFO | Processing source 1415+1320, all bands
2023-10-06 02:15:49 | INFO | Last AOFlagger command: time aoflagger -fields 1 -strategy /scratch/processing/CY15213/eMERLIN_CASA_pipeline/aoflagger_strategies/default/default_faint.rfis CY15213_L_Combine.ms
2023-10-06 02:15:49 | INFO | ---------------
2023-10-06 02:15:49 | INFO | Start flagstatistics
2023-10-06 02:15:49 | INFO | Running flagdata on flag_aoflagger
2023-10-06 02:15:49 | INFO | mode="summary", action="calculate", antenna="*&*"
2023-10-06 02:28:45 | INFO | Saving flagtable in versionname="eMCP_flag_aoflagger"
2023-10-06 02:29:10 | INFO | flagstats file saved to: ./weblog/plots/plots_flagstats/flagstats_flag_aoflagger.pkl
2023-10-06 02:29:10 | INFO | Flag statistics ready. Now plotting.
2023-10-06 02:29:21 | INFO | End flagstatistics
2023-10-06 02:29:21 | INFO | End flag_aoflagger
2023-10-06 02:29:21 | INFO | ---------------
```

What are the default parameters?

Parameter	Default	Comments
run	"auto"	This is set to "auto" by default which will enable aoflagger if the dataset is in L band, taken from the MS metadata in the start pipeline steps. If it is not L band, then aoflagger will not run.
fields	"all"	Select which fields to run aoflagger on. It is often useful to run it on all fields, but if you find you get excessive flagging on bright calibrators at this stage then it is worth running on just the target and phase calibrator fields.
separate_bands	false	This will separate into multiple bands if the data has multiple bands in it, which 99.9% of the time, it will not, so keep to false.

Troubleshooting at this stage

So long as you have aoflagger installed, then this step should run fine. You need to have at least aoflagger 2.9 or above, but we have not yet configured the pipeline for aoflagger 3.0+ due to it requiring slightly modified flagging strategies. If you get the warning *"aoflagger requested but not available."* then check the version of aoflagger or whether it can be seen in your path.

flag_apriori

This step removes from known areas of bad data plus some additional extra removal of data due to the e-MERLIN band not being as sensitive in certain areas; it flags a few channels at the edge of each spw, and around 5% of the channels at either end of the band. The start of each scan on the targets (4s) and calibrators (120s) are removed or "quacked" from the data. You can see how many flags have been applied to the data by looking at the "Flag Statistics" tab of the weblog. In the example below, approximately 1/5 of the data have been removed by the time we get to the end of this step. This step produces another flag plot in the flag statistics tabs of the pipeline weblog and produces another saved flag table.

What are the default parameters?

Parameter	Default	Comments
border_chan_perc	5.0	This sets the percentage of the edge channels to use for flagging on either end of the band. Note that this applies to the full band. In this case, it will be 26 channels on the *unaveraged* ms file on either end of the band.
observatory_flags	true	During the data export stage at JBO, a set of observatory flags is in the form of a text file is produced. This flag file is comprehensive and lists all the times when the telescopes were not on source, so that they may be removed from the data.
do_estimated_quack	true	This will quack your calibrator fields using the all_quack or std_cal_quack parameters below. All_quack will only affect targets and phase cals, whereas the std_cal_quack parameter will affect the other cals.
all_quack	4.0	Set to 4s for targets and phase calibrators to remove the initial time stamp of a scan, but it will not be sufficient if telescopes were still slewing and observatory_flags have not been applied.
std_cal_quack	120.0	Set to 120s, this parameter will cut the first 2 minutes of time on every scan of 3c286 (1331+3030), oq208 (1407+2827) and 3c84 (0319+4130), to make sure that the telescopes are on source and little bad data is included. These calibrators are usually observed for >30 minutes in a single scan, so a 2-minute quack will not affect the results.
flag_Lo-Mk2	true	This will flag the shortest e-MERLIN baseline of ~200m, from Lovell to Mk2, if it is present in the data. It is full of correlated RFI and is difficult to calibrate given the baseline is so much shorter than all other baselines, so it should always be flagged.
observatory_flags	true	As above, this is an accidental additional parameter, so make sure it is set to what is set to above, to avoid confusing the pipeline!

Troubleshooting at this stage

If the pipeline fails at this stage, it is usually because of the observatory flags file not being available. If that is the case, you should turn the observatory_flags parameter to “false” in the default_params.json file and run the pipeline again, but you may want to double check the data after this and see if you need to extend the quack interval to something more like 20s. See the image below of what the failure may look like in this case:

```
2023-10-06 03:01:16 | INFO | Flagging initial/end channels: 0:0~26 and 7:485~511
2023-10-06 03:02:27 | INFO | Flagging first 4.0 seconds from all scans
2023-10-06 03:10:50 | INFO | Initial obs time 2023-05-18 09:50:31.500000
2023-10-06 03:10:50 | INFO | Final   obs time 2023-07-13 16:39:57.500000
2023-10-06 03:10:50 | INFO | Trying to retrieve observatory flags (locally)
antenna_monitor.log
                                    100%  172MB  17.0MB/s   00:10
Traceback (most recent call last):
  File "/pipeline1/emerlin/casa/casa-release-5.8.0-109.el7/lib/python2.7/init_welc
ome.py", line 33, in <module>
    execfile(__candidates[0])
  File "eMERLIN_CASA_pipeline/eMERLIN_CASA_pipeline.py", line 367, in <module>
    skip_steps = args.skip_steps)
  File "eMERLIN_CASA_pipeline/eMERLIN_CASA_pipeline.py", line 211, in run_pipeline
    eMCP = em.flagdata1_apriori(eMCP)
  File "/scratch/processing/CY15213/eMERLIN_CASA_pipeline/functions/eMCP_functions
.py", line 1268, in flagdata1_apriori
    finished_autoflag = search_observatory_flags(eMCP)
  File "/scratch/processing/CY15213/eMERLIN_CASA_pipeline/functions/eMCP_functions
.py", line 1186, in search_observatory_flags
    data = read_flag_database(logfile, t0, t1)
  File "/scratch/processing/CY15213/eMERLIN_CASA_pipeline/functions/eMCP_functions
.py", line 1103, in read_flag_database
    names=column_names)
  File "/pipeline1/emerlin/casa/casa-release-5.8.0-109.el7/lib/python2.7/site-pack
ages/numpy/lib/npyio.py", line 1769, in genfromtxt
    raise ValueError(errmsg)
ValueError: Some errors were detected !
    Line #4562965 (got 1 columns instead of 3)
```

flag_manual

This step is the first one that you can have some manual intervention over. In some of the datasets, you will find a manual.flags file (note this differs from the manual_avg.flags file). The manual.flags file (if present) includes a list of commands of parts of the data that the operations scientist has seen that looks incorrect, due to a correlator/observatory failure that has not been picked up by the observatory flags files. For example, data in August 2019 have an issue where the first spectral window (spw 0) on the Darnhall to Cambridge baseline (Da&Cm) was consistently causing errors in the calibration. So, in the case when this fault was affecting the data, it can be added to the operations scientist to the manual.flags file to remove it from the data completely, so it does not end up in the plots from the plot_data step or effecting the calibration at all.

As with the previous flag steps, it will make a new flag versions table and add a new plot to the flag statistics page of the weblog. There are no default parameters for this stage of the pipeline. If the pipeline fails at this stage, it is possible that there are typos in your manual.flags file, so double check this by reading the comments on flag files later in this document.

average

The average step will do some additional averaging if required, using the mstransform task in CASA. As standard, the data is averaged (already) to 4s time bins, but at this stage we also reduce the data size in frequency space too, averaging down every 4 channels into 1. By performing this averaging step, we speed up the calibration steps without compromising the data quality. It leaves you with 128 channels in each spectral window, of which you have 4 spws. Therefore, there are 512 channels across your entire 512 MHz band, but natively the data will have 2k channels.

What does it produce?

The most important thing that the pre_processing steps produces in the Run_avg.ms MS file, in this case TS8004_C_001_20190801_avg.ms file. This file is the one the pipeline will operate on throughout the rest of the pipeline procedures. This pipeline step will also re-generate a listobs file of the newly created avg.ms file.

What are the default parameters?

Parameter	Default	Comments
field	""	You can choose a field specifically to average down.
timebin	"4s"	As previously for time averaging, but, if you set timebin to "1s" then the timeaverage parameter will be set to "False", therefore if you have 4s averaged data, you will get 4s averaged data in the "_avg.ms" file.
chanbin	4	Now that the data have been flagged with aoflagger, we average in frequency by a factor of 4, reducing the dataset from 512 channels per spw to 128 channels per spw. You can increase this number to reduce the size of your dataset. If you set this to "1", then it will not do any channel averaging.
datacolumn	"data"	Your data should not have any calibration tables yet, only flag tables, so this should be kept to "data" and not changed.
timerange	""	You can choose a time range to do the averaging over, but this is usually not necessarily
scan	""	Again, you can choose specific scans to average, but you do not need to do that.
antenna	""	Choose specific antennas
shift_phasecenter	false	This parameter should be set to false for most runs of the pipeline. The only time you may need this parameter is if you want to move the phase centre of your target (or phase calibrator) sources in the visibility data before you start the calibration procedures. This parameter needs to be carefully used and there is a section at the end of this document on how to do the phasecenter shifting properly.

Troubleshooting at this stage

This part of the pipeline should normally proceed ok without any intervention. So, it should not fail here. For the shift phasecenter option, if it is used then you should follow the steps carefully at the end of this document, as it requires the pipeline to be called in two parts during this step.

plot_data

This step is straight forward and produces some basic plots of the data pre-calibration. These plots are often helpful when diagnosing where calibration steps may be going wrong later. They are shown as the "Uncalibrated visibilities" plots in the "Plots" tab on the weblog. There is only one default parameter, which is "num_proc", which should be left to the default value of "1". You should not get too many issues at this stage, but you may get an error about not being able to plot a source if it is not in the MS file, however it is rare to see this message as it requires changing the inputs.ini file in between earlier steps.

save_flags

This step saves everything you have done above into a single flag versions table called "initialize_flags". The idea is that this is a natural break in the pipeline before you do calibration and hence why the first step of the calibration part of the pipeline starts with restoring these flags. There are no parameters to be set at this stage as it is saving the current flag state for the calibration part of the pipeline and in theory the pipeline should not fail at this stage.

Calibration

The calibration part of the pipeline is the section that a PI will be able to re-run, modifying the manual_avg.flags file and default_params.json file to improve the calibration. We go through each of the steps below in turn:

8. restore_flags
9. flag_manual_avg
10. init_models
11. Bandpass
12. initial_gaincal
13. Fluxscale
14. bandpass_final
15. gaincal_final
16. applycal_all
17. flag_target
18. plot_corrected
19. first_images
20. split_fields

restore_flags

This step restores the “initialize_flags” flag version file to the data, so that calibration can proceed with all the flags inputted in the “pre_processing” section of the pipeline. There are no parameters to be set at this stage and it should not fail at this stage.

flag_manual_avg

This step of the pipeline will do several things, but the main point of it is a chance for you read in any manual flagging commands that you may want to input into the data. These commands are added as part of the “manual_avg.flags” file. Other than flagging, it will also run an initial gaincal step to find the best reference antenna, and update this in the weblog on the Observation Summary. It does this by looking at the SNR on the calibrator fields from the “avg.ms” file. It will also do additional flagging on the Lo drop out scans – these are scans where the Lovell remains on the target field instead of slewing to the phase calibrator, as it has a slow slew rate, so it borrows the phase information from the Mk2 for calibration.

What does it produce?

Like previous flag steps, it will create a flag versions file and generate a flag plot for the Flag statistics tab of the weblog. It will also report in the terminal the number of antennas sorted by the percentage of good solutions and the SNR from the initial gaincal, and picks the one with the highest percentage as the reference antenna. For example:

```
2023-10-06 13:17:41 | INFO | Antennas sorted by % of good solutions:
2023-10-06 13:17:41 | INFO | Pi : 99.6, <SNR> = 15.8
2023-10-06 13:17:41 | INFO | Mk2: 99.6, <SNR> = 16.7
2023-10-06 13:17:41 | INFO | Cm : 83.2, <SNR> = 16.1
2023-10-06 13:17:41 | INFO | Da : 80.8, <SNR> =  7.0
2023-10-06 13:17:41 | INFO | Kn : 68.9, <SNR> = 15.3
```

```
2023-10-06 13:17:41 | INFO | De :  8.5, <SNR> =  4.4
2023-10-06 13:17:41 | INFO | Refant in eMCP: Pi,Mk2,Cm,Da,Kn,De
```

In the case above Pickmere is chosen as the main reference antenna, but Mk2 will be used as a secondary one if Pickmere fails. You will note also that the SNR is listed, so in this case it may also be worth choosing Mk2 as the reference antenna, so it is well worth looking at this information before deciding on a reference antenna. Note that if you set a refant in the default_params.json global parameters, then this mini step will not be run, so you will not get this information. Therefore, it is always useful to run this step first to have a quick look at reference antennas.

What are the default parameters?

All the parameters below should be left to defaults unless you 1) have Lovell in the dataset, 2) there are Lovell dropout scans, and 3) the automated search for the dropout scans is not working.

Parameter	Default	Comments
Lo_dropout	""	This should be a list of scans where the Lovell "drops outs" of the observation. The Lovell "drops out" every other phase calibrator run as it has a slower slew speed than the other telescopes, so the pipeline will use the Mk2 to interpolate phase solutions for the Lovell when the Lovell "drops out". You should not need to set this as the pipeline will automatically try to work out which scans are "dropout" scans for the Lovell.
Lo_datacolumn	"data"	This is the data column in which to search for Lovell drop out scans.
Lo_useflags	true	This is used for searching for Lo drop out scans.
Lo_spws	["3"]	This is the spw in which to search for Lovell drop out scans.
Lo_threshold	0.5	This is the threshold of the amplitude in which to search for Lovell drop out scans.
Lo_min_scans	""	This is used for finding Lovell drop out scans.

Troubleshooting at this stage

Like other flagging steps, this should not fail. However, the manual_avg.flags file may fail if there are syntax errors in it: see the section on these files further down in this document or consult the flagdata CASA manual.

Init_models

This step is crucial to the calibration of the pipeline procedures as it will read in the 3c286 model and add a CORRECTED_DATA column to the data which can then be used for the rest of the pipeline procedures. It first runs "clearcal" to make the CORRECTED_DATA column, then it runs "initweights" and "delmod" to remove any previous models in the data, before running "setjy" on 3c286 with the e-MERLIN calibrator models using the 2017 Perley Butler values for the source. This is important to do as 3c286 is slightly resolved on e-MERLIN scales, making it imperative that we give it a good model from which to extrapolate the correct fluxes from – if we observe 3c286 at a different elevation we will get a different flux, therefore the model is needed to overcome this. This step does not produce anything in the weblogs but does create the CORRECTED_DATA column in the "avg.ms" file and clears previous models from the data to put the 3c286 model in there with setjy.

What are the default parameters?

Parameter	Default	Comments
calibrator_models	"calibrator_models/"	This is the folder in the eMERLIN_CASA_Pipeline folder where all the calibrator models live. Leave this to the default.
manual_fluxcal	false	This should be left to false if you want to use 3c286 as a flux calibrator. If you are using anything else as a flux cal, then set this to true and fill out the following parameters.
fluxcal_flux	[-1]	This should be specified as [I, Q, U, V], but for the purposes of the pipeline the Q, U V values will not be used, so you can just set the I value here.
fluxcal_spix	0.0	The spectral index of the source
fluxcal_reffreq	"0GHz"	The reference frequency of the observations of the source.
wtmode	"nyq"	
dowtsp	false	This should be left to false, as it will put in one weight per spw if false. If True, then it will place one weight per channel (see usewtspectrum parameter from the import_eM section)

Troubleshooting at this stage

Usually this step should run successfully, as it only sets the models in the dataset for 3c286. However, when setting the flux manually, it is important that you set all the values properly in this section. These should be set to those below, for example if using 3c84 as your flux calibrator. At the same time, you should set 3c84 to be the flux calibrator in the inputs.ini file so the eMCP knows that that is the intended flux calibrator and not 3c286.

```
"manual_fluxcal"          : true,
"fluxcal_flux"            : [30.0],     <- set to calculated flux from eMCP run
"fluxcal_spix"            : -0.1,       <- set to calculated spectral index from eMCP run
"fluxcal_reffreq"         : "5.0GHz",   <- set to central frequency from listobs output
```

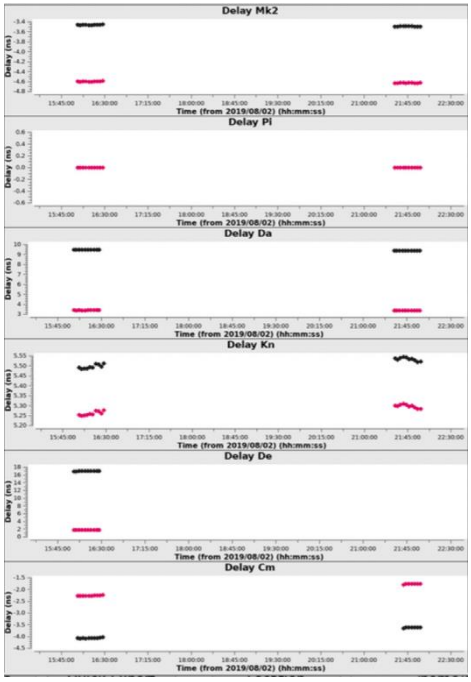
Bandpass

This first part of calibration procedures for the pipeline creates a bandpass table. To do this, it makes several subsidiary tables and works through various gaincal steps. It is important to note that the pipeline will run this step twice: first to compute all the tables using the band pass calibrator before doing a flagging step and then re-running the steps again to produce an improved band pass table.

What does it produce?

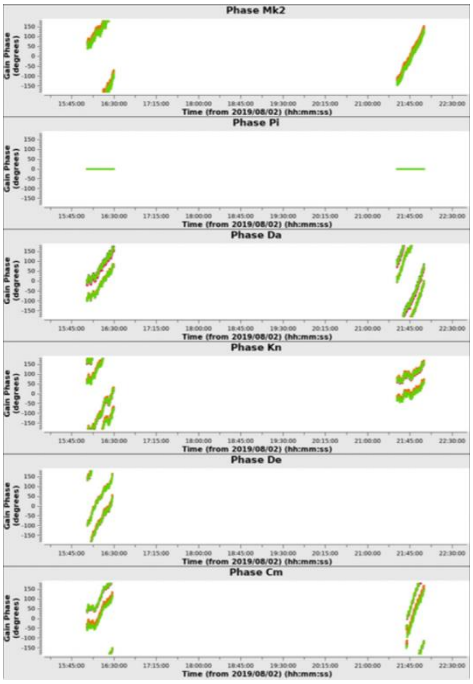
To make the bandpass table, it first performs a preliminary delay solution (creating bpcal_d.K0), a preliminary phase calibration table (bpcal_p.G0), and a preliminary Amplitude and Phase (ap) calibration table (bpcal_ap.G0). These are all performed for the band pass calibrator OQ208. It then has the necessary information to make the band pass table, bpcal.BP0. Below are the examples for the 3C277.1 dataset and what to look out for:

bpcal_p.K0



This first table shows the delay solutions over the time of the observations for the band pass calibrator. It creates a solution every 3 mins, over which the delays should be stable. A few things to make note of here are the absolute delay numbers – these should not normally be more than ~10 ns and should not vary between more than a few ns over the course of a scan. The two colours show the two polarisations (LL and RR) and it is normal for them to have slightly different delay offsets, but they should follow one another temporally. It may not be obvious at this stage, but sometimes you get delay jumps which can happen when the maser time signal from the telescopes becomes unstable. This can happen for a variety of varied reasons, but one common one is when it gets too hot and the light paths get affected from the telescopes, so we get small delay jumps which appear in the data. The pipeline can deal with these, so you should not need to worry about jumps, but delay variation over time is a problem which could require fringe fitting at the later stages of the pipeline. Notice here that Defford has a large offset between LL and RR correlations? This is normal for Defford and is instrumental but will be calibrated out through the pipeline steps. You can also see here that the pipeline has flagged the second scan of OQ208 for Defford and made Pickmere the reference antenna as it is delays are zeroed, i.e., everything here is relative to Pickmere.

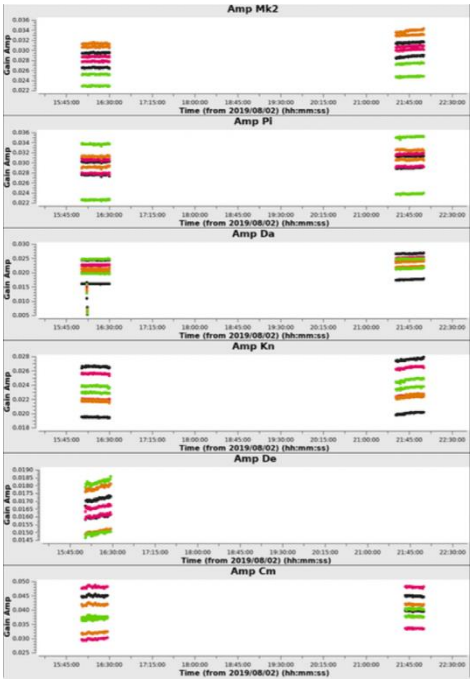
bpcal_p.G0



This solution table solves for the phases between the antennas for the band pass calibrator. The solution interval is now the integration time of the source, in this case 4s, but you can extend this to a longer time interval i.e., 20s, if you have poor SNR data. You should see here a clear evolving phase over time, with no significant excursions. Delay jumps may show up here too as small jumps, and anything which looks like there is no coherent phase (i.e., it looks like noise) should be flagged if it does not get picked up by the pipeline. The data here are colourised by the spws, so you can see two lines for each, which are the two polarisations. The phase will wrap around 180 degrees, so be worried if it appears to jump back to the bottom of the plot here, only be concerned if you see a vertical line at the same time stamp as that suggests the phase calibration hasn't gone well.

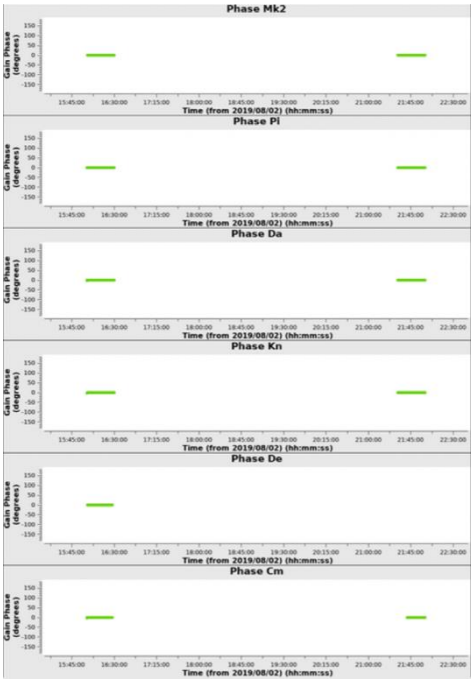
bpcal_ap.G0

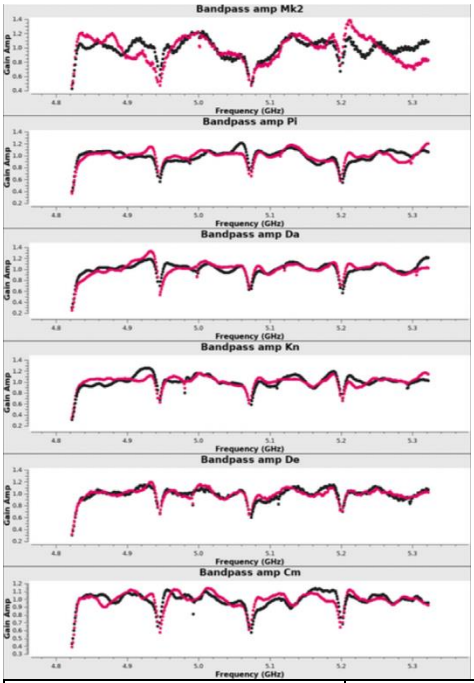
The amplitude and phase table provide two plots. One should be a phase plot with the phase now zeroed, thanks to our previous phase calibration table which we have applied to make this table, but also an amplitude vs time plot showing how the solutions for the amplitude, usually over a longer time interval such as 32s in this case. It is important that you check any areas of the data where the phase is not zero, and any areas where the amplitude drops or spikes. For example, here there is a small drop out on Darnhall in the first scan – this should be noted to be flagged from the data at a later point. Once again, the data is colourised by spw, so you should have two lines per colour, one for each correlation.



bpcal.BP0

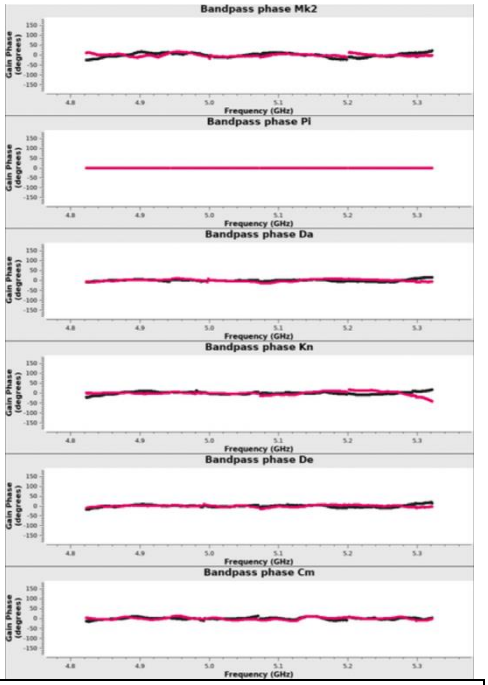
This is the finalised band pass table, and you will be presented with the solutions from applying all the previous 3 tables and then running the bandpass task in CASA. Like the previous table, you have an amplitude and phase plot on the left and right, respectively. The plots are generated with a solution interval of "int" and displayed as amp/phase vs frequency, so you should be able to see the effect of the band pass across the band. The amplitude plot shows 4 clear spws, with the edges of the band flagged entirely due to roll off, and dips in the amplitude at the edges of each of the spws. There are some wiggles across the band, but this is simply the instrumental response, and you should expect the two lines (each colour is a different correlation) to follow one another closely. If one correlation is significantly different from the other, then there is usually a problem with that spw and/or correlation that needs to be flagged. The phase plot at this point should be flat and set to zero, but it is ok if there are a few small wiggles.





What are the default parameters?

There are a lot of default parameters in this section, and many of them follow the same ordering as multiple runs of gaincal are made with slightly different parameters. Therefore, they have been grouped to make it easier to follow.



Parameter	Default	Comments
delay_tablename phase_tablename ap_tablename bp_tablename	"bpcal_d.K0" "bpcal_p.G0" "bpcal_ap.G0" "bpcal.BP0"	This sets the table names for the relevant tables to be made and then applied in this section. There are four tables, outlined here.
delay_solint phase_solint	"180s" "int"	These are the default solints for each of the tables: 3 mins for the band pass, integration time (4s in

ap_solint bp_solint	“32s” “inf”	this case) for the phase table*, 32s for the combined ap solve, and infinite (I.e., across the whole observation) for the final band pass table. *You can increase this to ~20s if you’re worried that the data are a bit noisy, and you need a bit more to get good phase solutions.
delay_combine phase_combine ap_combine bp_combine	“spw” “” “” “field,scan”	You can choose to combine via different methods if you need the additional SNR, but in most cases, you shouldn’t need it. Here, the delay combines over spw as the instrumental delay should remain the same across all spws, whereas the phase and amplitudes may vary so usually shouldn’t be combined. We combine by field and scan for the final band pass table as you want a single solution for all of the band pass observations.
delay_prev_cal phase_prev_cal ap_prev_cal bp_prev_cal	[] [″bpcal_d.K0″] [″bpcal_d.K0″,″bpcal_p.G0″] [″bpcal_d.K0″, ″bpcal_p.G0″, ″bpcal_ap.G0″]	An array of the previous calibration tables, and you need to apply them in turn to the next step.
delay_interp phase_interp ap_interp bp_interp	″linear″ ″linear″ ″linear″ ″nearest,cubicflag″	This determines the type of interpolation type for each of the tables. Linear interpolation is preferred for the first three tables to ensure the interpolation is linear in time across the whole observation, but the last one (for the BP0 table) has a nearest and cubicflag interpolation. These refer to using the nearest calibrator in time space, and then a cubic in frequency.
delay_spw phase_spw ap_spw bp_spw	[″*″,″innerchan″] [″*″,″innerchan″] [″*″,″innerchan″] [″*″,””]	We use the asterisk to refer to all spws to apply to, and the innerchan parameter will only work on the inner 80% of the band for each spw, so the poorer SNR at the band edges do not affect the calibration. Of course, you need to calibrate these for the BP0 table, so this is removed from that part.
delay_minblperant phase_minblperant ap_minblperant	3 3 3	Number of baselines for an antenna to produce a satisfactory solution in any given solution interval. Can be dropped to 2 for more solutions and 1 for a point source but be careful - dropping to 2 is ok, but 3 is preferred. 1 can be used as it is like doing a single baseline
delay_minsnr phase_minsnr ap_minsnr	2 2 2	This is the snr ratio on solutions below which the data should be flagged. Should be fine at 2 to preserve as much as possible, but you can drop to 1 if you are careful!
bp_uvrange	“”	You can stipulate a uvrange if you think your calibrator is resolved, but you should not need to for OQ208 as its unresolved on e-MERLIN baselines.

bp_fillgaps	8	Interpolates over flagged channels by interpolation. This default should be fine.
bp_solnorm	true	Normalise the solutions by dividing through by product of gains so that average gain is 1 per pol/antenna/spw. You should do this as it will make all the relative calibrations later in the pipeline easier.
apply_calibrators	["bpcal.BP0"]	This is used to apply to the data after the preliminary set of gaincal tasks, so that the second run through with flagging is closer to the correct answer.
apply_targets	[]	You should not need to apply to targets at this stage, but if you want to have a look then you can.
run_flag	true	This parameter will run a flagging routine at the end of the calibration cycle, to remove any low level RFI. All the parameters after this refer to whether this is set to true or false. Usually, it should be set to true to remove bad bits of data but can be set to false if you have a large flag percentage. Note that if set to false, care must be taken to manually excise the data and rerun the pipeline with the manual_avg_flags written.
mode	"tfcrop"	This will run tfcrop at the end of the preliminary gaincal runs. All the parameters after this are described in the flagdata online documentation . You should not need to change them at all.

Troubleshooting at this stage

You can have a look at the log files at this stage as the gaincal runs will tell you a lot about the data being flagged and whether CASA thinks it should find solutions or not. Remember also that this step runs all the tasks once, performs some flagging, and then re-runs all the gaincal steps once more. So, it is worth comparing what the solution levels are in the log files to make sure that the flagging is not causing significant issues, but it rarely does. In this case only 0.4% more flags are made during this step.

Initial_gaincal

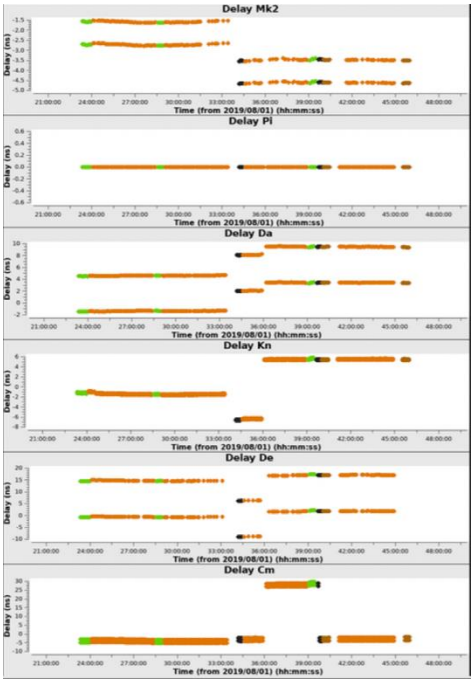
This step will do the main phase and amplitude calibration of the entire dataset, operating on all calibrators. It is the most important part of the pipeline and is liable to fail if your calibrators are not particularly good, so care must be taken at this part of the pipeline to ensure that the final solution tables are good. Like the previous step, this one also runs twice, once to do a preliminary calibration before doing some flagging and then re-doing the calibration.

What does it produce?

This will make three plots: the allcal_d.K1 delay calibration plot, the allcal_p.G1 phase calibration plot and the allcal_ap.G1 amplitude/phase calibration plot. It will also produce another flag statistics plot.

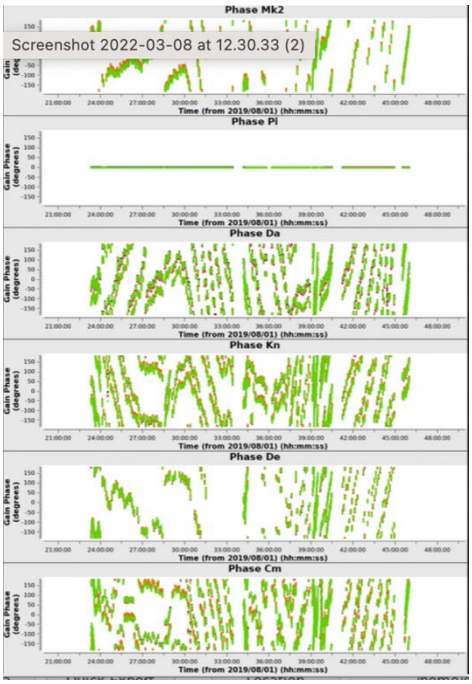
allcal_d.K1

This table represents the delay calibration across all the calibrators. You expect that the delays should be zero for the reference antenna and ideally for all other antennas, but you can see here that this is not the case. However, the fact



that the delays are constant in time with jumps suggests that these are small regions where we have had a delay jump in the correlator which CASA can handle and take of in the solutions. So, these solutions are fine. What we do not want to see is a gradual change over time, i.e., a drifting of the delays, as this would suggest that there is a rate that needs to be taken care of and some sort of fringe fitting is necessary. If you have multiple delay jumps on short timescales, then you might want to change the solint at this step to a shorter one to consider small jumps. As before, the colourisation in this plot is by spw.

allcal_p.G1

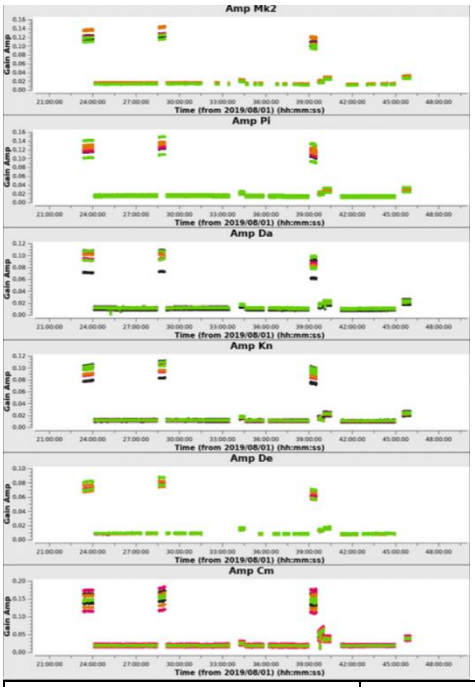


This table is the phase calibration table. It uses the integration time as the solint to create the table, but you can increase this if the phase calibrator is weak or the signal to noise ratio for the dataset is poor. You will see that the reference antenna is zeroed here, and that the other antennas show phase wraps over time. This is completely normal and just shows that the gaincal step has considered the atmospheric phase properly. If it has not then you will see vertical lines in this plot, I.e., noise, which should be flagged. Once more the colours represent the spws.

allcal_ap.G1

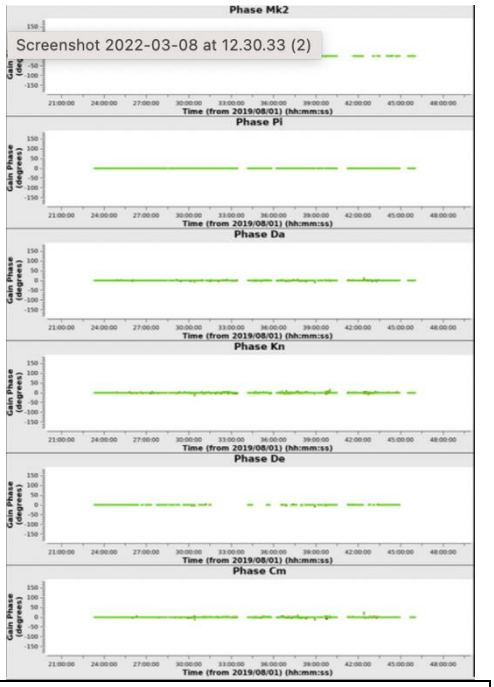
This is the amplitude and phase solution table for all calibrators. Just like the bandpass ap solution table, we get two plots, one for amplitude and one for phase. The jumps we see in the amplitude plot represent the different calibrator sources, so do not worry if you see jumps in this plot! You should expect to see regular jumps between main caibrators (3c286/3c84/OQ208) and your phase calibrator, with the phase calibrator remaining at the same amplitude throughout for each antenna.

If you look at the Cambridge baseline there appears to be a little amplitude jump at around 40:00:00 so this should be flagged. As previously for ap plots, you should make sure the phases also remain constant at this stage as we have applied the previous phase table to the data to make this plot. It is also useful at this point to look for gaps in the dataset in time, for example look at Mk2 around 33:00:00. It appears to have lost some solution around this time, so it may be worth checking the data to see if there are specific problems with this antenna at this point. As everything is referenced to the reference antenna, it can also represent a problem on the refant, so it is worth checking that too! For the sake of this demonstration though, these are good plots, and by the end of this section, we have only got 31% flags, an increase of 8% or so.



What are the default parameters?

The default parameters are split into three sections here, so we start with the delay tables, then the gaincal as performed in previous sections, and then options for flagging.



Parameter	Default	Comments
use_fringefit	false	We do not need to fringe fit the data as we have already done a delay calibration in the bandpass steps, but if you have rate issues then you can try it. It uses the fringe fitter in CASA so use at your own risk.
tablename	"allcal_d.K1"	Table name to generate
delay_cal	"default"	Use the default delay calibrator, I.e., OQ208 in this case.
solint	"180s"	As before, the instrumental delay should not evolve too quickly so a 3-minute solint should be fine here. But it may be worth dropping to 120s if you are worrying about getting a solution for every phase calibrator scan.
combine	"spw"	Combine by spw as the instrumental delay should be the same across all spws.
prev_cal	["bpcal.BP0"]	Apply our bandpass table made in the previous step.
interp	"linear"	LInear interpolation for the delays in time as we want to track this over the course of the of the observation.
spw	["*", "innerchan"]	As previous, we use only the inner channels to calibrate where there is the most SNR and least band roll off effects and apply it to all spws.
zerorates	true	This is only relevant if fringe fitting is turned on
minblperant	3	As before for minblperant. You will want to keep this at 2 as a minimum
minsnr	2	As previously for minsnr.

Next are the phase and amplitude and phase gaincal runs. These are more-or-less identical to the parameters in the bandpass section, but with some subtle differences to include the delay calibration just performed.

Parameter	Default	Comments
p_tablename ap_tablename	"allcal_p.G1" "allcal_ap.G1"	This sets the table names for the relevant tables to be made and then applied in this section. There are two tables now, outlined here. Note also that the suffix has changed to G1 from G0 previously.

p_solint ap_solint	“int” “32s”	As previously for the bandpass part of the pipeline
p_combine ap_combine	“” “”	As previously for the bandpass part of the pipeline
phase_prev_cal ap_prev_cal	[“bpcal.BP0”, “allcal_d.K1”] [“bpcal.BP0”, “allcal_d.K1”, “allcal_p.G1”	As previously for the bandpass part of the pipeline, but this time we use the relevant tables here and use the BP0 table to start with to make sure we take any effects of the band pass out first.
p_interp ap_interp	“linear” “linear”	As previously for the bandpass part of the pipeline
p_spw ap_spw	[“*”,”innerchan”] [“*”,”innerchan”]	As previously for the bandpass part of the pipeline
phase_minblperant ap_minblperant	3 3	As previously for the bandpass part of the pipeline.
phase_minsnr ap_minsnr	2 2	As previously for the bandpass part of the pipeline.
apply_calibrators	[“bpcal.BP0”, “allcal_d.K1”, “allcal_p.G1”, “allcal_ap.G1”]	Apply all the tables generated here to the calibrators.
apply_targets	[]	Do not apply tables to targets at this stage, but you can do if you want.
flagmode	“tfcrop”	This allows you to pick from tfcrop or rflag to flag the visibilities. See the next set of default parameters for explanations.

The final set of default parameters for this section concerns the flagging at the end of this step. As alluded to, you can pick between rflag and tfcrop as your main flagging strategies here. First the tfcrop parameters are listed, I.e., the first parameter is “mode”: “tfcrop”, and you have set “flagmode”: “tfcrop”, and then those for rflag. I will not go into these here as they are outlined in the CASA documentation noted earlier.

Troubleshooting at this stage

This part of the pipeline is usually where it fails (if it is going to), because it depends on the phase calibrator being good throughout the observation. Any issues can easily be compounded at this stage and lead to a cascade of flagging which leads to a failure of the pipeline. If this part of the pipeline fails, then it is best to go through gaincal by gaincal and check that the number of solutions is sensible and make some minor changes to the default parameters file if necessary. Remember that this runs twice, so once without flagging and once with flagging, so if you have a high flag percentage then this could explain why it fails.

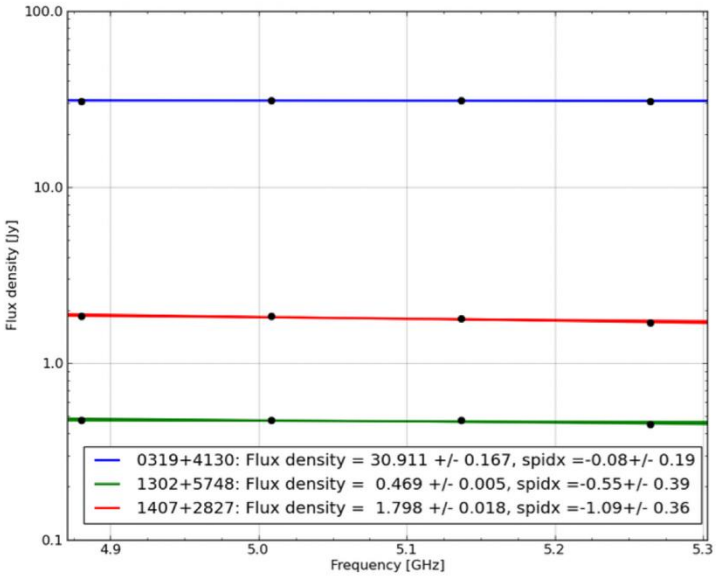
fluxscale

This step will scale the fluxes by running fluxscale and setjy to transfer all the fluxes from the model of 3c286 to the

calibrators. It is utmost importance that this step is successful and produces sensible fluxes and spectral index values for the sources, as it will affect all of the final fluxes on the target field.

What does it produce?

This step produces one plot, the fluxscale plot on the right, and returns the fluxes set by the pipeline. You can see that each source has a different colour, and if you have a suitable calibration across the entire band, then you should expect to see a single line crossing all data points (spws). This step also produces a text file called allcal_ap.G1_fluxscaled_fluxes.txt, which includes an "eMfactor". This factor is a scaling factor due to 3c286 being resolved out on the shortest baselines by e-MERLIN. Therefore, the log values will be corrected by this factor and the ones in the text file are not, but there is a warning.



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)
# Flux density for 0319+4130 in SpW=0 (freq=4.8805e+09 Hz) is: 31.0463 +/-
0.429427 (SNR = 72.2971, N = 12)
# Flux density for 0319+4130 in SpW=1 (freq=5.0085e+09 Hz) is: 31.396 +/-
0.463267 (SNR = 67.7709, N = 12)
# Flux density for 0319+4130 in SpW=2 (freq=5.1365e+09 Hz) is: 31.3674 +/-
0.491595 (SNR = 63.8074, N = 12)
# Flux density for 0319+4130 in SpW=3 (freq=5.2645e+09 Hz) is: 30.7706 +/-
0.516247 (SNR = 59.6043, N = 12)
# Flux density for 1407+2827 in SpW=0 (freq=4.8805e+09 Hz) is: 1.86567 +/-
0.0945614 (SNR = 19.7297, N = 12)
# Flux density for 1407+2827 in SpW=1 (freq=5.0085e+09 Hz) is: 1.86602 +/-
0.0999827 (SNR = 18.6634, N = 12)
# Flux density for 1407+2827 in SpW=2 (freq=5.1365e+09 Hz) is: 1.80584 +/-
0.104122 (SNR = 17.3434, N = 12)
# Flux density for 1407+2827 in SpW=3 (freq=5.2645e+09 Hz) is: 1.7089 +/-
0.112365 (SNR = 15.2085, 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
# Fitted spectrum for 0319+4130 with fitorder=1: Flux density = 31.1468 +/-
0.167987 (freq=5.07048 GHz) spidx: a_1 (spectral index) =-0.0752787 +/-
0.190325 covariance matrix for the fit:  covar(0,0)=5.85006e-05
covar(0,1)=0.000673298 covar(1,0)=0.000673298 covar(1,1)=0.386241
```

```
# Fitted spectrum for 1407+2827 with fitorder=1: Flux density = 1.81183 +/-
0.0184906 (freq=5.07048 GHz) spidx: a_1 (spectral index) =-1.08585 +/- 0.364224
covariance matrix for the fit:  covar(0,0)=0.000815434 covar(0,1)=0.0123799
covar(1,0)=0.0123799 covar(1,1)=5.50668
# 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.
```

What are the default parameters?

Parameter	Default	Comments
tablename	"allcal_ap.G1_fluxscaled"	The table name to generate after flux scaling.
ampcal_table	"allcal_ap.G1"	The amplitude table name.
apply_calibrators	["bpcal.BP0", "allcal_d.K1", "allcal_p.G1", "allcal_ap.G1_fluxscaled"]	Apply all our previously generated tables.
apply_targets	[]	Do not apply to the targets at this stage.

Troubleshooting at this stage

A couple of things to be aware of here are the spectral indices that are calculated by the pipeline and the fluxes of the sources. You should be able to look up your sources in the astrogeo vlbi calibrator list to check that the fluxes are coming out about right for the sources. Note that 3c84 is variable so do not use that to check! OQ208 is usually stable, so ~1.8Jy at C band is about right here. The spectral index values also look reasonable but keep a look out for alpha values of +7 or –7, which suggests something has gone wrong with the flux scaling!

Also keep aware that 3c286 is slightly resolved at e-MERLIN baselines. The pipeline will solve for this using the dfluxpy script and give an eMfactor correction factor. Then, the pipeline will run the script dfluxpy to find the correction factor eMfactor. The values in the logger in eMCP.log are corrected by this eMfactor, but the fluxscaled_fluxes.txt file is not corrected by this factor.

bandpass_final

Now we have an accurate flux scale, we re-make the bandpass table to include the spectral index of the calibrators as the previous bandpass tables assumed the source was flat spectrally (same flux at all frequencies/spws). This step is like the initial "bandpass" step, but this time we have a delay, phase and ap solution table which we can use to derive the bandpass table, and a flux scale table which enables us to fit the bandpass calibrator, considering any spectral index variations.

What does it produce and what are the default parameters?

A new bpcal.BP2 table which is the new band pass table that we use for the rest of the pipeline. The default parameters are the same as in the previous band pass steps, but only for re-creating the BP table.

Parameter	Default	Comments
bp_tablename	"bpcal.BP2"	As previous for bandpass step
bp_solint	"inf"	As previous for bandpass step
bp_combine	"field,scan"	As previous for bandpass step

bp_prev_cal	["bpcal_d.K0", "bpcal_p.G0", "bpcal_ap.G0"]	As previous for bandpass step
bp_interp	"nearest,cubicflag"	As previous for bandpass step
bp_spw	["*",""]	As previous for bandpass step
bp_uvrange	""	As previous for bandpass step
bp_fillgaps	8	As previous for bandpass step
bp_solnorm	true	As previous for bandpass step
apply_calibrators	["allcal_d.K0", "bpcal_p.G0", "bpcal_ap.G0", "bpcal.BP2"]	As we are re-generating the bandpass table, we want to re-apply it on top of the other calibrations we have already made, so unlike the bandpass step, we are applying our newly made BP2 table.
apply_targets	[]	As previous for bandpass step

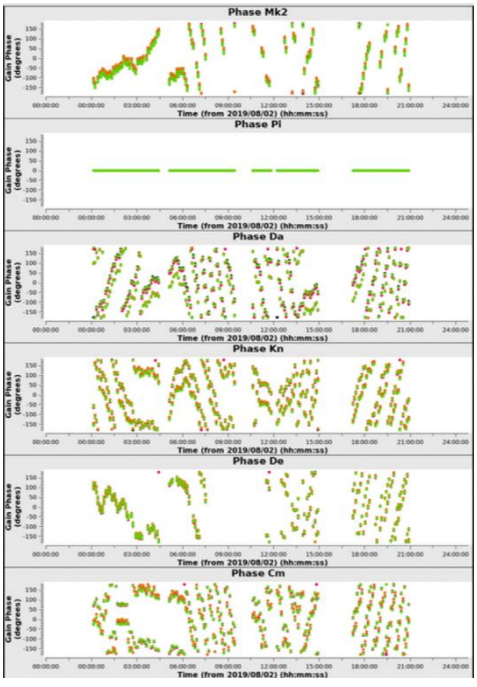
Troubleshooting at this stage

This stage is unlikely to fail – if the flux scaling is good then this will usually run safely and produce a new bandpass table.

gaincal_final

Like the initial_gaincal steps, this set of tasks will (re-)compute the phase and the ap solution tables, as well as making per-scan solution tables which will then be used to apply to the target fields.

What does it produce?



This step produces 4 main plots, a phase calibration plot, an ap calibration plot (exactly like those in the initial_gaincal step), as well as the scan averaged phase and ap table plots. The phase and ap calibration plots are identical to those in the previous initial_gaincal step so here we only go over the per-scan plots:

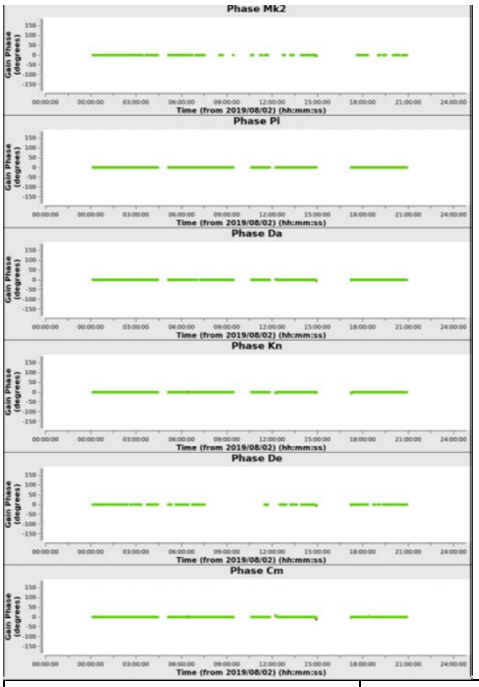
phscal_p_scan.G3

This is the per-scan phase calibration plot, providing a phase solution per-scan so that it may be interpolated across to the target in later steps. Like other phase calibration plots, you should expect to see a smooth variation of phase over time, with the refant being set to zero. Colours are once again the spws.

phscal_ap_scan.G3

Like the per-scan phase plot above, this is the amplitude and phase version. You should expect to see your amplitudes to not vary, or if they do to not vary much over time. You can see that there is some small long-term timescale variation, but this should be ok. You can also see some dropouts in the amplitude plots on Darnhall and

Cambridge, as well as Defford being noisier than the rest of the data. We should look at the data at this point and see if there are any issues.



What are the default parameters?

The default parameters for this section are similar to the previous gaincal parameters, so I have removed those which are for the p and ap tables, but I have left those for the per-scan tables. Note that this time we apply the new BP2 table but use the initial delay table.

Parameter	Default	Comments
p_scan_tablename ap_scan_tablename	"phscal_p_scan.G3" "phscal_ap_scan.G3"	As previously for the other p/ap tables but now for scan, these are the table names.
p_scan_prev_cal ap_scan_prev_cal	["bpcal.BP2", "allcal_d.K1"] ["bpcal.BP2", "allcal_d.K1", "allcal_p.G3"]	Apply the BP2 table and the delay tables already computed first.
p_scan_solint ap_scan_solint	"inf" "inf"	These are set to inf, as we want a solution per-scan
p_scan_spw ap_scan_spw	["*", "innerchan"] ["*", "innerchan"]	As per previous p/ap tables, we apply to all spws and usually only the higher SNR inner channels to make our solutions.
p_scan_combine ap_scan_combine	"" ""	As per previous gaincal steps.
p_scan_interp ap_scan_interp	"linear" "linear"	As per previous gaincal steps.
p_scan_minblperant ap_scan_minblperant	3 3	As per previous gaincal steps.
p_scan_minsnr ap_scan_minsnr	2 2	As per previous gaincal steps.
ap_calibrator	"default"	This parameter allows you to choose a specific calibrator if you have a separated amplitude calibrator that is further away from your phase calibrator. This is

		useful for K band observations, and you can read about that as an extra section in this document.
apply_calibrators	["allcal_d.K1", "bpcal.BP2", "allcal_p.G3", "allcal_ap.G3"]	Apply all our solution tables to the calibrators, as they were created for the calibrators.
apply_targets	["allcal_d.K1", "bpcal.BP2", "phscal_p_scan.G3", "phscal_ap_scan.G3"]	Here we apply the global delay and final band pass tables as before, but we apply the per-scan tables to the targets.

There are also some parameters in this section that are exclusively for spectral line datasets. These all include "offset" or "narrow" in the parameter name. For continuum work you can safely ignore these parameters. These do two things. The "offset" tables (and related parameters) compute the offsets between the continuum spw and the narrow spw and applies it to the narrow spw. The "narrow bp" tables re-computes the bandpass over the narrow spw as it may be different to the continuum spw band pass in that part of the band.

Parameter	Default	Comments
p_offset_tablename narrow_bp_tablename	" narrow_p_offset.G3" "narrow_bpcal.BP2"	Table names
p_offset_prev_cal narrow_bp_prev_cal	["allcal_d.K1", "allcal_p.G3"] ["allcal_d.K1", "allcal_p.G3", "allcal_ap.G3", "narrow_p_offset.G3"]	Apply the previous tables
p_offset_solint narrow_bp_solint	"int" "inf"	Phase offset is done on the integration time as a solint, as we want to see how the phase evolves over time. You could increase this to ~20s if you needed to for SNR requirements. The bp table requires inf as we want a single solution for all times.
p_offset_spw narrow_bp_spw	["*",""] ["*",""]	The spw to apply to, this time we do not use innerchan as the band itself has a lot of channels (512) compared to the spw band width, and we want to apply to everything.
p_offset_combine narrow_bp_combine	"scan" "field,scan"	We can combine by scan for the phase offset as it should remain the same throughout all scans, and we also combine by field for the bp table as that should be the same across all bp cal scans.
p_offset_interp narrow_bp_interp	"linear" "nearest,cubicflag"	As previously for interpolation for phase and band pass tables.
p_offset_minblperant	3	As preivously
p_offset_minsnr	2	As previously
narrow_bp_uvrange	""	As previously, you can restrict the uv range if you want to.
narrow_bp_fillgaps	8	As previously
narrow_b_solnorm	true	Normalise the solutions as previously

narrow_apply_calibrators	["allcal_d.K1", "narrow_bpcal.BP2", "allcal_p.G3", "allcal_ap.G3", "narrow_p_offset.G3"]	Apply the tables in order. In this case we use the initialy delay table, but then apply the new narrow bpcal table before applying the G3 p and ap solution tables and then the offset table when apply to calibrators.
narrow_apply_targets	["allcal_d.K1", "narrow_bpcal.BP2", "phscal_p_scan.G3", "phscal_ap_scan.G3", "narrow_p_offset.G3"]	As previously, but here the difference is we use the per-scan p and ap solution tables to apply to the targets.

Troubleshooting at this stage

Like the initial_gaincal part of the pipeline, this step can fail due to heavy flagging on the phase calibrator. Any issues can easily be compounded at this stage and lead to a cascade of flagging which leads to a failure of the pipeline. If this part of the pipeline fails, then it is best to go through gaincal by gaincal and check that the number of solutions is sensible and make some minor changes to the default parameters file if necessary.

applycal_all

This step will take all the solution tables and applies them all to the data. Up to this point we have been creating solution tables and applying them “on-the-fly" as we make the next solution table, taking into account the previous ones we have made. This step will take all of those solution tables and apply them to the data into the CORRECTED_DATA column of the ms file, so that when it comes to imaging, we do not have to list lots of tables to apply on the fly. It is often better to do this at this stage as we will want a final calibrated dataset that we can image without all the rest of the calibrators. This step will also run “statwt” which re-weights the data based upon the variance of the data. It is useful to read up on this step in the CASA documentation.

What does it produce and what are the default parameters?

This step only applies the calibration tables to the data, so it only produces a final flag percentage plot for the calibrators in the flag statistics tab. It is worth looking at the flagging and calibration accepted solutions in the logger at this stage.

The calibration tables are then applied to the data using the applycal_all section of this part of the pipeline.

Parameter	Default	Comments
apply_calibrators	["allcal_d.K1", "bpcal.BP2", "allcal_p.G3", "allcal_ap.G3"	Apply these tables to calibrators
apply_targets	["allcal_d.K1", "bpcal.BP2", "phscal_p_scan.G3", "phscal_ap_scan.G3"]	Apply these tables to targets

apply_narrow_calibrators	["allcal_d.K1", "narrow_bpcal.BP2", "allcal_p.G3", "allcal_ap.G3", "narrow_p_offset.G3"]	Apply these tables to the spectral zoom (narrow) spws for calibrators.
apply_narrow_targets	["allcal_d.K1", "narrow_bpcal.BP2", "phscal_p_scan.G3", "phscal_ap_scan.G3", "narrow_p_offset.G3"]	Apply these tables to the spectral zoom (narrow) spws for targets.
run_statwt	true	Decide whether to run statwt or not – this will re-weight the data in a statistical way (see statwt docs)
statwt_timebin	"0.001s"	This is the timebin for statwt. I recommend leaving this as it is unless you know what you are trying to do with statwt as there are work arounds for different CASA versions in the functions of the eMCP to make sure this comes out correctly.

Troubleshooting at this stage

If you have had a lot of flagging, you may end up with a failed apply cal here. You may also want to selectively apply tables but if you do this, I recommend doing this manually rather than running this step. For example, for K band data, you may want to selectively apply different tables for the p and ap solutions from different calibrators to the entire dataset, so it is often easier (and better) to have more control if you are doing this. For standard L and C band though, this will take the tables made previously and apply them to the data as you have done so far on the fly.

A note on statwt: Statwt is the CASA task for re-weighting the data. The CASA webpage notes that this task will re-weight visibilities according to their scatter, which for e-MERLIN should improve weightings on Cambridge and other stable telescopes with good SNR, but down-weight Defford baselines as Defford has poorer sensitivity and more scatter in general. However, this task has gone through some iterations, so it will do different things based on which CASA version you are using and this is in-built into the pipeline, hence the `statwt_timebin` parameter. We are still working out what the best value for this parameter is for CASA 5.8 and above, as it sometimes will cause NaNs to be put into the data so if you get to the imaging step and see a green screen with no flux, then it is worth changing the `statwt_timebin` parameter to something like "4s" or longer, to reduce the likelihood of this happening.

flag_target

This step is as it suggests, flags the target data now that we have applied all the calibration tables to it and will produce another flag versions table as well as a corresponding plot in the flag statistics tab. We do this now instead of earlier as we did not want to overflag data that looks bad due to calibration errors, and/or, we want to make sure that everything is calibrated before searching for low-level flagging points to remove from the target data. Generally, this is a very safe way of doing it, but you may find that there is additional low-level RFI especially in L band datasets that does not get removed from this step as we have not run aoflagger on the data.

As previously with running flag modes, you can pick from tfcrop and rflag using the "mode_to_run" parameter. By default, the pipeline runs rflag, with the parameters the same as those shown previously but we now operate on the

targets on the corrected data column. This part of the pipeline should be ok, as it is just running tfcrop/rflag to remove some bad data.

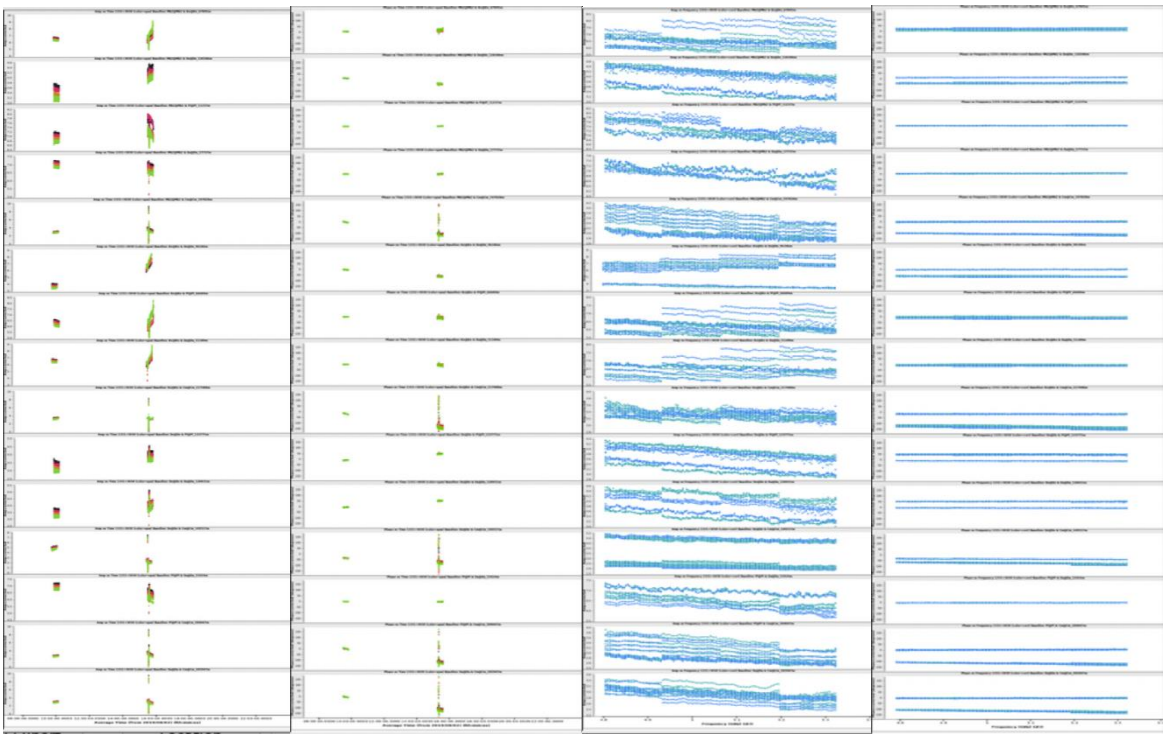
plot_corrected

This step produces a lot of standard plots from the calibrated data, and as such there are no default parameters. In theory this step should not fail either, as it is just making multiple calls to e.g., plotms. The plots it produces are outlined below, with notes on what to look out for:

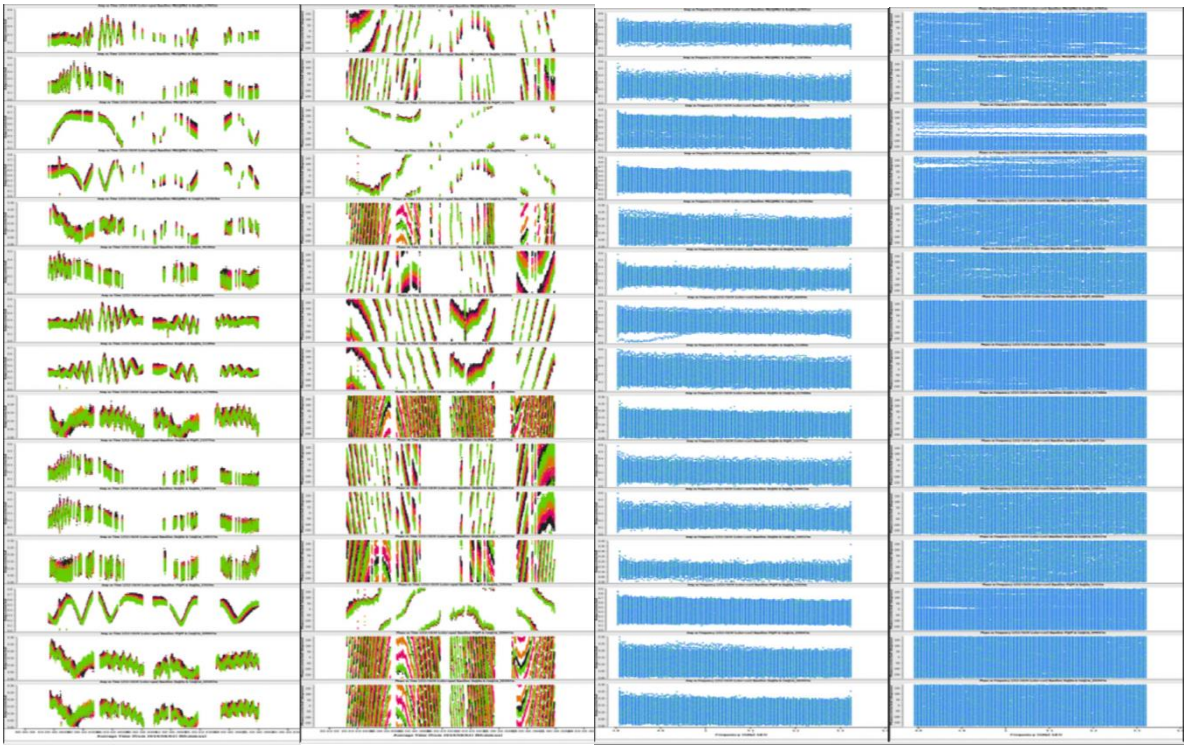
Calibrated visibility plots

The calibrated visibility plots are added to the uncalibrated visibility plots, one per source, with the plots opening in a new tab in your browser window. Below are the plots for 3c286. You can see that each plot has each baseline in it. The four plots from left to right are: amplitude vs time, phase vs time, amplitude vs frequency and phase vs frequency. You are looking for a flat phase for all calibrators except 3c286 in both the phase plots. For the amplitude, your calibrator fields (except 3c286) should also be always the same and you will see the effect of the spectral index should cause the amplitude vs frequency plot to vary across the band. For target scans, if you have a bright source, you should see something similar to the phase calibrator sources in these plots, but if it is slightly resolved then you may see some phase changes at the longest baselines and/or amplitude variation in time as the baseline resolves more (or less) of the source.

In terms of issues, you want to look for excursions in the amplitude and phase of the calibrators. In the example below for 3c286, you can see that the second scan has some issues whereby the phase has some times where it jumps around by 180 degrees. This phase issue affects the amplitude, so we should remove this from the data.



The target plots are below. You cannot see anything in the per frequency plots, but the per time plots show significant amplitude variability and phase variability. This is real source structure and *should not be flagged!*

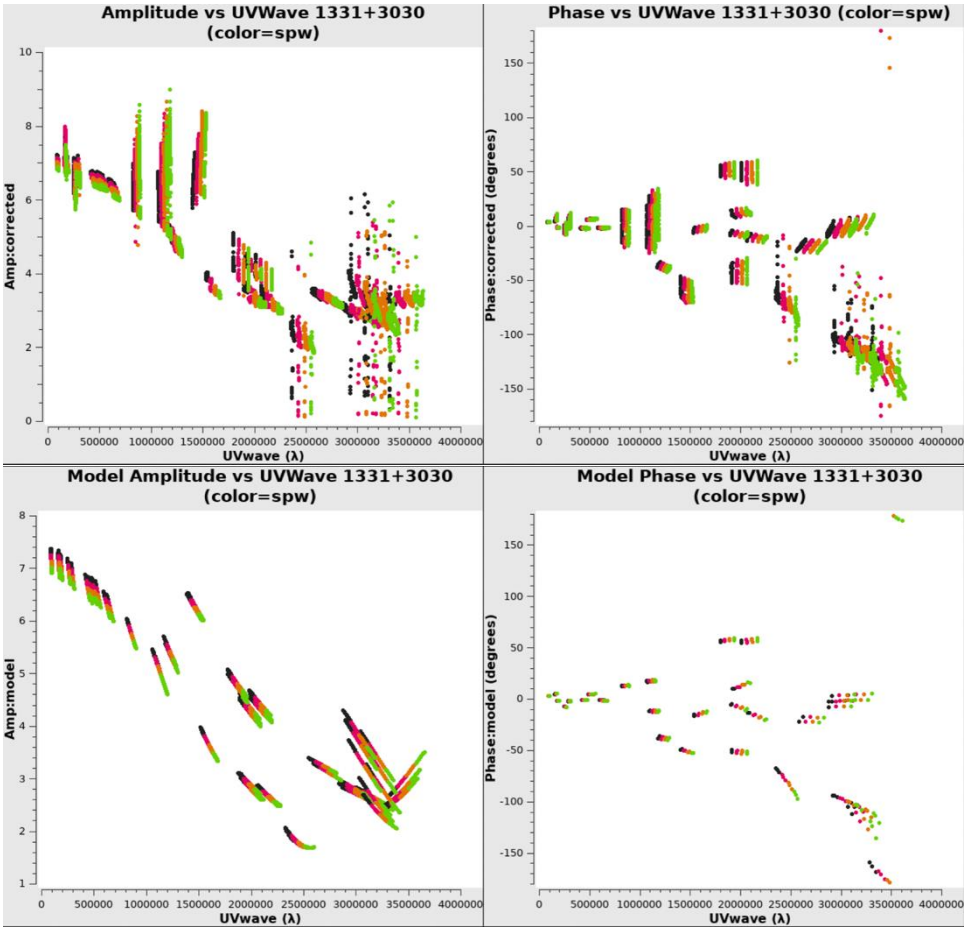


Calibrated uv visibility plots

There are two main types of these plots which show amplitude/phase versus uv distance (I.e., baseline length) – the calibrator plots and the target plots. The calibrator plots will show the model data next to the corrected data plots allowing you to compare what the pipeline thinks the source should look like and what the data are showing. As the models are treated as point sources (or a model image in the case of 3c286), you can see that the phases in the model plots are set to zero and the amplitudes are constant, with the only difference being due to the spectral index we calculated previously. You should be able to compare these to the corrected data plots, and, with a bit of scatter due to noise in the data, see that they align quite nicely. When you see large discursions in phase, or dropouts in amplitude, then you should consider looking at these sources more closely. For the target plots, you do not get a model column as there is no known model for this source, but it is often helpful to understand what a plot may be showing you, which we will go through further below.

In the case of a bright point source, we should see something like the calibrator sources, I.e., a constant amplitude at all uv distances, and a zero phase. If you see the phase start to vary at longer distances, then that tells you that the source is slightly resolved, revealing an extension in one direction or diffuse flux that may not easily be visible in the data without manual cleaning. If you have a source that is variable over the timescales of your observation, then it is possible that you will have a funky uvplot and you should check the calibrated visibility plots if you think this may be the case (see end of this document for examples).

In the plots below I have shown the 3c286 corrected data and model plots. You can see that there is significant scatter at certain points in time in the amplitude and this is also reflected in the phase plot too. We should investigate why this is the case and make flags if necessary. The target plot (not pictured here) shows a few dropouts at the shorter baselines so we should also have a look at that before imaging, but we do not necessarily need to put any flags into the manual_avg.flags file for the target field as the pipeline only operates on the target at the end.



General information for looking at these plots:

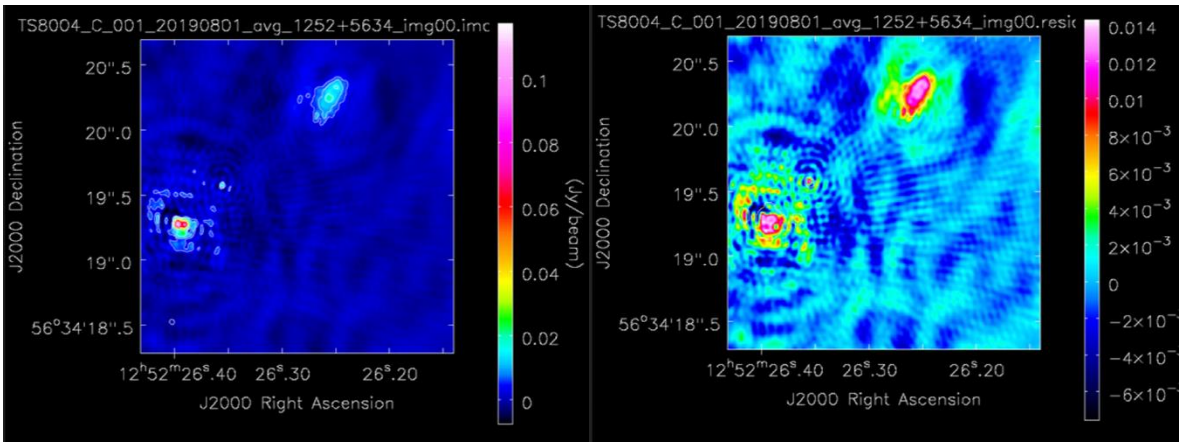
1. Always look at the calibrators *first.*
2. Check that the calibrators corrected data looks like the model
3. Look for areas where there may be amplitude/phase excursions in the corrected data plots
4. Be careful looking at intermediate baselines as this include Defford which is noisier at C band.
5. Flag the main calibrators to remove the bad data – you only need a short amount of time on 3c286/OQ208 for the calibration to work
6. Be careful flagging too heavily on the phase calibrator field
7. Make small flags to the target field only if you are *certain* the data are bad
8. Although tempting, don't clip the data on the target field.

first_images

This step will run tclean in CASA to create the preliminary images of your target and phase calibrator fields, based off the corrected data from the pipeline. You can play around with the imaging parameters, but it is best to leave them as the defaults as you will want to make a manual imaging run for science quality images at a later point. It will also fill out the images page on the weblog with a peak flux from the intensity map and a rms calculated from the residual image. It also produces a zoomed in image so that you can look at any core flux of the source easier. Only the zoom images are included so you can get an idea for the image quality directly.

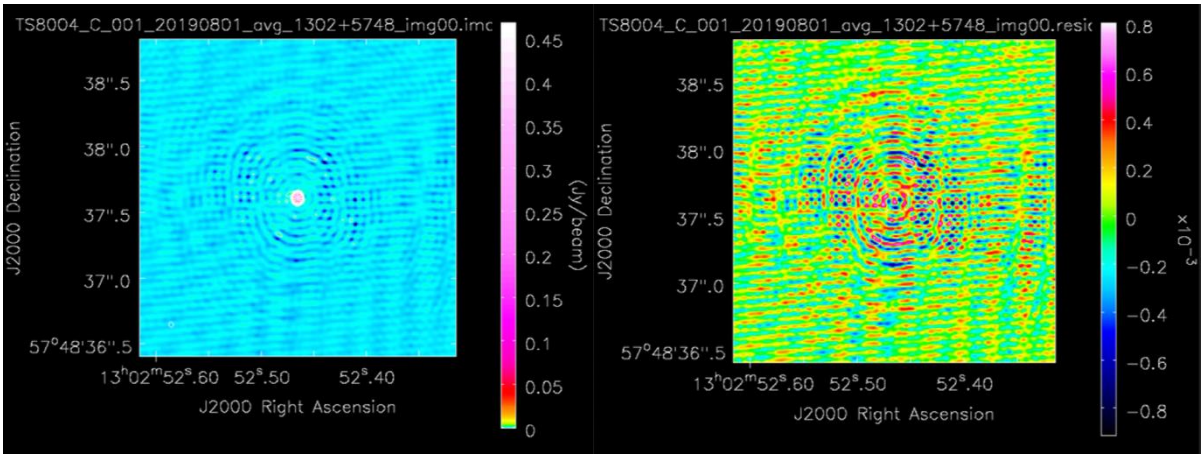
Target images

The zoom image of the target field shows clearly a three-component structure – a central fainter blob, a northwestern diffuse elongated component directed away from the central blob, and a southeastern double component with some small-scale extended emission around it. You can see from the residual map that the imaging has clearly not cleaned out all these components properly and this is due to the



Phase calibrator images

You can see the phase calibrator is a nice point source here, and the residuals are good but not gaussian in noise structure. This is caused by some small-scale residual calibration issues that we should check. The other option you have is to self-calibrate the phase calibrator and apply these to the target field again, which will help remove any issues on the target field especially if it is faint and self-calibrate on the target field is not possible.



Some of the default parameters below are better explained in the CASA guides, and in those cases I have included the [CASA guide description instead.]

What are the default parameters?

Parameter	Default	Comments
imsize	1024	Default image size. A zoom image is also made, so you can keep this slightly larger if you wish, so you can search for extended emission in the wider map, and then delve into the smaller scale emission with the zoom image.
niter	80	Enough clean component to pick out the main bright bits and put them in the model. You will want more than this for manual

		cleaning, but as this is an automatically thresholded imaging for first look images, we do not want to go too high with this.
deconvolver	“hogbom”	Standard hogbom clean algorithm
nterms	1	Only produce an intensity map. You can run this with nterms = 2 but only through CASA, and not through the pipeline, as it will fail.
scales	[]	You can choose imaging scales if you want to search with the automated imager at different scales.
weighting	“briggs”	Use briggs weighting. This is usually the best weighting scheme to use with e-MERLIN due to the heterogeneous array. This goes hand in hand with a robust parameter which is a little off natural weighting.
robust	0.5	Robustness value to choose, we will discuss these later, but 0.5 is usually pretty good
gain	0.1	Set the gain of the imaging, which is set to CASA tclean defaults.
uvrange	“”	You can choose a uv range
uvtaper	[]	You can choose a uv taper (see imaging section)
restoringbeam	[]	You can set a restoring beam, if you want to e.g., circularise the beam.
nsigma	5.0	This sets the auto-deconvolution threshold that tclean will search down to.
sidelobethreshold	1.0	Max threshold based of sidelobe levels: sidelobethreshold * max_sidelobe_level * peak residual. NB this is 1 for e-MERLIN but tclean’s default is 3, so you can increase this to reduce sidelobe structures.
noisethreshold	8.0	This is the masking threshold based on the noise level: noisethreshold * rms + location (=median). The CASA default is 5 here, whereas with e-MERLIN we use 8.
lownoisethreshold	1.5	mask threshold to grow previously masked regions via binary dilation: lownoisethreshold * rms in residual image + location (=median)
minbeamfrac	0.2	minimum beam fraction in size to prune masks smaller than minbeamfrac * beam. This parameter can be useful for low uv coverage (snapshot) datasets, as it will help reduce the area that the automated imager will look for pixels to place in the model. See imaging section below.
growiterations	25	This is a sub-parameter for an auto thresholded map (“auto-multithresh")
parallel	true	Parallelise this process if possible and set up on the machine.
level0	3.0	This is for the contour levels and denotes the 0 level at 3 sigmas of the intensity map.
zoom_range_pix	150	This sets the zoom level for the zoom image

Troubleshooting at this stage

Sometimes you run into CASA issues at this stage with the imager falling over due to errors in the data. We have found that if you are using CASA 5.8, the statwt task in the “applycal_all” step can cause issues by putting NaNs in the data where it cannot find data. Consequently, the imager fails and gives you a green screen. If this happens, either use CASA 5.6 to apply the data again and re-run this step, or remove the statwt from the applycal_all step and run this part again.

split_fields

This will split the corrected data into a separate measurement set, and by default this only does this for the target field using mstransform.

What are the default parameters?

Parameter	Default	Comments
field	"targets"	You can split all fields separately, if you like, but the default is only to split the target fields
timeaverage	true	We average further in time to make the dataset smaller.
timebin	"8s"	Time averaging, so we halve the time resolution of the avg.ms data by using an 8s average
chanaverage	true	We average further in frequency to make the dataset smaller.
chanbin	2	Channel averaging by a factor of 2, halving the data in this axis.
datacolumn	"corrected"	Only split the corrected data – this will mean that your corrected data column becomes the data column in the split ms file
createmms	false	Whether to create a mms file or not, default is not to.
output_dir	"./splits"	Put it in a separate directory.

Re-running the pipeline

Inspecting the data and making a manual_avg.flags file

Once you have your weblogs sent to you, quite often the first thing you may want to do is explore the data and the weblogs to see if the calibration has proceeded correctly or not. While the steps above have highlighted what each step does and offered some help in how to fix it if they fail, they do not show you what to do if you have some bad data that needs excising from the data still, or how to find it. This section will go through what to look for in the weblogs and data to find bad data, and how to write a flagging file to remove the bad data before the calibration procedures start.

Flagging the data

Once you have found the areas of bad data, you can now put these into a flagging file. The eMCP has a straightforward way of including your own flagging information into the pipeline via a the "manual_avg.flags" file that will be ingested during the start of the calibration part of the pipeline as part of the "flag_manual_avg" step. Similarly, if you have flag commands that you want to specify for narrow band spectral line data, you can do this with the "manual_narrow.flags" file which is ingested at the same point. For the purposes of this demonstration, we will only write commands into the manual_avg.flags file for the continuum data.

The syntax of these files abides by what is needed for CASA for the casa task flagdata. The main rules are:

1. Use only ONE white space to separate the parameters (no commas). Each key should only appear once on a given command line/string.
2. There is an implicit mode for each command, with the default being 'manual' if not given.
3. 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.

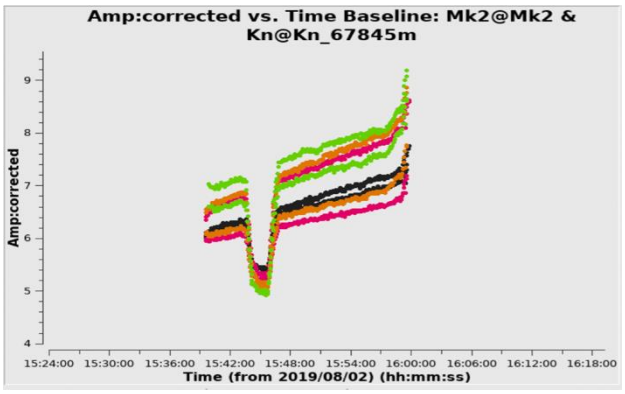
We now go through and inspect the data to find the areas that should be flagged in this dataset

Inspecting the data

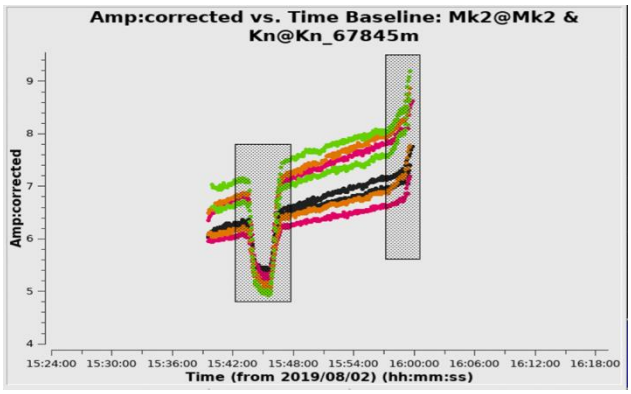
To inspect the data, you should look at the calibration plots first. We noted that there were issues on all the calibrators, specifically 3c286 where there appeared to be a significant phase and amplitude excursion during the second flux cal scan. To do this, we use plotms in CASA. Either load up CASA and type "plotms", or run `/path/to/casa/casaplotms &` which will load CASA's plotms in a separate window but keep you on the command line. Below we will outline the basics of what to plot up to look for in e-MERLIN data, but a full explanation of the plotms gui can be found in the online CASA documentation.

To look through the data, we will start with 3c286, so select that in the "field" part of the gui. Load in the file with "_avg.ms" with the "Browse" button. Then we want to only show the LL, RR correlations as we have not calibrated the cross polarisations, I.e., corr should be set to "ll,rr". We are going to start off plotting by time, so we want to average by channel – each spw has 128 channels, so w average by 128 channels. On the axes tab, choose "Time" on the x axis and "Amp" "corrected" in the Data and corrected axes, respectively. On the page tab we will plot via Baseline. Finally, under the display tab, we will choose to "colourize" by "spw". This will give you broadly similar plots to the pipeline, but the gui will enable you to pan and zoom through the data on a per baseline basis, hunting for areas of bad data.

For example, plotting amplitude vs time for the first baseline in the dataset, we see the following:



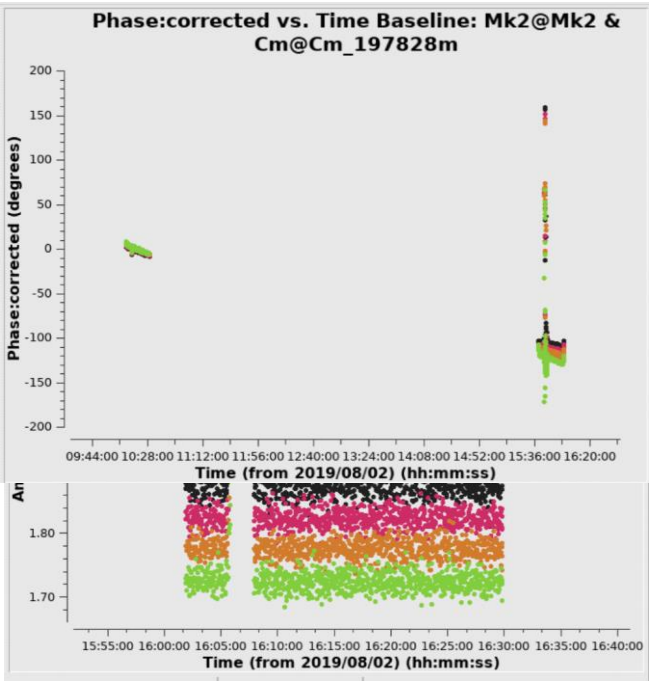
The plot on the right shows the section we should flag from the data.



If we flick through the other baselines we see a similar issue, so we want to note the following flag commands to put into the manual_avg.flags file:

```
mode='manual' timerange='2019/08/02/15:43:00~2019/08/02/15:47:00'
mode='manual' timerange='2019/08/02/15:57:00~2019/08/02/16:00:00'
```

If you also look at the phase vs time plot, you will see that there is a phase excursion at similar times, so we are certain that this is bad data. E.g. see the plot on the right.



We can now move onto 1407+2827, the band pass calibrator. There is only a small amplitude spike in the 1407+2827 data, which looks like the pipeline has flagged some bad data nearby but has not flagged all the bad data:

This image is for the Mk2&Da baseline, and if we flick through the baselines we see that this spike is only prevalent on the Darnhall baselines. You can therefore choose to flag only Da baselines for this small piece of bad data using `antenna='Da'` but as it is only a 20s piece of bad data, then we could also flag all baselines as it will do no harm. Add to the manual_avg.flags file:

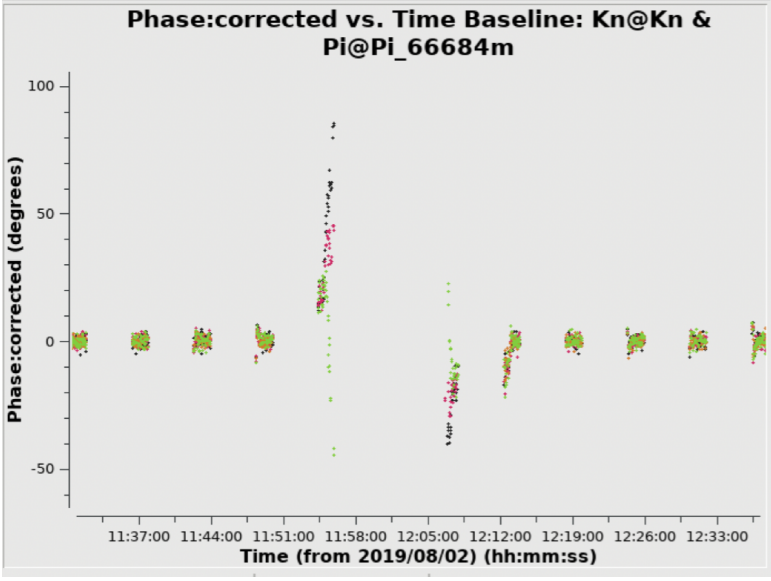
```
mode='manual' timerange='2019/08/02/16:05:00~2019/08/02/16:06:00' antenna='Da'
```

If we check the phase plots however, we do not see any issues in this section of time. Instead, we see a few excursions from zero phase on several baselines at the start of the OQ208 scan. It is likely that the telescopes were not on source at this point and settling, so we may expect to see some amplitude variation too thought it is not clear. I would recommend flagging the first part of these scans that we see the phase discursions on, just to be on the safe side – remember, so long as we have at least 10 mins of good data on these calibrators then the pipeline should have enough to calibrate.

```
mode='manual' timerange='2019/08/02/21:41:00~2019/08/02/21:43:00'
mode='manual' timerange='2019/08/02/16:01:00~2019/08/02/16:02:00'
```

We now want to do the same for 3c84 (0319+4130), but it is not necessary to do so unless you are wanting to do polarisation work afterwards. Look through and decide on flags for 0319+4130 yourself and see if you can remove the worst of the data.

The main final part is to look at the phase calibrator and target. Remember that you can flag the target later, so it is not imperative that you flag the target here, it is more important to get the phase calibrator flagged appropriately. Judging by the uv plots, the phase calibrator is quite stable but has some phase excursions. You do not need to worry about the small ones, only the big ones, as you can phase self-calibrate both the phase calibrator and target later to get rid of these issues. However, there is clearly a period on the Kn baselines which need looking at:



This significant excursion in phase should be flagged for all Kn baselines. Looking at other baselines, it affects Pi&Da too, so lets flag it for all baselines.

```
mode='manual' timerange='2019/08/02/11:50:00~2019/08/02/12:15:00'
```

You should now look at the other slight excursions from the zero phase. Most of these are small and will not make a significant difference to the final calibration, but if you are using this pipeline to make final calibrated datasets then you should really go through and flag all the small regions where we have phase discrepancies on the phase calibrator. Some of these are noted below, but have a look for yourself and check:

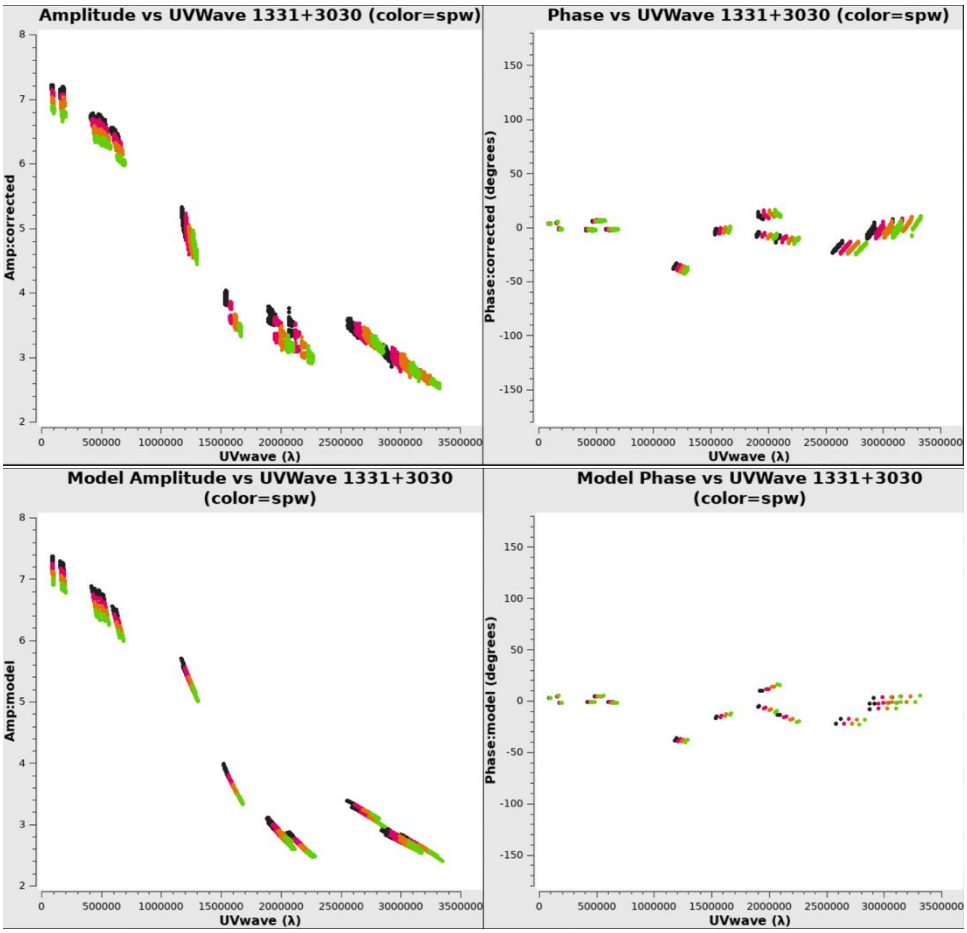
```
mode='manual' timerange='2019/08/02/01:13:20~2019/08/02/01:14:00' mode='manual'
timerange='2019/08/02/06:13:52~2019/08/02/06:14:00'
mode='manual' timerange='2019/08/02/06:24:20~2019/08/02/06:25:00'
mode='manual' timerange='2019/08/02/14:48:20~2019/08/02/14:48:40'
mode='manual' timerange='2019/08/02/14:54:20~2019/08/02/14:54:40'
mode='manual' timerange='2019/08/02/14:55:00~2019/08/02/14:56:00'
mode='manual' timerange='2019/08/02/17:05:00~2019/08/02/17:15:00'
mode='manual' timerange='2019/08/02/18:24:20~2019/08/02/18:25:00'
mode='manual' timerange='2019/08/02/20:25:00~2019/08/02/20:26:00'
```

Notice that these flags wrap over some target scans. This is fine, as if we cannot use the phase calibrator either side of the target scan, then it is likely that there are issues with the target too. You could set the flags to only flag on the field ID, I.e. by adding `field='1302+5748'` to the end of every line, but the pipeline will not be able to find any solutions on these phase calibrator scans to transfer to the target fields, and therefore will flag the target fields as part of the applycal_all step anyway.

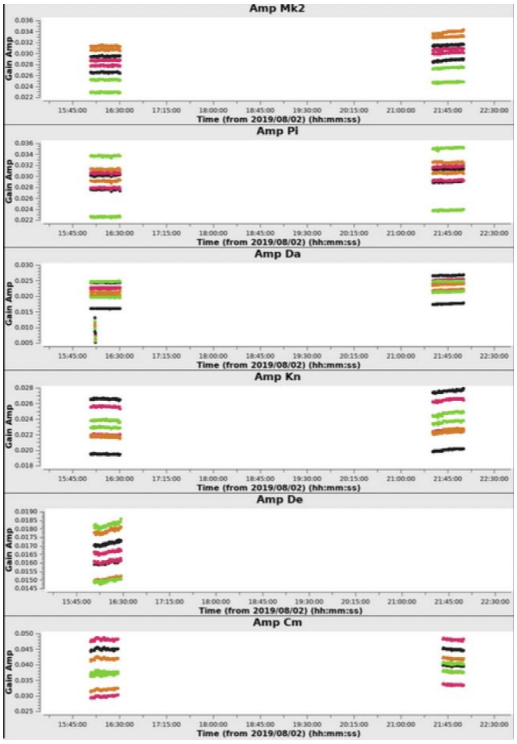
The data can now be rerun with all calibration turned on as the manual_avg.flags file will need to be read in at the start and calibration re-started. E.g.:

```
/path/to/casa -c eMERLIN_CASA_pipeline/eMERLIN_CASA_pipeline.py -r calibration
```

This will take a while to run so make a cup of tea! When you get back you want to go through the same process again and make any small flags and add them to the manual_avg.flags file, inspecting all the plots in the weblog as you go. For reference on this data set, the calibration process from end-to-end takes about 2 hours. Now, let us look at the plots we end up with from the pipeline. First look at 3c286 and OQ208 calibrated plots:

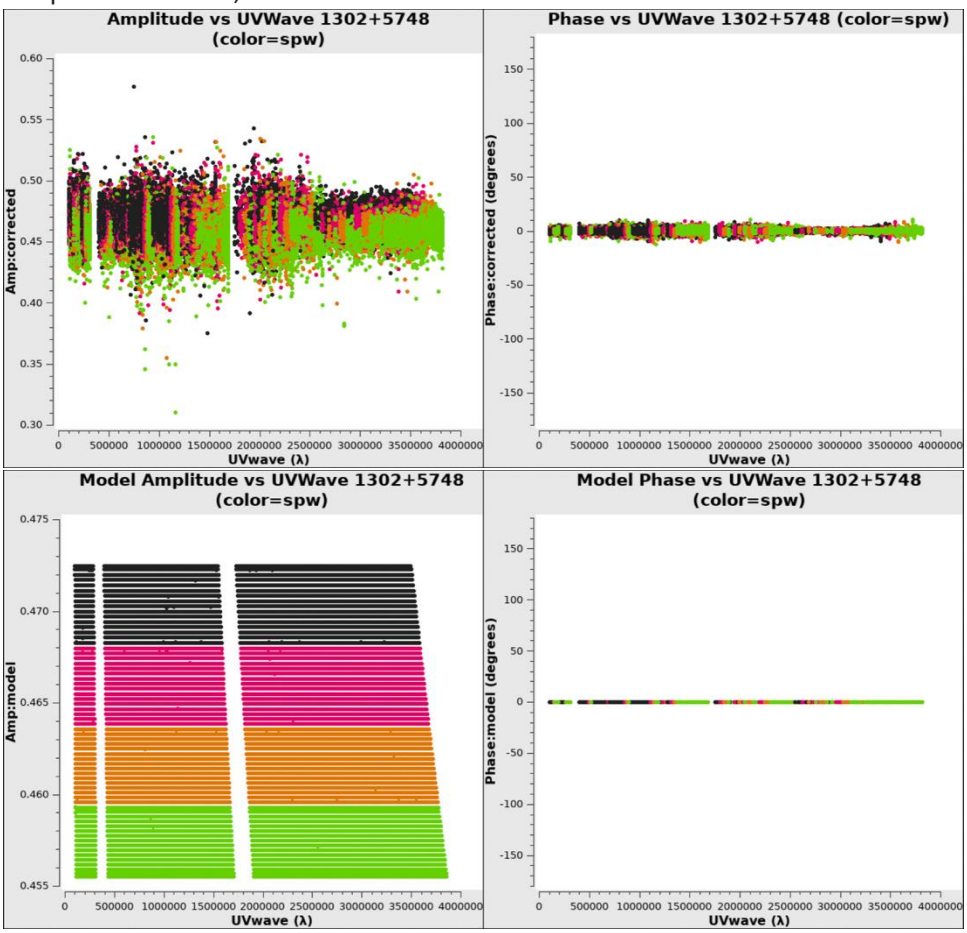


3c286 looks excellent! The data looks identical to the model. The band pass calibrator looks good too. What about the calibration plots though? There appears to still be a small drop in the calibration table bpcal_ap.G0 for Darnhall antennas:



... but this does not seem to have affected the uv visibility plots and if we look at the amplitude vs time of the final calibrated visibilities for OQ208 we can see that this section has been flagged, so we do not need to do anything else – the pipeline has taken care of this bad piece of data and removed it.

Now we look at the phase calibrator, although note that if you wanted to do polarisation analysis, you should check the point calibrator, 3c84 too.



The amplitude is flat with some scatter around 0.48Jy, with a flat phase. This looks like a great calibrator and an impressive set of solutions that have been applied to the data. If you wanted to you could self-calibrate this source and re-apply the solutions to the target, but it is not necessary. The bigger issue will be flagging the target field now, and self-calibrating that. At this stage, I would start looking at my target field with the aim of imaging it, and we will go through that in a later section.

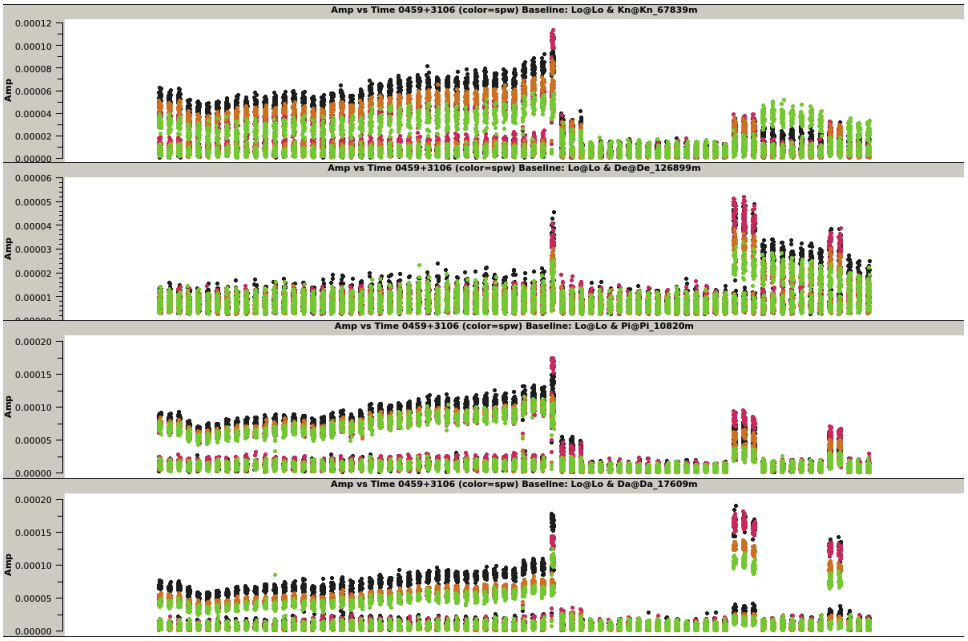
Other things to be looking for in logs

While the 3C277.1 data is useful, it does not show all the possible issues that you can run into with e-MERLIN Data.

Delay jumping

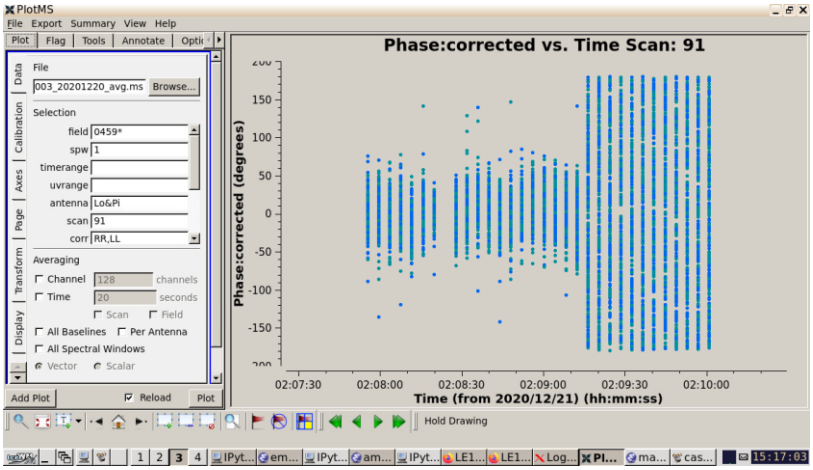
Delay jumps can occur in the e-MERLIN correlator when there is an issue with the digital installation on the telescope, possibly due to hot weather which causes more problems with the electronics. This causes the signal to be delayed and causes the delay to “jump” when received in the correlator. These things happen rarely, but usually if it is an issue in an observation then it may happen several times in an observation. Another reason for this happening can be electrical surges at the telescope sites which causes the signal to lose lock. When the lock is regained, it is at a slightly different delay length than previously. This is all in theory calibratable by the eMCP, as long as there is no rate change – so long as the delay jumps from one value to another and stays there for a period of time, then we can recover the signals. But what does a delay jump look like? Below are some images of data from another dataset which shows evidence for delay jumping. First, looking at the uncalibrated amplitude vs time plots for the phase calibrator, you can see that the amplitude jumps around a fair bit on the Lovell baselines (note that

this is not the case on the other baselines in this dataset), suggesting that the Lovell has a specific issue. Note also that the pipeline flags the Lovell baseline in this dataset, presumably due to this issue, so it is important that we save the data.

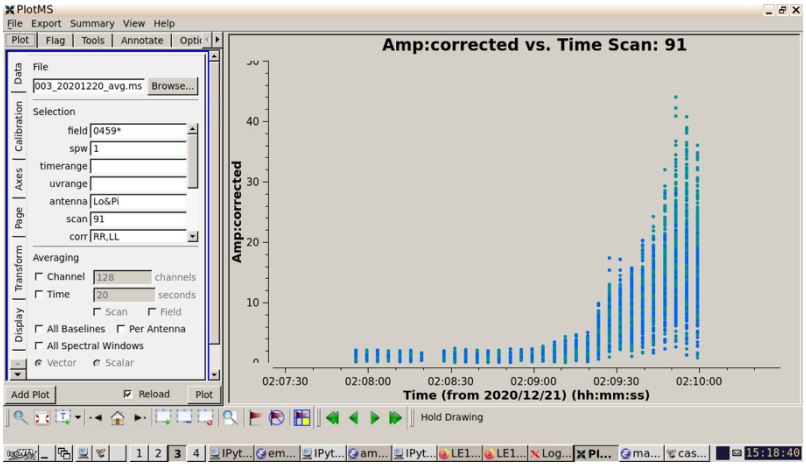


Once we re-run the pipeline and make some manual changes to the pipeline to include Darnhall as the reference antenna, rather than Cambridge which is what was automatically picked by the eMCP. However, to see how this is affecting the data, we look at a specific scan (scan 91) on the phase calibrator:

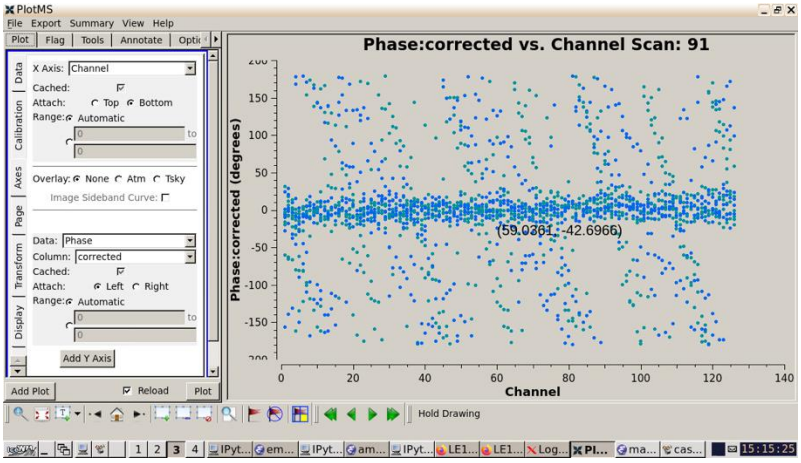
You can see the calibrated phase looks fine, but then it has very noisy phase solutions at a certain point:



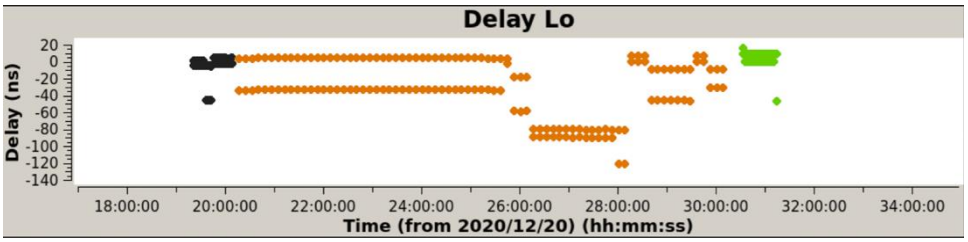
Then checking the corrected amplitude, we see that that jumps too:



And then if we plot the phase per channel, we see that there are two blue and two green lines, one which has calibrated out to zero and one which has calibrated out to being flat bur wrapping around, suggesting we have two delay solutions in this scan.

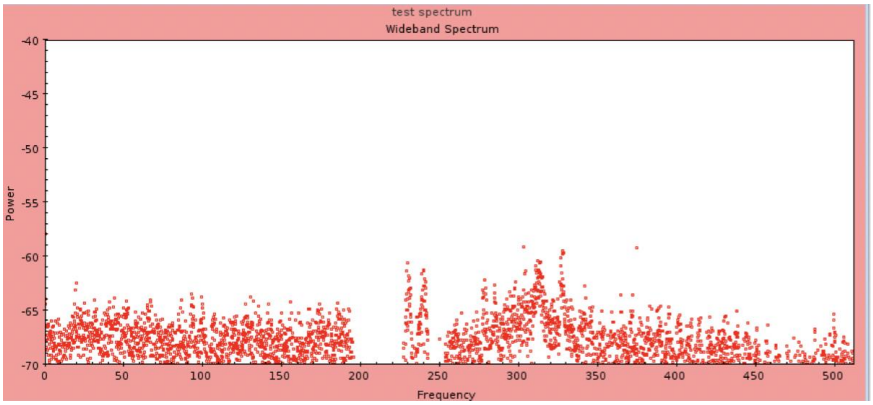


So how do we solve for this problem? We can't change the fact that there has been a jump, but the pipeline solutions will take this into account as seen above, but there will be an offset in the final calibrated dataset, which you will see in the form of a allcal_d.K1 solution table which will have multiple horizontal lines, with each one corresponding to different solves.

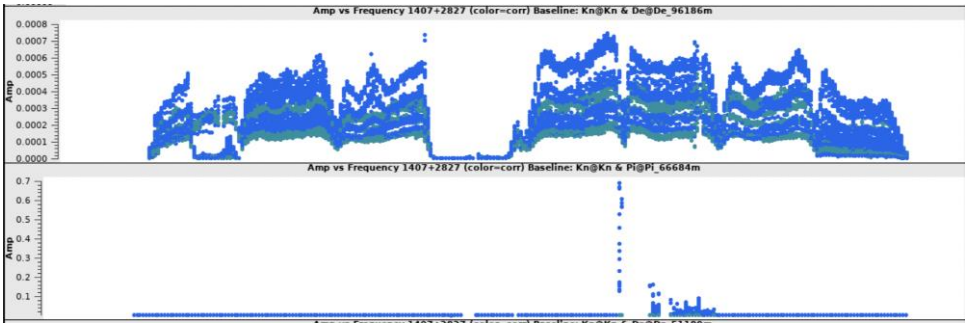


So this is fine for calibration, and will save the Lovell data in this particular dataset. If you find that there are multiple jumps in a short time frame, then you could try reducing the delay calibration solint, as this will force the pipeline to solve on a shorter timescale, and thus find a solution for every individual phase jump. So if you are finding delay jumps on times of ~1 minute, then perhaps set the delay solint to 20s, so long as you have enough SNR to get a good solution.

Spw 3 in L band data + Correlator chip errors



Since the COVID 19 pandemic, we have noticed an increase in RFI in the L band, mostly caused by 4G+ emissions. These emissions live in the region where spw 3 is at L band in e-MERLIN data. Consequently, we have had to put in a filter to remove this region of the band at the telescope stage, so it does not cause saturation of the antennas across the entire band. The pipeline *should* therefore not see any emission in this part of the band; however, we have seen some instances where RFI breaks through the filter and is bright enough to be considered "real data" but the eMCP. In this case, if the RFI in this part of the band is bright enough, then it may take over the pipeline, with the eMCP flagging the good data, and leaving only spw 3. You can see an example below from our internal monitoring diagrams, showing the two lines breaking through the filter halfway up the band. If this affects your data, then it is best to flag spw 3 in your data set if it is L band. In general, we now flag this spw anyway to be on the safe side.



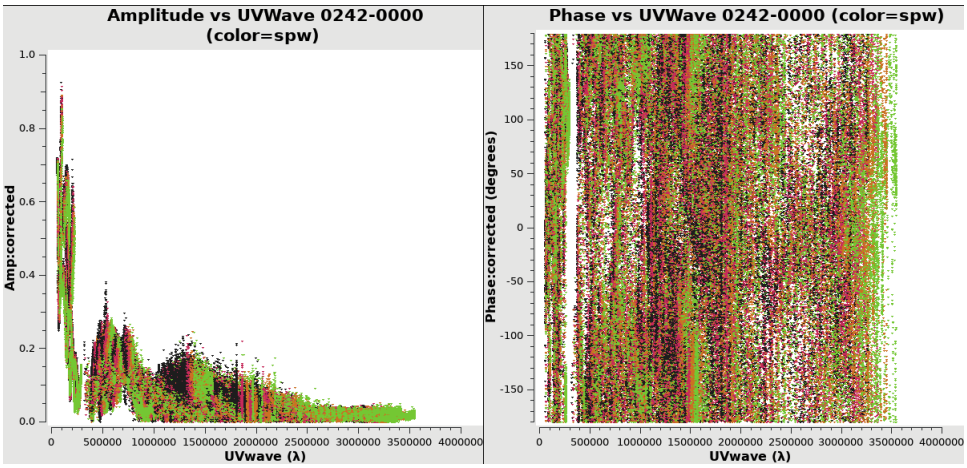
Another example can be seen in the weblogs for L band data. Below is an example of an *uncalibrated* uv plot of amplitude plotted against frequency at L band for the band pass calibrator. It shows a clear band structure for the top (Kn&De) baseline as we have seen before and calibrate for. However, you can see that spw 3 is effectively gone, but not entirely, so it is important to flag all of spw 3 so spurious edge channels do not get calibrated without the rest of the spw to help to pin down the band pass. You can also see a different issue on the Kn&Pi baseline which we discuss in the next section.

As shown in the previous plot, the Kn&Pi baseline has a clear issue that is different to just spw 3 being unavailable in the L band data. Due to the ageing analogue correlator, we sometimes have errors on the chips on the boards in the correlator. We move these chips around to try and reduce the number of failed chips as these have a direct effect on the correlated data. In this case, we lose all of spw 5 on the Kn&Pi baseline. This is unrecoverable and should be flagged. The pipeline usually does a good job of finding these, but not always – sometimes it believes that spw 5 (in this example) is the real signal from the telescope and the other spws are zero signal. Note that the y axis is from 0 – 0.7, whereas the Kn&De baseline had an uncalibrated amplitude of 0 – 0.0007, so there is clearly a difference here that is not related to the source and purely an instrumental problem. If you see these issues in the data then it is best to flag these in the manual_avg.flags file to get rid of them before the calibration procedures begin.

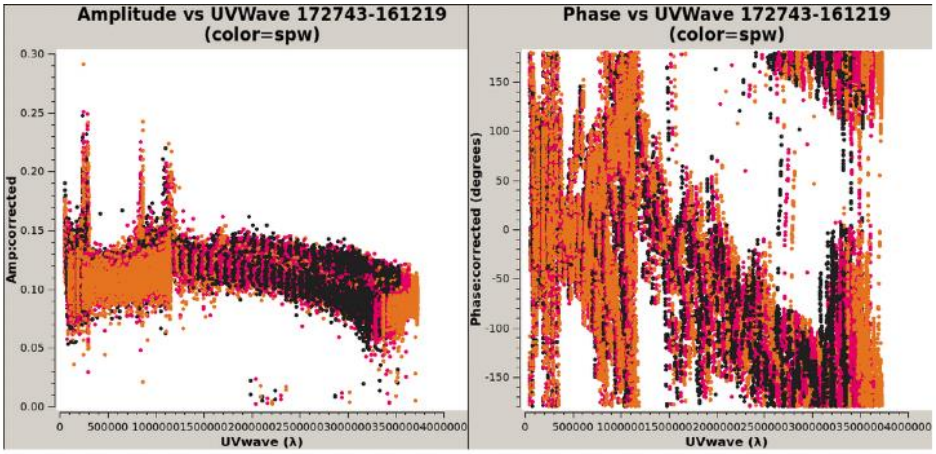
Examples of odd calibrated uv plots

Here, we go through some plots of datasets taken with e-MERLIN that show interesting target uv structures straight out of the pipeline. These are all target sources so you in principle should not see this on your phase calibrator field

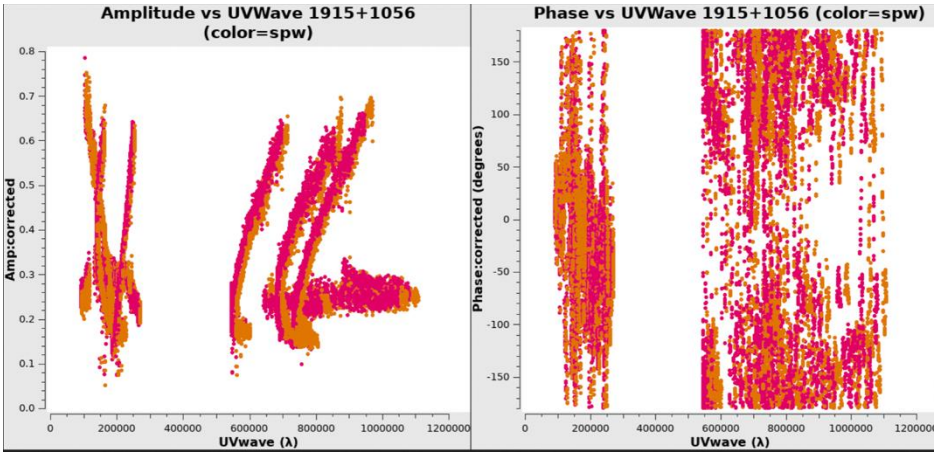
as we try to make sure that they are unresolved and bright, although this is not always possible in all situations. If you see something like these examples on your dataset, then it is best to look at the data very closely to make sure you do not over flag the target field and remove some of the interesting science. Due to e-MERLIN's combination of short (~10km) and long (~200km) baselines, it is normal to see extended structures and so this should be considered before flagging the target field. For example, the plot below is of a source with multiple bright unresolved components and large diffuse flux that e-MERLIN is not sensitive to. The phases look very messy, but the variable amplitude information suggests that there is structure on a variety of different baselines.



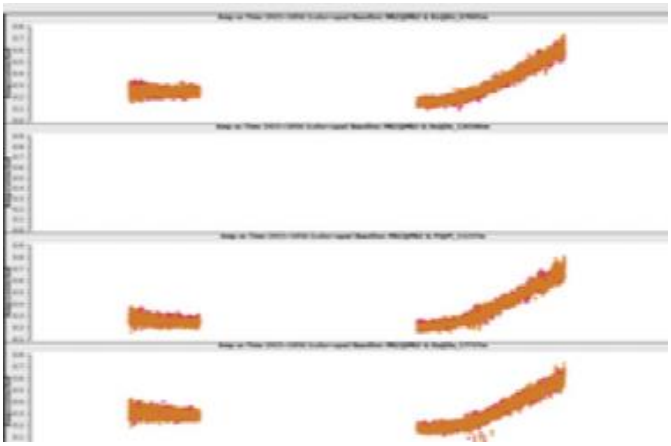
The plot below is of an X-ray binary source, which as a resolved structure which is approximately the size of the e-MERLIN beam, so we see a flat calibrated amplitude up to a distance of 100km, but on the longest baselines the amplitude drops and splits, suggesting multiple components or a slightly resolved structure, which you may see in the e-MERLIN images.



The next odd calibrated uv plot is when you have a variable source during the observation. What you may find here is that the target source shows one calibrated amplitude across all uv distances, and then an increase at all uv distances at a later time. You can only see this in the amplitude vs time plots, but it may look like the following amplitude vs UVwave plot below:



And to show that it was variable, you can see in the amplitude vs time plot below that in the second half of the observation the source started brightening again.



Additional methods of running the pipeline

Running the pipeline with manual intervention

The pipeline is designed to enable most users to reduce most e-MERLIN datasets in a satisfactory manner. However, this is not always the case. Sometimes there are issues with individual correlations or additional requirements of the pipeline are needed. For example, if you have a resolved phase calibrator, you may want to phase and ap self-calibrate this source first before applying it to the target field. This can also be a problem if you have a field with a secondary bright source which needs cleaning and including in the model.

Another example which is common with e-MERLIN is that the pipeline was written before some newer parameters were incorporated into gaincal. One such parameter is “corrdepflags” which will by default not use data for solutions if any data on that time stamp is flagged on another correlation. Every so often, the e-MERLIN correlator has an issue with one polarisation, I.e., LL, but the RR polarisation is ok. In these cases, you would want to keep the RR polarisation which is fine. To do this with the pipeline is not currently possible, as it requires changing “corrdepflags = True”. One way of potentially doing this in the pipeline would be to edit the eMCP_functions.py python script that is run as part of the pipeline and can be found in the functions sub-directory of the eMERLIN_CASA_Pipeline folder. You would have to edit all the gaincal commands in this script to include this parameter.

Phase shift

In some cases, a source may be localised to a region of sky that is not as precise as the e-MERLIN resolution, I.e. a several arcsec localisation of a source. Or, in some cases you may find that there is an "in-beam" calibrator (see self-calibration section) that you want to use for self-calibration purposes and add it into the ms as an additional source for calibration. In this case, you must use the "shift_phasecenter" parameter in the average section of the default_params.json file. Whilst you can use this parameter to move a field pointing center, there are several things you should be aware of first, which usually means it is better to make several calls to the pipeline to properly move the field.

First of all, we should go through how the shift_phasecenter parameter works in the eMCP. If turned to true, it will read in a text file, called "shift_phasecenter.txt", which should be a list of fields and where you want to move them to. It will run the CASA task mstransform to do this transformation. The basic structure is as below

```
MStargetname, inbeam1, J2000 HHhMMmSS.s DDdMMmSS.s
MStargetname, inbeam2, J2000 HHhMMmSS.s DDdMMmSS.s
MStargetname, inbeam3, J2000 HHhMMmSS.s DDdMMmSS.s
```

So in the case of the 3c277.1 dataset, if there was a source directly north of the target by 2', then we would write the following:

```
1252+5634, inbeam1, J2000 12h52m26.2859s 56d36m19.488s
```

This will move the target field and rename it as inbeam1, but as "inbeam1" is not in the inputs.ini file, it will not be included in the rest of the pipeline steps. Hence, it is important to make two inputs.ini files, one to do the steps up to and including the averaging step which will do the position transformation, and one inputs.ini file to do the rest of the pipeline with the inbeam calibrator included. So, the first inputs0.ini file should look like (for this demonstration file):

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

fits_path = /path/to/DATA/
inbase    = TS8004_C_001_20190801
targets   = 1252+5634
phscals   = 1302+5748
fluxcal   = 1331+3030
bpcal     = 1407+2827
ptcal     = 0319+4130
```

And you run the pipeline with the following command:

```
/path/to/casa -c ./eMERLIN_CASA_pipeline/eMERLIN_CASA_pipeline.py --run-steps
pre_processing -s plot_data save_flags -i inputs0.ini
```

Once this has run successfully, you can then run a second inputs file, here named inputs1.ini, as below:

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

fits_path = /path/to/DATA/
inbase    = TS8004_C_001_20190801
targets   = 1252+5634,inbeam1
```

```
phscals   = 1302+5748,1302+5748
fluxcal   = 1331+3030
bpcal     = 1407+2827,0319+4130
ptcal     = 0319+4130
```

And you can run this with

```
/path/to/casa -c ./eMERLIN_CASA_pipeline/eMERLIN_CASA_pipeline.py --run-steps
plot_data save_flags calibration -i inputs1.ini
```

Which will include both fields in the final calibrated dataset.

A couple of things to be aware of when doing this two-level approach to shifting the phase center. First, is that to do this you will need the raw, uncalibrated data and run the pipeline from the start. If you only have an avg.ms file from your support scientist then you should ask them for the raw uncalibrated fits files so you can run this part of the pipeline manually yourself. Second, you should turn off all of the averaging when you run the inputs0.ini call to the pipeline in the importfits step as you want to make sure you avoid any bandwidth and time smearing in the data by using the unaveraged data. You can then use the standard averaging in all other steps, including the "average" step in the inputs0.ini part of the pipeline, as the phase center shifting will occur *before* the data are averaged down.

K band

K band observations with e-MERLIN are not as common as L and C band observations as the weather in the UK is not quite as good, and there are only 5 antennas that are K band capable with e-MERLIN. Another complication is the lack of high frequency bright calibrators to do calibration with a single phase calibrator. Instead, a two-level phase calibration strategy is employed, whereby a further away bright source is used as an amplitude calibrator and a nearby fainter calibrator is used as a phase calibrator. However, it is not straightforward to do in the pipeline. First of all, it is better to use 3c84 (0319+4130) as the band pass, flux and point calibrator as OQ208 is too faint for K band and we do not have a 3c286 flux calibration model.

In order to pipeline K band data with this two-level calibration strategy, first the inputs.ini file needs to be changed:

```
targets   = near cal,target,target
phscals   = far cal,near cal,far cal
fluxcal   = 0319+4130
bpcal     = 0319+4130
ptcal     = 0319+4130
```

This enables the pipeline to read in both calibrators to attach to the target, but also for the far cal gains to be applied to the phase calibrator too. Next, specific lines in the default parameters file need to be changed:

```
4     "refantmode"             : "strict", 5     "refant"                 :
"Kn",
80    "init_models": { 81    "calibrator_models"      : "calibrator_models/",
82    "manual_fluxcal"         : true, 83    "fluxcal_flux"           :
[25.3473,0,0,0], 84    "fluxcal_spix"           : 0.5684, 85
"fluxcal_reffreq"        : "24.100GHz", 86    "wtmode"                 : "nyq",
87    "dowtsp"                 : false 156   "delay_cal"              :
"far_cal", 168   "p_solint"              : "40s", 247   "ap_calibrator"
: "far_cal,near_cal", 168   "p_solint"              : "40s", 255
"ap_calibrator"          : "far_cal,near_cal",
```

Make sure that the refantmode is strict and your best refant is chosen, in this case Kn. We want to do a manual flux calibration, so you can see above how one would do a manual flux calibration for other non-K-band programmes

(lines 80-87). The flux value above comes from Medicina monitoring of 3c84. Finally, you need to use your far calibrator to do the delay solutions and your ap solutions, include your near calibrator to your ap calibrator list, and increase the solint for phase up to 40s to increase the SNR. The solint needs to be more than "int" and you may find that you need to increase the other solints appropriately too. Adding the near calibrator to the ap_calibrator parameter is not necessary, but I have found helps preserve solutions, but that may be because the nearer calibrator in the test data I used was 0.1Jy. Optionally, you can set minsnr to 1 and minblperant to 2 and these may help stop data being flagged at early stages. Similarly, calflag may be used instead of calflagstrict and you can turn off all additional flagging stages too.

Running the pipeline

To run the pipeline, you should run all pre_processing steps and all calibration steps up to *but not including* the applycal_all step:

```
/path/to/casa -c eMERLIN_CASA_pipeline/eMERLIN_CASA_pipeline.py -r
pre_processing calibration -s applycal_all flag_target plot_corrected
first_images split_fields
```

Check that the data looks ok first before then doing applycal manually like so:

```
applycal(vis="./project_avg.ms",field="far_cal,0319+4130,near_cal",spw="",inten
t
="",selectdata=True,timerange="",uvrange="",antenna="",scan="",observation="",m
sselect="",docallib
=False,callib="",gaintable=['./weblog/calib/project_allcal_d.K1',
'./weblog/calib/project_bpcal.BP2', './weblog/calib/project_allcal_p.G3',
'./weblog/calib/project_allcal_ap.G3'],gainfield=['nearest', '0319+4130',
'nearest', 'nearest' ],interp=['linear', 'nearest,cubicflag', 'linear',
'linear'],spwmap=[[0, 0, 0, 0], [], [], []],cal
wt=[True],parang=False,applymode="calflagstrict",flagbackup=False)
```

```
applycal(vis="./project_avg.ms",field="near_cal",spw="",intent="",selectdata=Tr
ue,timer
ange="",uvrange="",antenna="",scan="",observation="",msselect="",docallib=False
,callib="",gaintabl e=['./weblog/calib/project_allcal_d.K1',
'./weblog/calib/project_bpcal.BP2', ' ./weblog/calib/project_phscal_p_scan.G3',
'./weblog/calib/project_phscal_ap_sc an.G3'],gainfield=['far_cal', '0319+4130',
'near_cal', 'near_cal'],interp=['linear', 'nearest, cubicflag', 'linear',
'linear'],spwmap=[[0, 0, 0, 0], [], [], []],calwt=[True],parang=False,applym
ode="calflagstrict",flagbackup=False)
```

In essence the first applycal will work on the phase calibrator, whereas the second one will work on the target field. Note that the second applycal will use the per-scan solution tables and apply information from the phase calibrator to the target field. The phase calibrator has had the information from the nearest amplitude calibrator in the previous step applied to it. It is often useful to make a preliminary run with applymode='trial' to see how this affects the data and flagging percentage first. You can optionally run statwt manually, but as mentioned previously, be careful running this in CASA 5.8 as it has a habit of introducing NaNs into the data. You can now run the rest of the pipeline:

```
/path/to/casa -c eMERLIN_CASA_pipeline/eMERLIN_CASA_pipeline.py -r flag_target
plot_corrected first_images split_fields
```

You can then go ahead and treat the K band data as any other dataset and image/self-calibrate it as you wish.

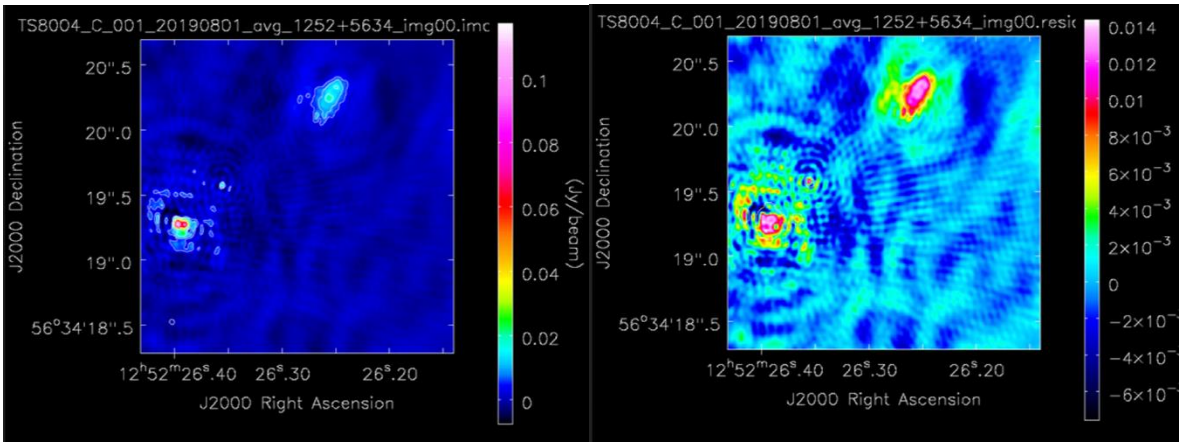
Imaging + Self Calibration

The e-MERLIN pipeline will perform basic imaging of your data in an automated way, but these use a niter of 80 and not be publication quality. This is because you can improve the data significantly by doing a manual cleaning with more iterations to better model the source. You can also change the robustness parameters of the data to highlight specific aspects of the data such as large-scale diffuse structures or point-sources. For imaging, the best tutorial to follow is the one online for ERIS 2022: https://www.jb.man.ac.uk/DARA/ERIS22/imaging.html

The above tutorial uses the same e-MERLIN data set providing parameters for manually cleaning data and improving the imaging fidelity. During the e-MERLIN Data school, we will go over a live demonstration of this to improve the data with self-calibration also. This section is designed to provide some information on changing the default parameters for the eMCP to make better or different pipelined images and provide some information on cleaning outside of CASA.

Examples of changes in auto imaging parameters for the pipeline and when to use them

Below is the image we made in the first run of the pipeline which used the pipeline default parameters and without any additional flagging or manual intervention than what is already in the pipeline. This section will remake these images by only changing the parameters in the pipeline to automatically make images, which may be useful for different purposes.



Example 1: Using CASA default auto thresholding parameters as opposed to e-MERLIN ones

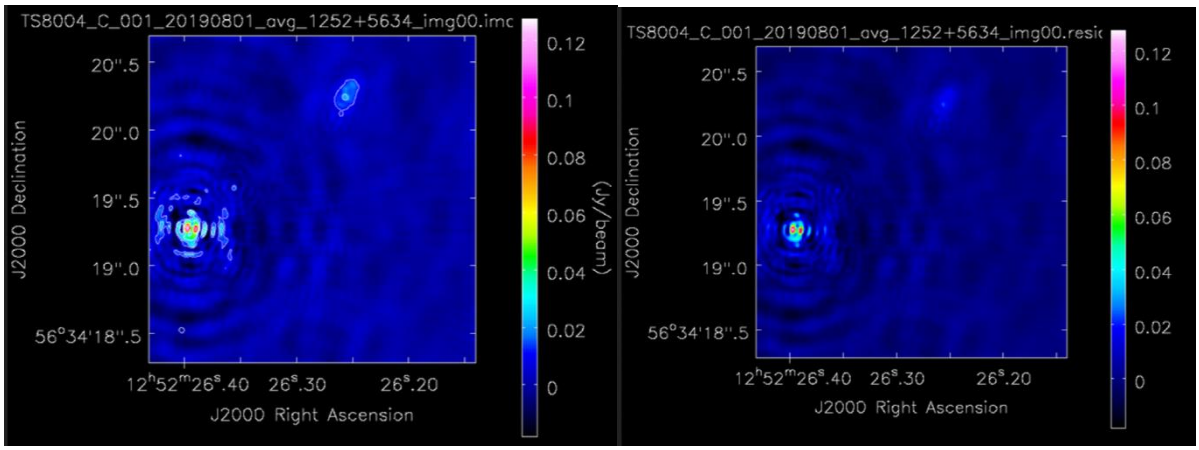
The auto-multithresh cleaning in the pipeline will automatically produce images, but it uses non-default values for some of the CASA tclean task parameters as they can work better for e-MERLIN images. The key parameters to have changed are:

- Minbeamfrac -> 0.3 (default e-MERLIN is 0.2)
- Growiterations -> 75 (default e-MERLIN is 25)
- Noisethreshold -> 5.0 (default e-MERLIN is 8)
- Sidelobethreshold -> 3 (default e-MERLIN is 1)

You can see that the images show much better fidelity on the compact structure but almost none of the diffuse structure is seen in the e-MERLIN images anymore. The peak flux is 127mJy and the residual rms is 1.797 mJy. This compares to the previous images which peaked at 117mJy but had a noise level of 0.8mJy. So the difference here is

in the image fidelity to diffuse emission. Note also that using the CASA default values makes the pipeline run quicker, but at a cost to the image fidelity to the diffuse structures.

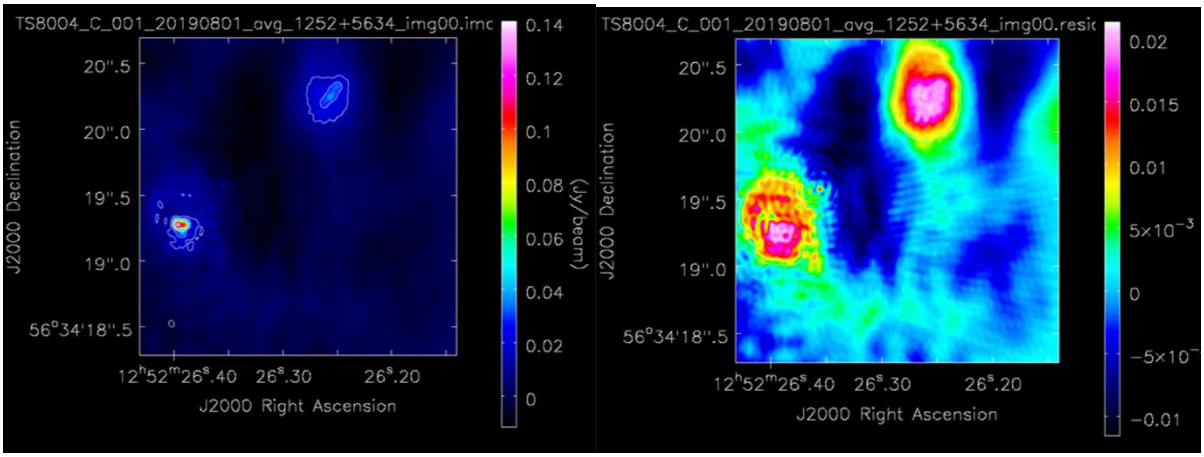
One thing you can do if you have poor uv coverage is play with these parameters, specifically the minbeamfrac parameter to improve the image fidelity as this will search a smaller area of the beam size for emission.



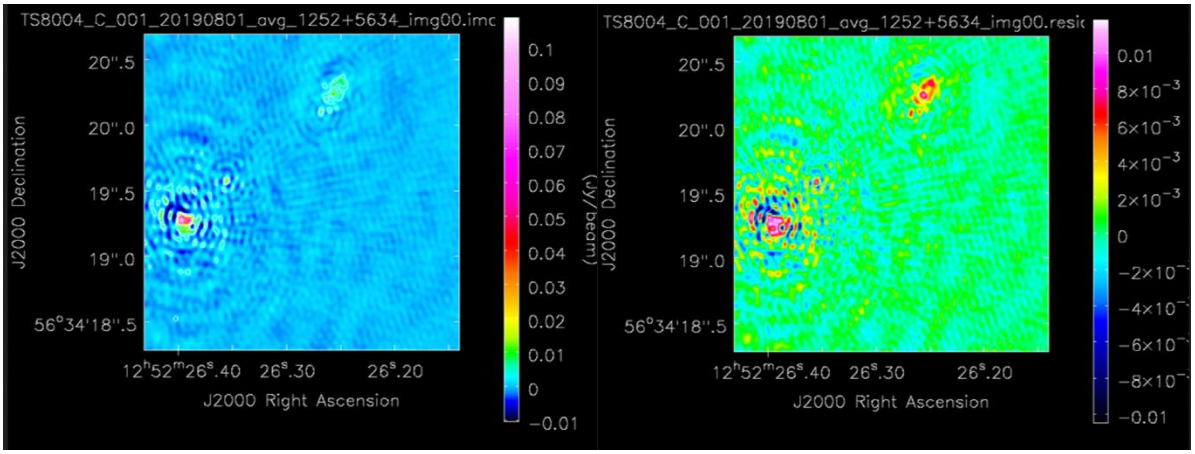
Example 2: Imaging with natural/briggs/uniform weighting

Below are some examples of just changing the weighting to natural or uniform. You can see quite clearly that the former brings out a lot of diffuse flux, but misses a lot of the compact emission, almost appearing like a double source, whereas the uniform map struggles with the most diffuse flux. The fluxes and rms values differ too, but mostly the difference is in the image fidelity, with the briggs weighting shown previously being the best middle ground between these two weighting schemes.

Natural: peak 141mJy, rms 2.5mJy



Uniform: peak 109mJy, rms 0.49mJy



Widefield Cleaning with wsclean

e-MERLIN has a large field of view for a long baseline interferometer, due largely to the antenna sizes and the possibility of reducing smearing by using unaveraged data. The best way to make widefield images is using wsclean by Andre Offringa. Below are some suggested wsclean variations to make widefield e-MERLIN images. Note that if you are using unaveraged data then the primary beam of e-MERLIN is ~30' at L band and ~7' at C band, but if using the e-MERLIN calibrated datasets in 4s and 4 channel averaging mode, the data will become time and bandwidth smeared at this point. Consider using the widefield calculator by Emil Lenc to work out how badly affected your data may be with smearing.

```
wsclean -save-source-list -weight briggs 0.5 -size 9000 9000 -name
./images/imagename -scale 0.004asec -niter 10000 -mgain 0.3 -auto-threshold 2 -
taper-gaussian 0.015asec -gain 0.1 -auto-mask 8 -beam-fitting-size 0.7 -field 0
-multiscale -multiscale-scales 0,3 project_target.ms
```

The above call to wsclean does several things:

`-weight briggs 0.5` is similar to the briggs/robust parameter in the tclean calls of the pipeline.

`-size 9000 9000` will make a 9000x9000 pixel map, with the pixel size set by the `-scale 0.004asec` parameter. In this case, this is set for a K band observation, but generally you want a pizel size about 5/6 times smaller than the synthesized beam size of a dataset.

`-name ./images/imagename` Is the image name, in this case it is putting it in the images folders already produced by the pipeline.

`-niter 10000` number of iterations, like the niter parameter previously. As wsclean is auto-masking and you want to clean out all sources in a widefield, this is set to an arbitrary high number.

`-mgain 0.3` using a Cotton-Schwab algorithm this will reduce the peak by 0.3 for each major iteration. This is usually ok for a quick look widefield image with e-MERLIN data, but try a higher value for more complex fields.

`-taper-gaussian 0.015asec` This adds a gaussian taper at 0.15 arcseconds – see the documentation for more information.

`-gain 0.1` This sets the minor loop cleaning gain and is different to the `-mgain` parameter. This is set to the wsclean defaults and the wsclean documentation suggest that this parameter rarely needs changing.

`-auto-mask 8` and `-auto-threshold 2` seem to give good auto masking levels for e-MERLIN data similar to those in tclean but play around with these as it has not been extensively tested.

`-beam-fitting-size 0.7` this helps to fit the beam when cleaning.

`-field 0` if working on a single source dataset, then this will clean only that source, but you can set field IDs here.

`-multiscale -multiscale-scales 0,3` This is the wsclean version of multiscale cleaning implemented in CASA. See the documentation for more information.

`project_target``.ms` This is your measurement set with only the target source split out.

Self-calibration

The process of self-calibration improves the data and images by iteratively improving the model on shorter timescales to get a better representation of the true sky brightness distribution over time. Where the calibration steps assumed a point source nearby calibrator to correct for the atmospheric changes over time. However, the phase calibrator is only visited once every 8 minutes, and the atmosphere can vary on the timescales of the target scan itself. Therefore, if we have a bright source, we can use the imaging and self-calibration process to model the atmosphere on shorter solution intervals and tweak the phase or amplitudes of the target field to improve the image. The ERIS 2022 tutorials on self-calibration are useful for this process. This section describes how to self-calibrate the *phase calibrator* which can then be applied to the target field. This may be useful to do for the case when you have a slightly resolved phase calibrator. If you want to speed up cleaning, then try averaging the data in time and frequency first.