The e-MERLIN CASA Pipeline

Javier Moldon
January 25, 2018

Contents

1. Introduction to the e-MERLIN CASA Pipeline (eMCP)

The e-MERLIN CASA Pipeline is a python package composed of different modules that can be run together sequentially to calibrate an e-MERLIN data set, producing calibration tables, calibrated data, assessment plots and a summary weblog. In the most general mode, it can take FITS-IDI files provided by the observatory and will produce a calibrated data set and initial crude images to check the quality of the observation. It can process automatically L and C band continuum e-MERLIN data sets.

The objective is to provide a common framework with standard, optimized calibration rules specific to e-MERLIN. It is meant to be used by any user interested in reducing e-MERLIN data. The key idea is that the pipeline provides and easy, ready-to-use toolkit that delivers calibrated data in a consistent, clear and repeatable way. We tried to keep the dependencies at minimum so all external users can use it. The only requirements are CASA (v4.7+), python (v2.7), and optionally AOFlagger[1] (to conduct automatic RFI flagging). All scripts of the pipeline can be downloaded from the open github repository[2] and used immediately as no further installation is required.

The pipeline is based on simplicity so it can work with minimum input from the user: usually the minimum input required is a project name, and the name of the sources to used. The pipeline accepts other common but optional parameters like the reference antenna or a list of flagging commands to discard additional bad data. The rest of the parameters are fixed by default and have been chosen to produce a successful calibration for the majority of the standard e-MERLIN operations (see limitations below).

To run the pipeline the user just needs to fill a pre-designed inputs file with the basic parameters and with the list of steps to run, and execute the pipeline as a CASA script from the terminal. Apart from the different outputs (data, calibration table, plots, etc), the pipeline will produce a text log with detailed information of what has been executed and an html weblog containing all results.

[1] https://sourceforge.net/p/aoflagger/wiki/Home/
[2] https://github.com/e-merlin/CASA_eMERLIN_pipeline

The development of the eMCP has been coordinated openly on github. That means that the code for all versions of the pipeline are freely accessible. Github also includes an issue tracker so it is easy to open a ticket with any question or request, and also to visit all the previous history of tickets. Github also hosts the pipeline documentation[3] with detailed explanation of all scripts, steps, parameters and products of the pipeline. This platform is ideal to develop and improve the pipeline, opening the possibility to implement new features, fix bugs, and have feedback from the community at any time.

2. Pipeline procedures and products

The pipeline procedures can be divided in two main blocks. The first pre-processing part is meant to transform a series of raw FITS-IDI files into a ready-to-use measurement set (MS). The second part will conduct standard calibration on continuum data. This is a brief summary list of the main steps conducted in each part:

- **Pre-processing**
  - Convert FITS-IDI into MS.
  - Fix visibility UVW values, flag autocorrelations.
  - Apply Hanning smoothing if needed.
  - Autoflag data to remove RFIs using Aoflagger. This step uses specific strategies for different types of sources and can work with C and L band data.
  - Remove data based on a-priori e-MERLIN knowledge: antenna slewing, edge channels, etc.
  - Optionally read list of flag commands prepared by the user.
  - Average data to a standard channel and time resolution.
  - Produce visibility plots (amplitude and phase vs time and frequency) for all baselines and sources.
  - Save flag status
- **Calibration**
  - Optionally read additional list of flag commands prepared by the user.
  - Initialize source models, using observatory models for the flux scale calibrator 3C286.
  - Obtain bandpass table.
  - Automatically flag data using *'tfcrop'* mode in CASA task *flagdata*.
  - Obtain delay calibration.
  - Obtain phase and amplitude gain corrections.
  - Find absolute flux density scale by bootstraping from 3C286.
  - Re-calculate bandpass table including spectral index information.
  - Re-calculate amplitude calibration including spectral index information.
  - Apply all previous calibration to the data.
  - Automatically flag data using *'rflag'* mode in the CASA task *flagdata* using corrected data.
  - Produce corrected visibility plots (amplitude and phase vs time and frequency) for all baselines and sources.
  - Produce automatic images of target and phase reference source to assess quality of the calibration.

[3] https://github.com/e-merlin/CASA_eMERLIN_pipeline/blob/master/documentation/docs.md

- Produce weblog organizing all the information produced by the pipeline in an html file.

All calibration steps will also produce plots of the calibration tables, and the users can choose if they want to apply the calibration up to a particular step in the process. Therefore, the user can easily find solutions, apply them and check the effect on any step, and repeat the operation as many times as needed without disturbing the data and without having to restart the process from the beginning.

One of the products of the pipeline is the weblog, a set of html pages that summarizes the observation and the pipeline process and shows assessment plots. It is organized in five pages:

- *Home*. A summary of the technical specifications of the observation (dates, frequency configuration, etc).
- *Observation summary*. Quick access to source names, source separations, scan distribution, frequency configuration, list of antennas, plots of source elevation against time, and plots of uv-coverage of each source.
- *Calibration*. Plots of all calibration tables produced by the pipeline, and a summary of the parameters used to produce ach table. Useful to quickly assess the quality of the calibration.
- *Plots*. Amplitude and phase plots against time and frequency for each baseline and source, for uncalibrated and calibrated data. Also include calibrated amplitude vs uv-distance plots. Useful to check how the calibration changes the visibilities and understand the source properties.
- *Images*. The pipeline produces automatic images of targets and calibrators without any human intervention. This is useful to have a first impression on the source structure and also to assess the quality of the pipeline calibrated data.

3. Current purpose and limitations. Future plan

The pipeline is meant to be executed mostly unattended. That means that it is capable of producing reasonably good calibration on most of the scheduled projects. In that sense, the pipeline is a tool to quickly execute tasks with optimized parameters. However, for a final high quality scientific result the intervention of an experienced radio astronomer is required. That being said, our experience shows that for the most simple case, continuum C-band observations, good quality images can be obtained with minimum human intervention, basically consisting on identifying small amounts of bad data that the autoflagger could not eliminate.

We note that only basic calibration is performed by the pipeline. Detailed imaging and self-calibration are not implemented, although the pipeline contain tools already available but under development that can help to self-calibrate data. The pipeline does not have a mode to include external phase calibrator models into the calibration process, but including a model can be done manually before starting the calibration process, and that would not affect the pipeline operations.

Current version of the pipeline (v0.7) can deal with C- and L-band observations. It cannot automatically calibrate K-band data, although it contains the tools to conduct the calibration with human intervention. Also, the parameters used during the calibration are optimized for C and L band data, and not specifically for K band.

e-MERLIN is capable of observing simultaneously in wideband continuum mode and also record high spectral resolution data zooming in specific parts of the band. The pipeline will split the continuum and the spectral line data into different MS. Currently, the pipeline can only calibrate continuum data, and will not modify or calibrate the spectral line data. It is possible to run specific steps on the spectral line data, but a full data reduction is currently not possible.

There are plans to implement a series of upgrades to the pipeline, always trying to make the changes user friendly, and implementing procedures known to work reliable for the majority of the e-MERLIN data as automatically as possible. Our list of priorities or features under development for the future are:

- Self-calibration. Probably implement a stand-alone script to automatically self-calibrate individual sources that had been previously calibrated with the main pipeline.
- Polarization calibration. We are currently investigating a general approach to include polarization calibration in the main pipeline. If the procedure is robust, we may run it by default in all C and L band observations.
- Spectral line calibration. There are different possible spectral observing modes and types of sciences cases, so we are discussing what would be the best and cleanest implementation. We will probably implement an optional automatic step in the main pipeline for the most obvious cases, and we will prepare a set of independent tools to be run manually on more complicated cases.
- Widefield imaging. This is not a fundamentally different step, but there are several issues that should be taken into account when imaging wide fields. We may implement a simple mode to generate these type of images with a generally optimized set of parameters.

Additionally, we have a partially developed mode to be used only by the observatory. This mode is meant to calibrate automatically and daily the bright calibrators observed regularly (3C286, 3C86 and OQ208). The idea is to have a regular QA system to monitor the performance of e-MERLIN as a whole. We aim to monitor and update a database containing information such as array sensitivity, antenna performance and sensitivity, phase stability, monitor RFI environment, etc.

4. Product examples

Example 1: pipeline log

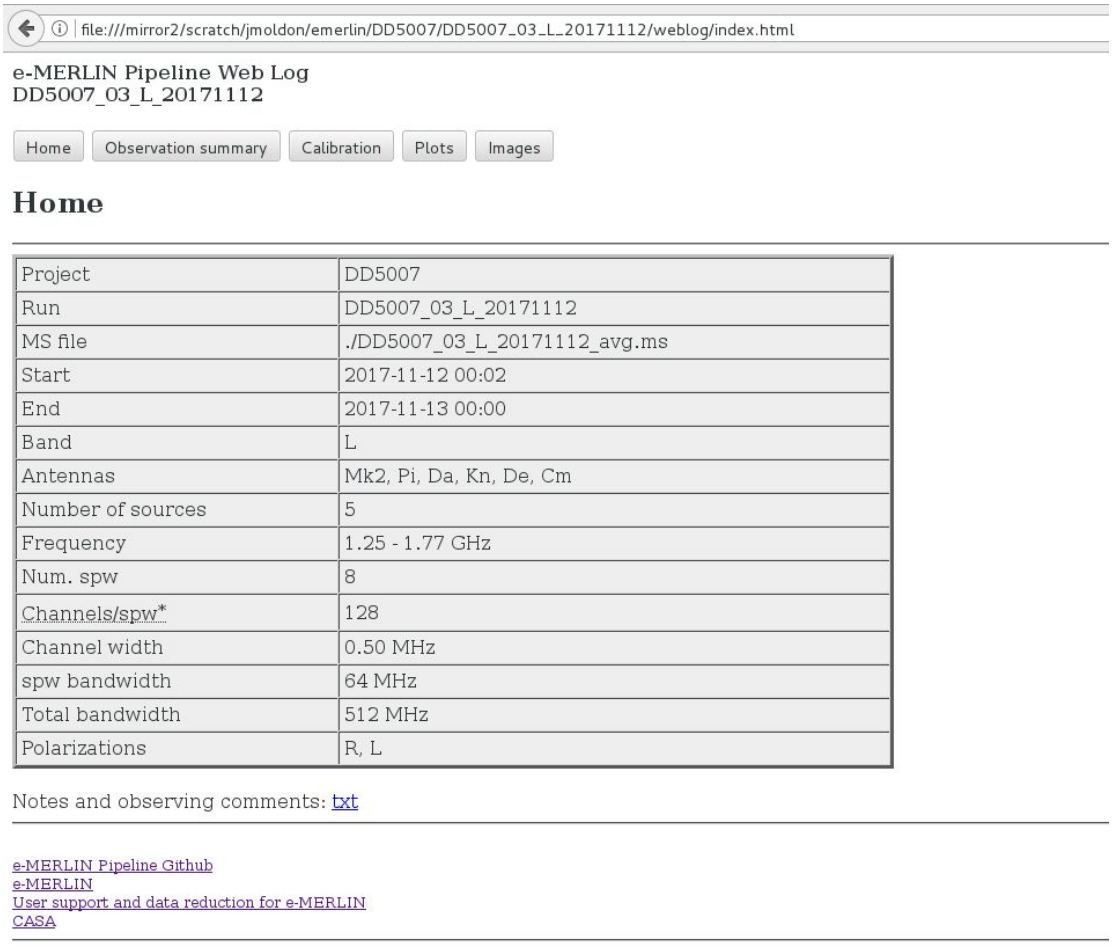
Example of text written in the pipeline log eMCP.log during a-priori flagging:

```
2018-01-19 15:47:57 | INFO | Reading ms file information for MS: CY6004_C_001_20180114.ms
2018-01-19 15:47:57 | INFO | Targets:   1309-2322,1259-2310
2018-01-19 15:47:57 | INFO | Phasecals: 1311-2329,1311-2329
2018-01-19 15:47:57 | INFO | Fluxcal:   1331+305
2018-01-19 15:47:57 | INFO | Bandpass:  1407+284
2018-01-19 15:47:57 | INFO | Pointcal:  0319+415
2018-01-19 15:47:57 | INFO | Sources in MS CY6004_C_001_20180114.ms: 1311-2329,1331+305,1259-2310,1407+284,0319+415,1309-2322
2018-01-19 15:47:57 | INFO | Antennas in MS CY6004_C_001_20180114.ms: ['Mk2', 'Pi', 'Da', 'Kn', 'De', 'Cm']
2018-01-19 15:47:58 | INFO | Refant: Pi
2018-01-19 15:47:58 | INFO | Start flagdata1_apriori
2018-01-19 15:47:59 | INFO | Antennas in MS CY6004_C_001_20180114.ms: ['Mk2', 'Pi', 'Da', 'Kn', 'De', 'Cm']
2018-01-19 15:47:59 | INFO | MS has 512 channels/spw
2018-01-19 15:47:59 | INFO | Flagging edge channels *:0~3;508~511
2018-01-19 15:48:09 | INFO | Flagging first 20 sec of all sources.
2018-01-19 15:48:21 | INFO | Flagging 5 min from bright calibrators
2018-01-19 15:48:27 | INFO | Flagging first 20.0 sec of target 1309-2322 and phasecal 1311-2329
2018-01-19 15:48:37 | INFO | Flagging first 30.0 sec of target 1259-2310 and phasecal 1311-2329
2018-01-19 15:48:44 | INFO | New flags applied: flagdata1_apriori
2018-01-19 15:48:44 | INFO | Current flags applied: ['flagdata1_apriori']
2018-01-19 15:48:44 | INFO | End flagdata1_apriori
```

Example of eMCP.log describing one section of the delay calibration step:

```
2018-01-19 15:55:05 | INFO | Start solve_delays
2018-01-19 15:55:05 | INFO | Running gaincal to generate: delay.K1
2018-01-19 15:55:05 | INFO | Field(s) = 0319+415,1407+284,1311-2329,1331+305, gaintype = K, calmode = p
2018-01-19 15:55:05 | INFO | solint = 300s, spw = *:13~115,  combine = spw
2018-01-19 15:55:05 | INFO | Previous calibration applied: ['bpcal.B0']
2018-01-19 15:55:05 | INFO | Previous calibration gainfield: ['1407+284']
2018-01-19 15:55:05 | INFO | Previous calibration spwmap: [[]]
2018-01-19 15:55:05 | INFO | Previous calibration interp: ['nearest,linear']
2018-01-19 15:55:05 | INFO | Generating calibration table: ./calib/CY6004_C_001_20180114_delay.K1
2018-01-19 15:55:21 | INFO | caltable delay.K1 in ./calib/CY6004_C_001_20180114_delay.K1
2018-01-19 15:55:21 | INFO | Delay calibration delay.K1: ./calib/CY6004_C_001_20180114_delay.K1
2018-01-19 15:55:25 | INFO | Delay calibration plot: ./plots/caltables/CY6004_C_001_20180114_delay.K1_2.png
2018-01-19 15:55:25 | INFO | End solve_delays
```

Example 2:
Weblog home page with general information (this page) and observation summary (next page).



file:///mirror2/scratch/jmoldon/emerlin/DD5007/DD5007_03_L_20171112/weblog/index.html

e-MERLIN Pipeline Web Log
DD5007_03_L_20171112

Home | Observation summary | Calibration | Plots | Images

Home

Project	DD5007
Run	DD5007_03_L_20171112
MS file	./DD5007_03_L_20171112_avg.ms
Start	2017-11-12 00:02
End	2017-11-13 00:00
Band	L
Antennas	Mk2, Pi, Da, Kn, De, Cm
Number of sources	5
Frequency	1.25 - 1.77 GHz
Num. spw	8
Channels/spw*	128
Channel width	0.50 MHz
spw bandwidth	64 MHz
Total bandwidth	512 MHz
Polarizations	R, L

Notes and observing comments: txt

e-MERLIN Pipeline Github
e-MERLIN
User support and data reduction for e-MERLIN
CASA

Observation summary

Summary:

Summary of the observation (listobs): txt

Sources:

Source pairs and separations: txt

Target	Phase cal	Separation [deg]
2032+4127	2007+4029	4.72

Sources in MS:

Source	Intent
0319+415	ptcal
2007+4029	phscals
1331+305	fluxcal
2032+4127	targets
1407+284	bpcal

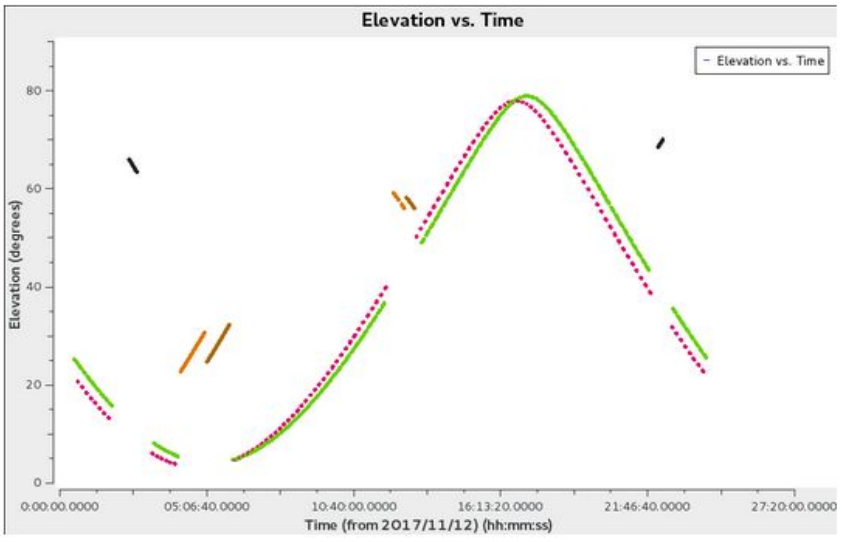
*All sources specified in the inputs file are in the MS.

Antennas:

Mk2
Pi
Da
Kn
De
Cm

Reference antenna: Mk2

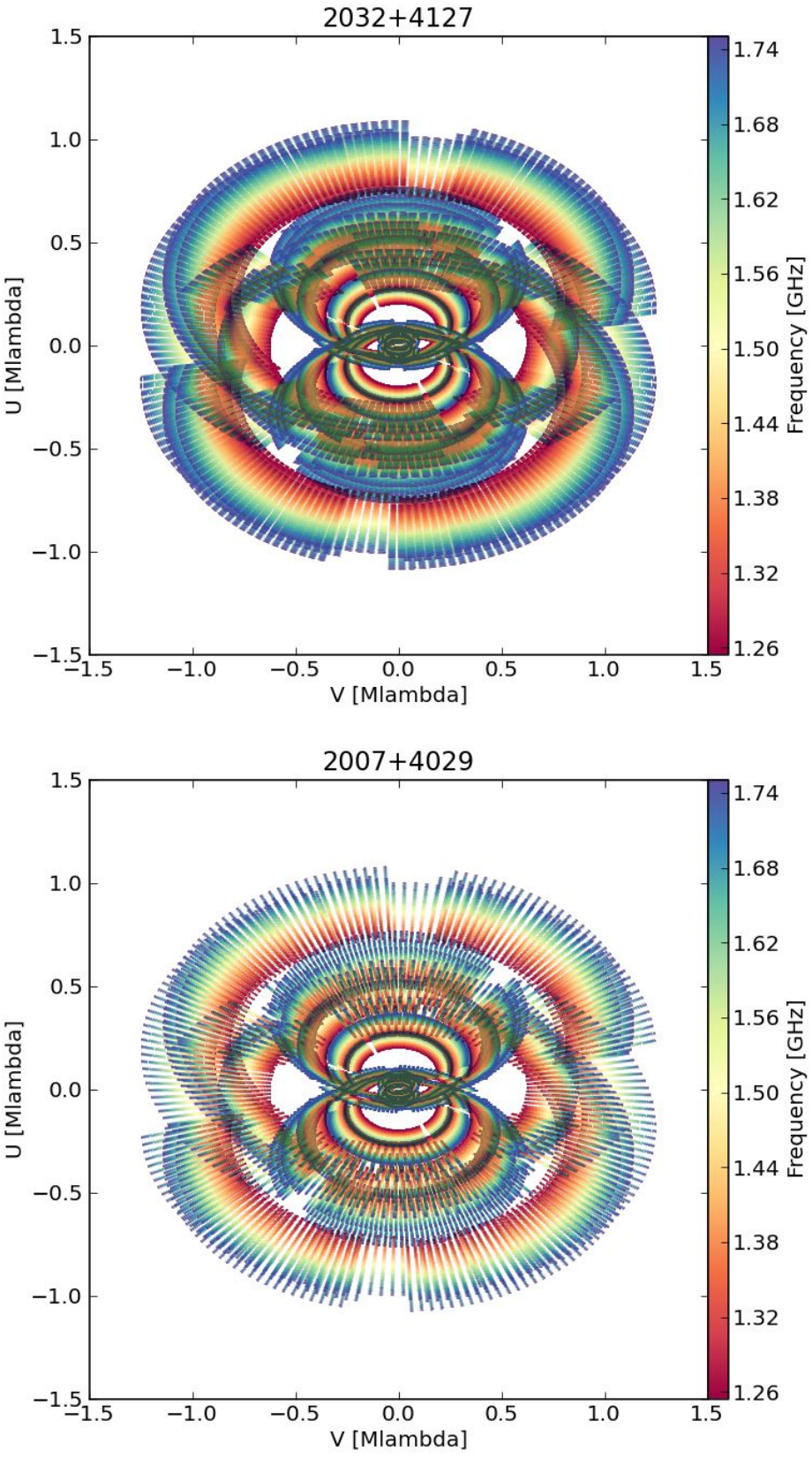
Source elevation:



UV coverage:

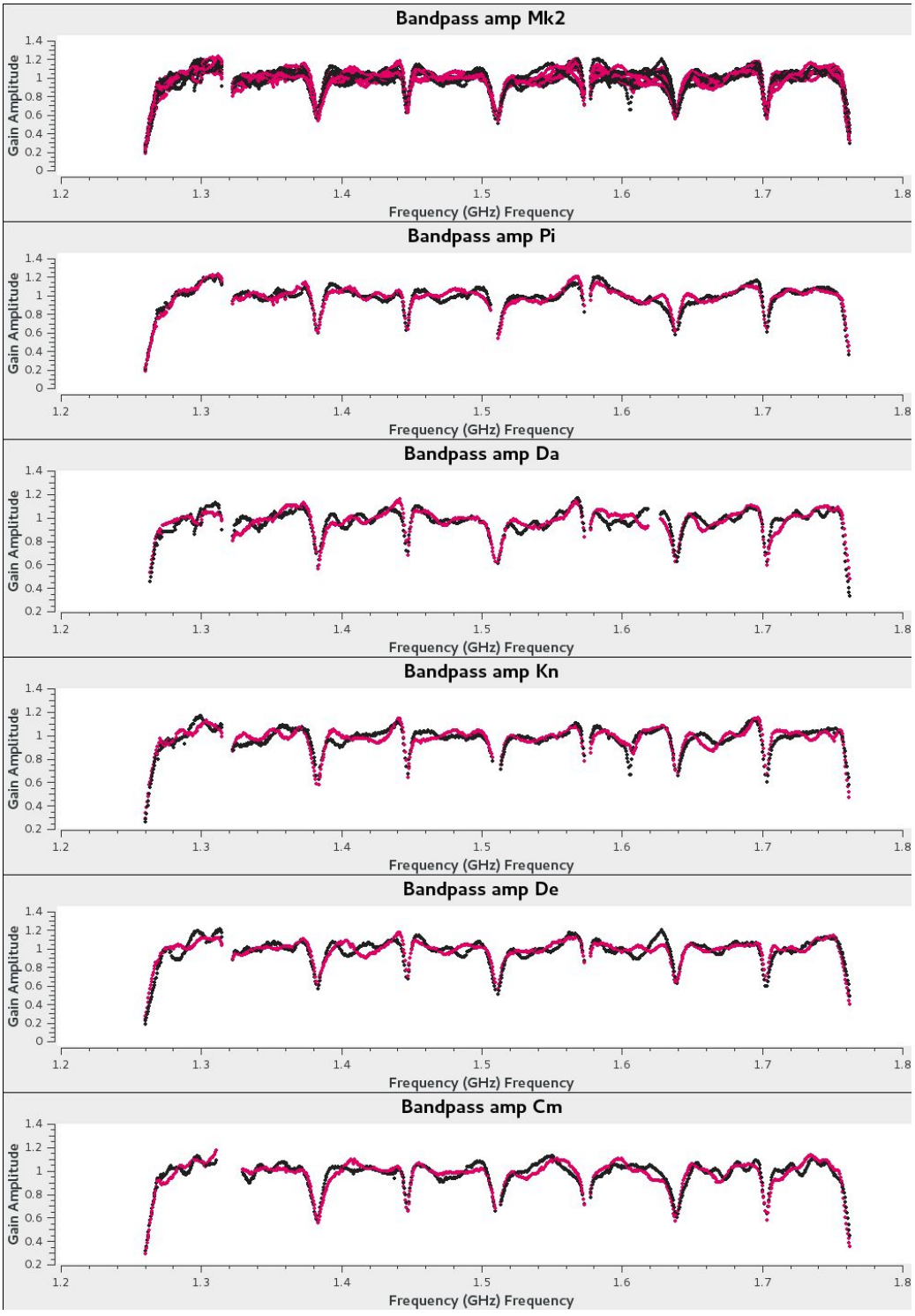
Example 3: UV coverage

UV-coverage plots for a target source (top) and a phase calibrator (bottom).



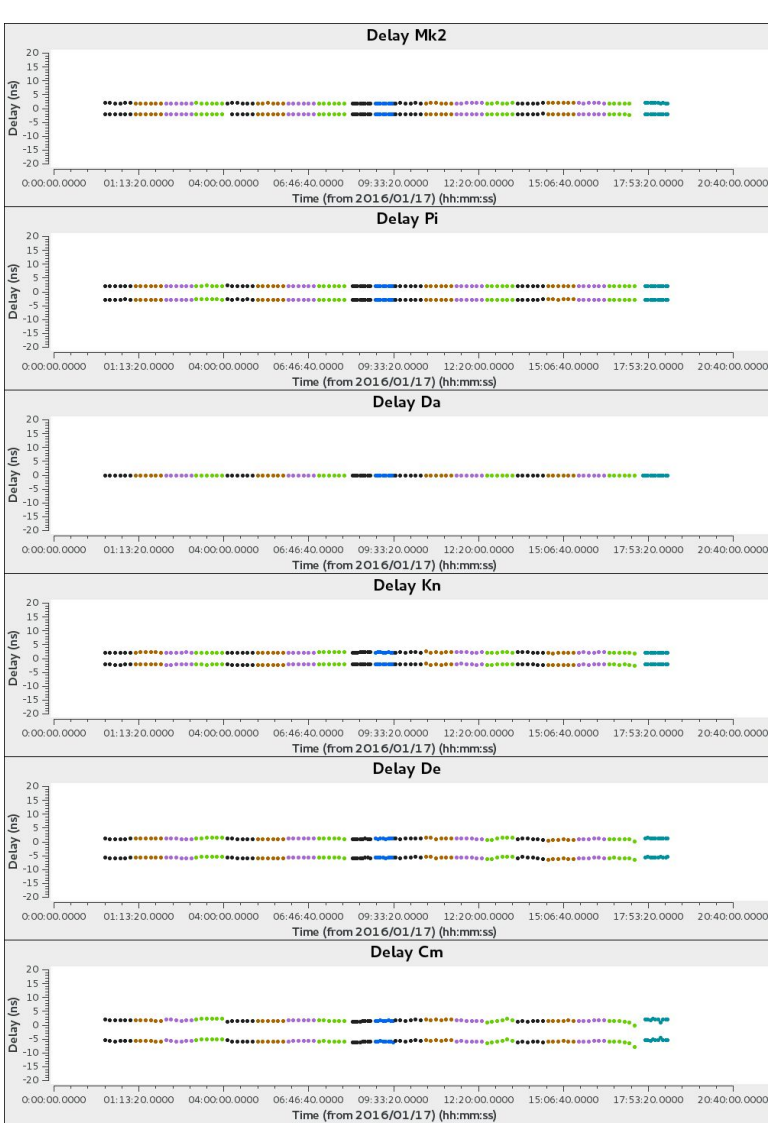
Example 4: bandpass table

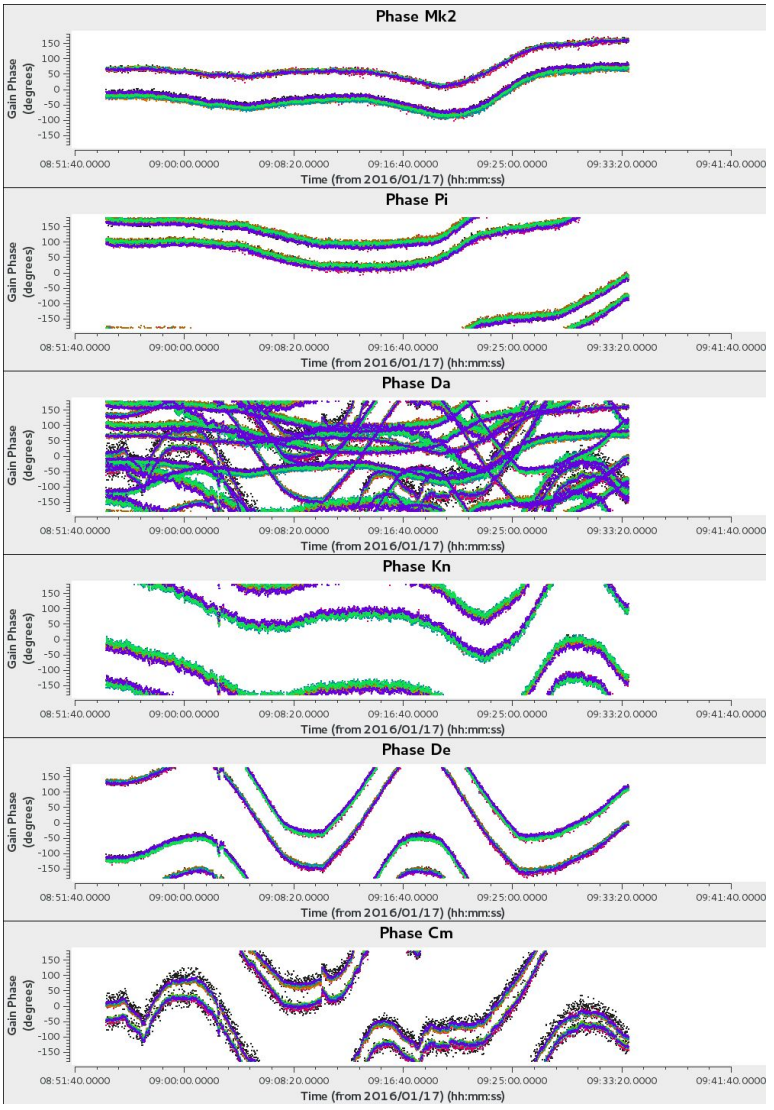
Bandpass calibration table for an L-band observation after autoflagging with AOFlagger. The reference antenna is Mk2.



Example 5

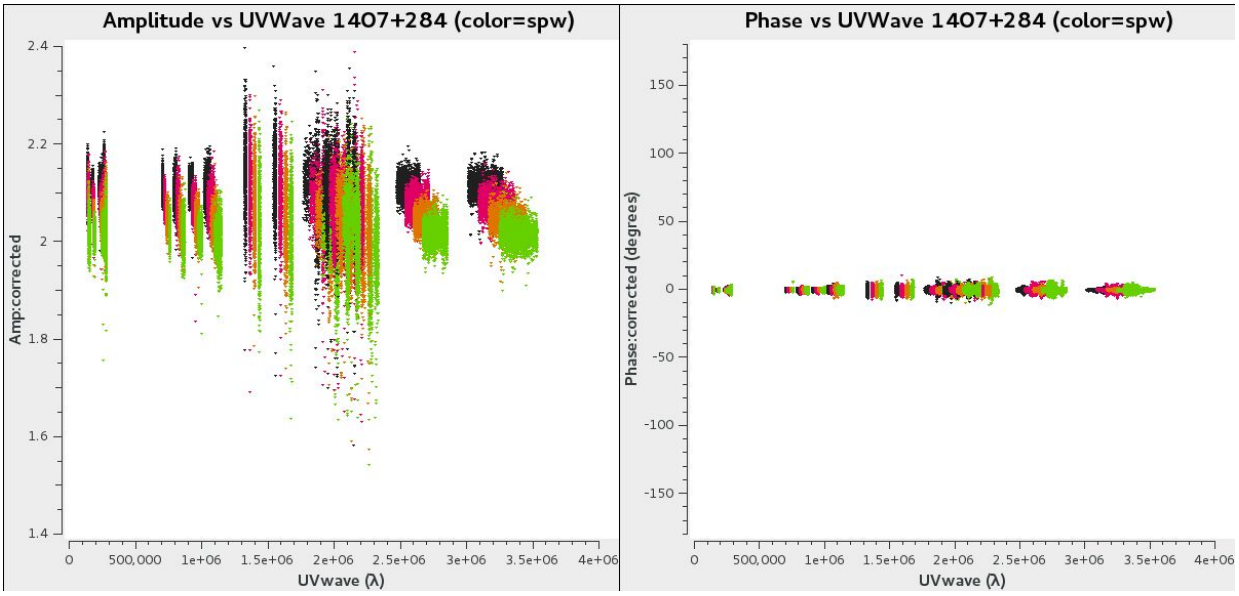
Calibration tables. *Left*: delay calibration for an L-band observation interleaving groups of 4 phase calibrator-target pairs (color coded by source, fixed range from -20 to +20 nanosec). *Right*: phase solutions for the bandpass calibrator after delay and bandpass correction (color coded by spw, reference antenna is Da).

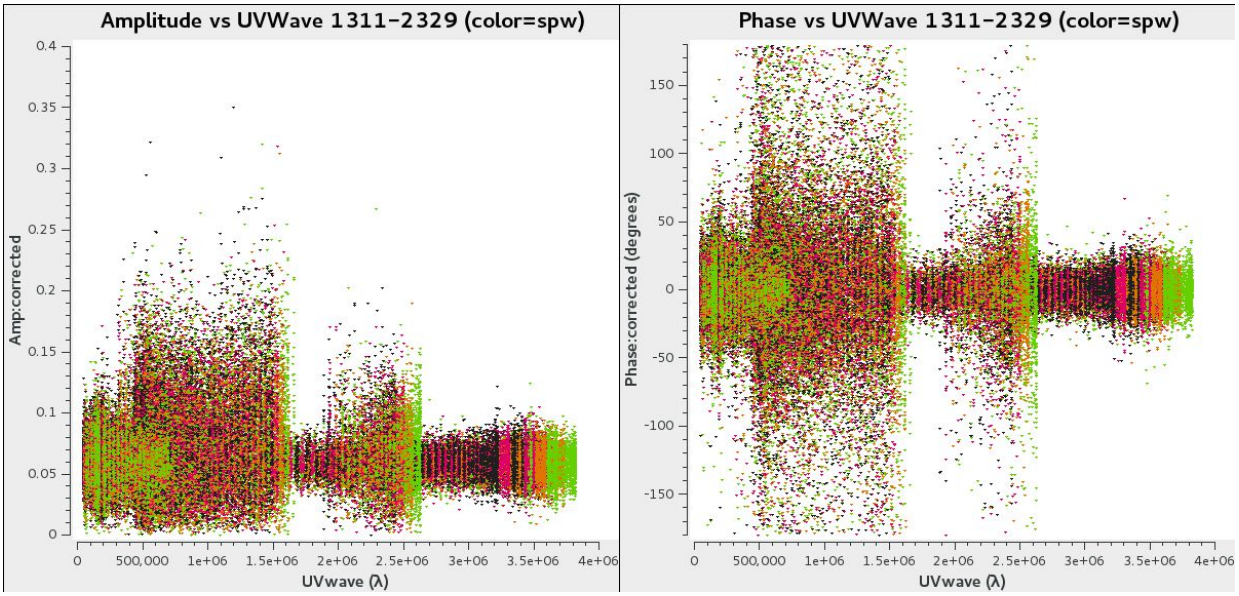




Example 6: UV plots of calibrated data

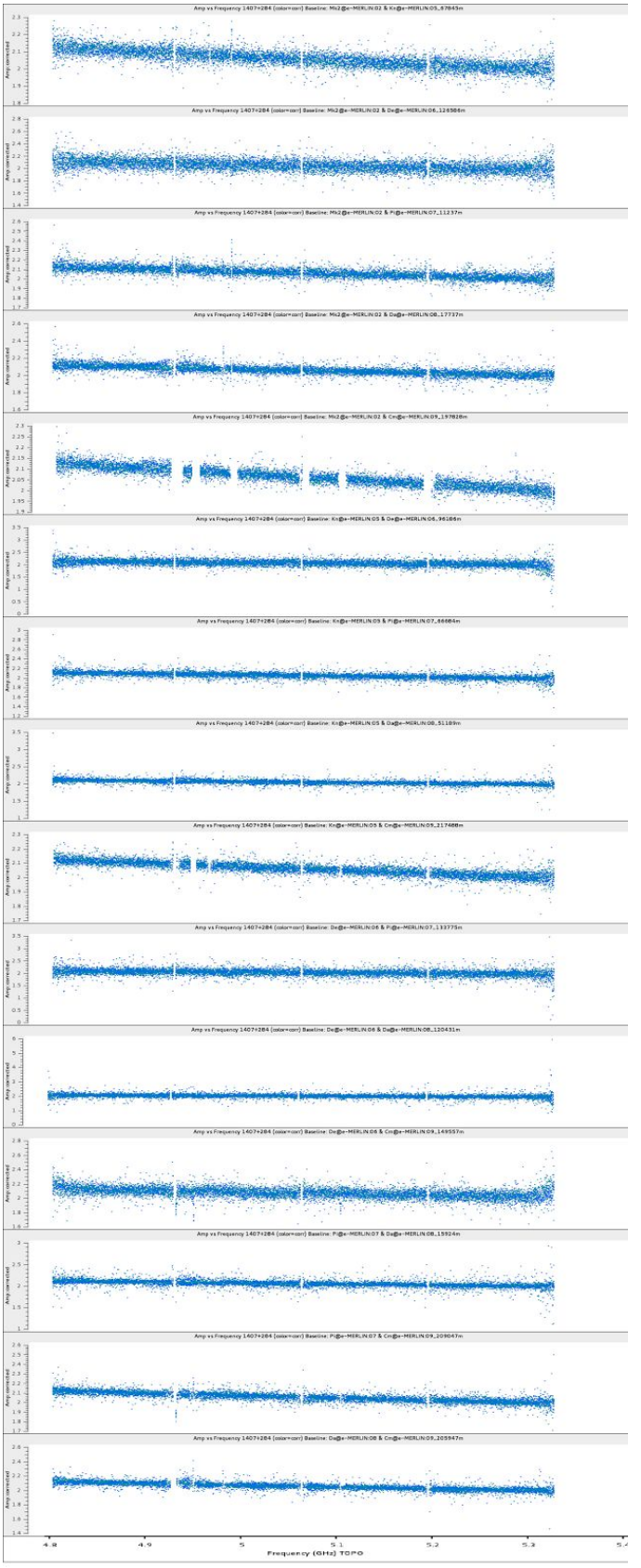
Top: visibility amplitude (left) and phase (right) vs uv-distance for the bandpass calibrator OQ208 at C band, colors represent different spectral windows (IFs). *Bottom*: the same for a 60 mJy phase reference source at a declination of -23 degrees. In both cases the higher noise data corresponds to baselines with the less sensitive antenna De.





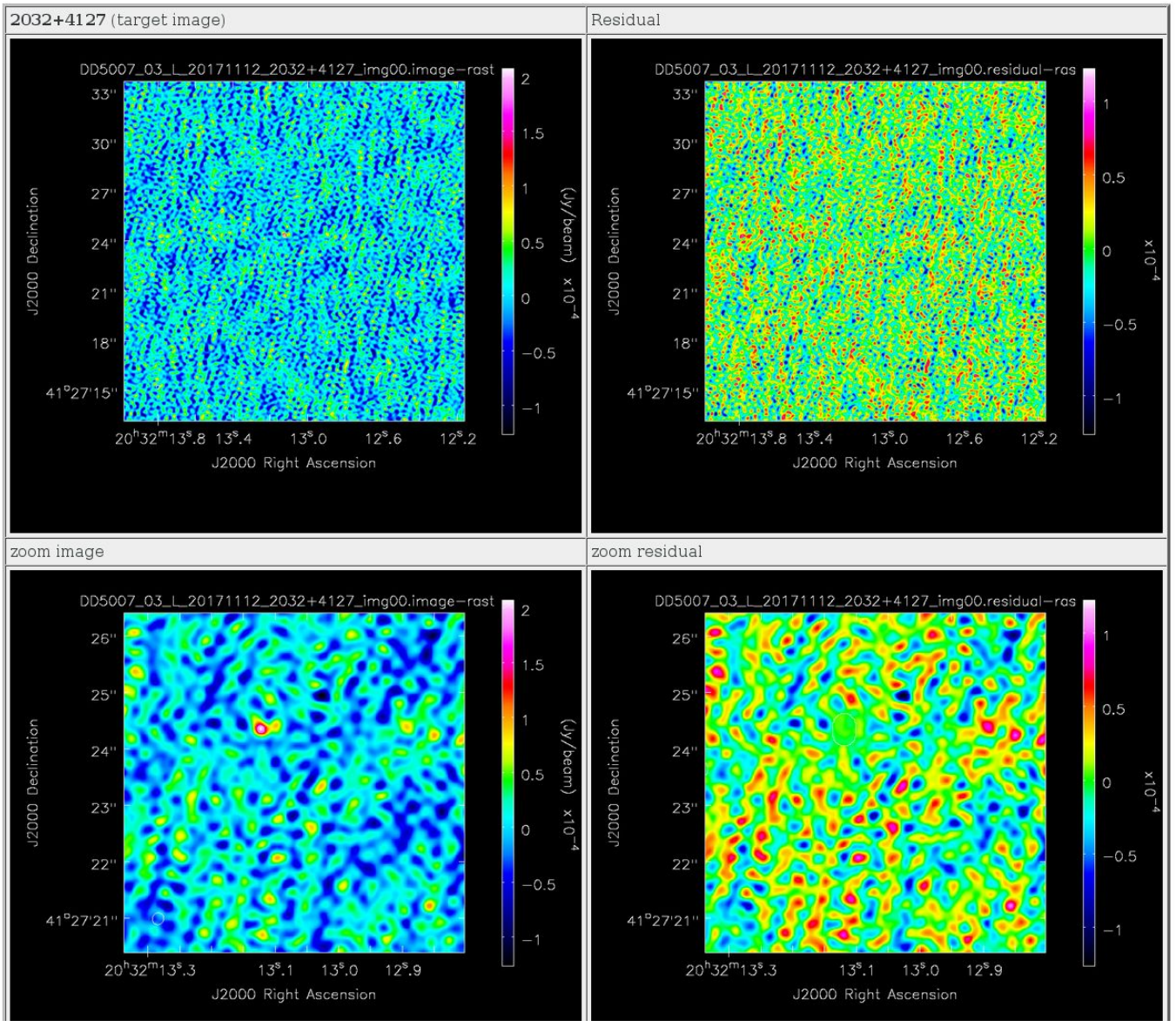
Example 7: visibility plots of calibrated data

Amplitude against time (left) and frequency (right) for the bandpass calibrator OQ208 at C band for each baseline.



Example 8: automatic imaging

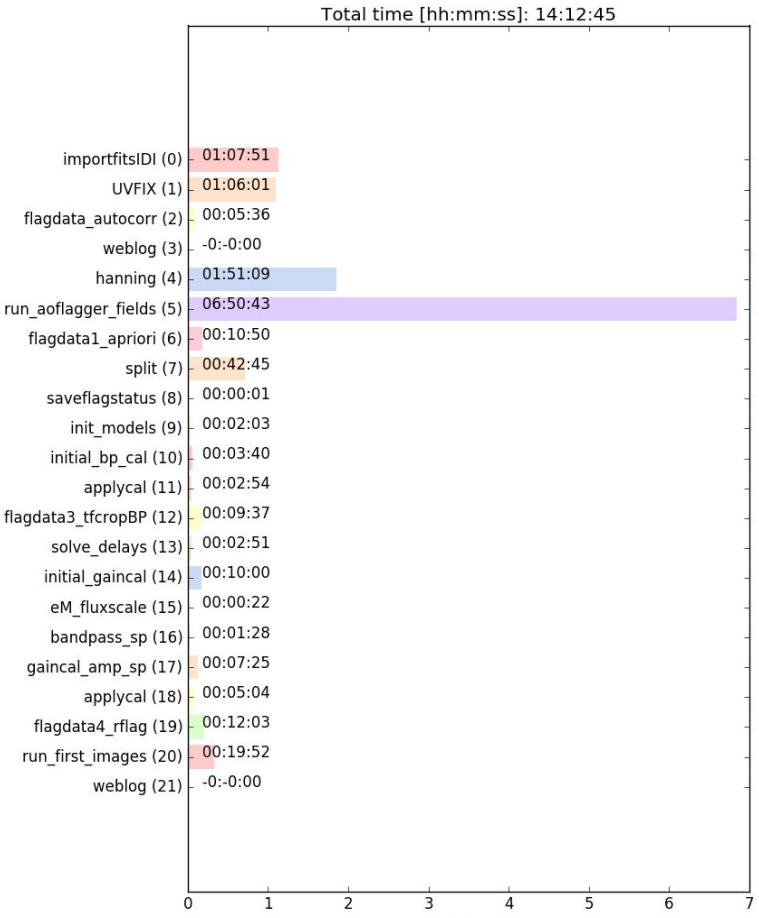
Example of an automatically generated image of an L-band data set calibrated by the pipeline. The top row shows the clean image (left) and the residuals (right). The bottom row is just a zoom of the top row images. In this case, a 100 microJy source is clearly detected (see bottom-left plot) without any manual processing.

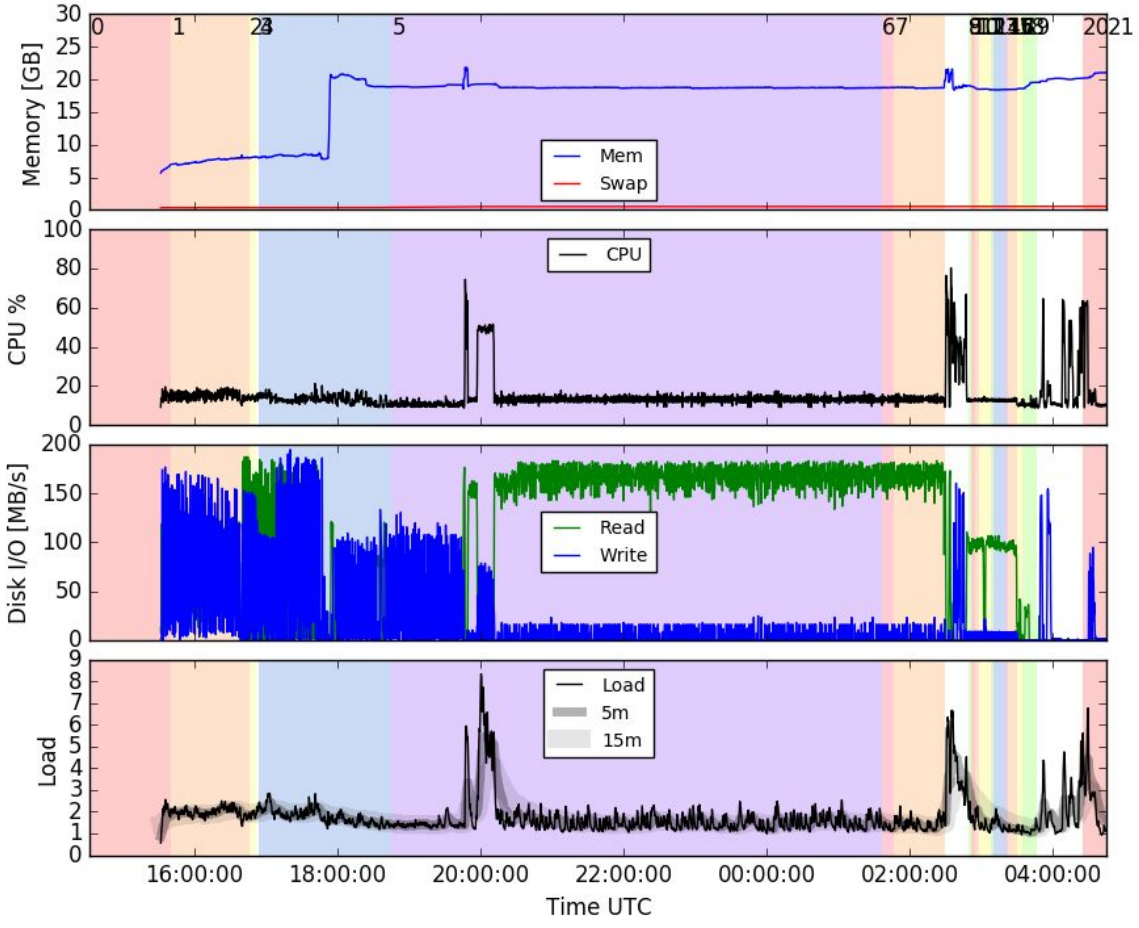


5. Pipeline performance

Test 1

The test was conducted on *flanux* (12 CPU @ 3.50GHz, 124 GB of RAM). The data set was a 24h L-band observation including the Lovell. The raw data has a size of 300 GB. The complete run of the pipeline (from reading FITS-IDI files to producing calibrated images) took 14.2 h. We note that automatic flagging with AOFlagger took 50% of that time, due to limitation of memory[4]. With more available memory, the whole process would have taken about 9h. In summary, with a machine with slightly more memory the observation-to-process time ratio would be below 0.4, allowing automatic pre-processing of all e-MERLIN data on real time.





[4] As seen in the bottom plot the memory never reached the maximum of 128 GB. However, AOFlagger uses a slower procedure that does not use memory when the available memory is not at least x2 the needed memory.

Test 2

The test was conducted on computer *richards* (12 CPU @ 3.50GHz, 64 GB of RAM). The data set is a 12h L-band observation, 512 chan/spw and 1 sec integration time. The original file size is 184 GB. In the split step the data are averaged to 128 chan/spw and 2 sec integration time. The total processing time (not including visibility plots and imaging) was 7.5 hours. Again, the most relevant bottleneck is the automatic flagging with aoflagger.