e-MERLIN CookBook - version 3.0, 2015 February

Editorial team: Pierre-Emmanuel Belles (PEB), Rob Beswick (RJB), Megan Argo (MKA), Neal Jackson (NJJ), Tom Muxlow (TWBM), Anita Richards (AMSR)

Revision history:
version 1.0 - 2012 (RJB, most early sections written)
version 2.2 - 2013 July (PEB, MKA, extensive expansion and later sections)
version 2.3 - 2013 August, revised 2013 September-October (NJJ)
version 2.4 - 2013 November, revision of v 2.3 (mainly appendices), after comments from AMSR (PEB); further editing (NJJ,AMSR)
version 2.4b - 2014 July, incorporated 3C286 models in section 7 (NJJ)
version 3.0 - 2014 December - 2015 February, extensive revision (NJJ et al)
A pdf version can be found at http://www.e-merlin.ac.uk/data_red/tools/e-MERLIN_cookbook_latest.pdf

0. Introduction

0.0 Getting help

For help with any stage of data reduction described in this cookbook, you can either talk to your e-MERLIN contact person, or email the helpdesk: emerlinhelp@jb.man.ac.uk.

0.1 e-MERLIN: brief description and capabilities

e-MERLIN is an upgrade to the existing hardware and telescopes of the MERLIN array, an interferometer system based at Jodrell Bank Observatory with a range of baselines from 6-220km. e-MERLIN comprises 5 remotely linked telescopes (Cambridge; Defford, near Malvern in Worcestershire; Knockin, near Oswestry, Shropshire; Darnhall, in west Cheshire; Pickmere, about 6km from Jodrell Bank), plus up to two local stations (MkII and Lovell telescopes) situated at Jodrell Bank Observatory (JBO). Details of the array can be found on the e-Merlin website, http://www.e-merlin.ac.uk. e-MERLIN uses new dedicated optical fibre connections to bring back wide-bandwidth astronomical data from each of the remote and local telescopes to a newly commissioned correlator situated at JBO. The purpose of this document is to provide a set of instructions for e-MERLIN data reduction. Since e-MERLIN is a new instrument, the methods outlined in this document are evolving and being constantly improved. Users should always consult with e-MERLIN staff for updates in the recommended data reduction path.

0.2 Basic prerequisites

You will need the following for the reduction of e-MERLIN data:

- A working copy of AIPS (Astronomical Image Processing System) which is available from http://www.aips.nrao.edu. This is distributed as binaries for most common operating systems and is relatively easy to install. If a multi-user version is installed on your system you may need to request an AIPS number from your system administrator. You will also need basic familiarity with AIPS syntax, including inspecting data and image files, setting inputs for tasks and running tasks. This is achievable using the AIPS cookbook, which is available at the same website. e-MERLIN analysis does not yet use the newer CASA system, because some necessary functionality is not yet available; porting the reduction to CASA is a long-term aim.
- A working copy of Parseltongue; this software is available from http://www.jive.nl/parseltongue/codex.html. Parseltongue provides a Python interface to AIPS and is used for most of the e-MERLIN scripts. In principle it is possible to do the data reduction without this, but you will not then have the advantage of the pre-written scripts. Parseltongue requires a number of other packages, notably the Obit system together with python, numpy and scipy.
- A computer capable of dealing with processing the 50-100Gb datasets typically generated by the e-MERLIN system. In practice, this usually means a high-end multi-core machine. The demanding parts of the pipeline are those which involve sorting or automatic editing of the data. Later parts of the analysis can be done on less powerful machines, provided the data is averaged sufficiently.

Much of the early part of the data analysis is done using automatic scripts written in Parseltongue. In order for these to work, you must always begin by loading the AIPS definitions and setups from the aips root directory (often /aips). Failure to do this causes immediate exit with a non-obvious error message.

- source $aips_root/LOGIN.CSH

Some scripts can be executed from the command line using arguments:

- parseltongue my-script.py arg1 arg2 arg3...

but the more common method is to prepare an input file of instructions and pass it to parseltongue:

- parseltongue my-script.py input-file

You should check the inputs carefully, as the main pipeline script takes a long time to run.
In this guide, instructions will frequently be given for the running of AIPS tasks. These tasks have a number of parameters which are set by the user, and which can be reset using the command `default taskname` for a particular task `taskname`. Throughout this guide, the instruction, for example:

- Run `UVFLG` with `antenna=7` and `stokes='LL'`

means that you should type the following series of commands:

- default uvflg
- getn [N]
- antenna 7
- stokes 'LL'
- go uvflg

where N is the catalogue number of the file you want to operate on, found by `pcat`; you can also give its full specification; see `help uvflg` for more information.
Additional information about the parameter settings is then given in notes after each instruction. Remember that AIPS parameters are common to different tasks in many cases, so you should always either specify the default explicitly, or review the inputs carefully to make sure that none are left over from the last task.

0.3 e-MERLIN data

The e-MERLIN correlator is a flexible and powerful machine which is capable of producing data with a wide range of spectral and time resolution and array composition, depending on the scientific objectives. The default data products vary with the observation frequency (L-band at around 1.5GHz, giving a resolution of 160mas and C-band around 5GHz, giving a resolution of 50mas) and with observation cycle (cycle 0 Feb-Jul 2012, cycle 1 Sep 2013 - Feb 2014) as the system develops. The lists below summarise the properties of the data from these first two cycles.

L-Band, cycle 0

- Standard continuum mode:
  - Data are split into 8 to 12 sub-bands;
  - Each sub-band is 32 MHz wide (12 by 32 MHz = 384 MHZ /BW), divided into 512 spectral channels;
  - Available bandwidth is 1350-1750 MHz if observations were made pre-July 2012 (when filters were not installed on all telescopes), 1250-1750 MHz if observations were made in July 2012 (new filters installed);
  - Expected rms noise for a 12 hr on-source observation run: 6 μJy/beam, twice this value if Lovell Telescope not included.
- Spectral-line mode:
  - Ten 32 MHz spectral windows can be placed within the available bandwidth, and four more can be used at higher spectral resolution. Note that not all sub-bands may be usable due to RFI;
  - Position and width (32 MHz, 16 MHz, 8 MHz, ..., 0.25 MHz) of sub-bands are flexible, but using standard 512 chan/sub-band/pol.

L-Band, cycles 1 and 2

- Standard continuum mode:
  - Data are split into 8 sub-bands;
  - Each sub-band is 64 MHz wide (8 by 64 MHz = 512 MHZ /BW), divided into 512 spectral channels;
  - Available bandwidth is 1250-1750 MHz;
  - Expected rms noise for a 12 hr on-source observation run: 6 μJy/beam, twice this value if Lovell Telescope not included.
- Spectral-line mode:
  - Up to four additional spectral-line sub-bands (narrower spectral windows) are available to be placed on individual spectral features;
  - Position and width (32, 16, 8 MHz, ...) of sub-bands are flexible, but using standard 512 chan/sub-band/pol.

C-band, cycles 0, 1 and 2

- Standard continuum mode:

  - Data are split into 4 sub-bands;
  - Each sub-band is 128 MHz wide (4 by 128 MHz = 512 MHZ /BW), divided into 512 spectral channels;
  - Total 512 MHz bandwidth tuneable within 4.5-7.5 GHz (contiguous 512 MHz band). Standard (default) band is in the centre of tuneable range;
  - Expected rms noise for a 12 hr on-source observation run: 7 μJy/beam, twice this value if Lovell Telescope not included. This assumes the 0.512 GHz bandwidth is placed in the most sensitive part of the band (about 5.8-6.3 GHz) and minimal losses due to interference.

- Spectral-line mode:
  - Up to four additional spectral-line sub-bands (narrower spectral windows) are available to be placed on individual spectral features;
  - All sub-bands are correlated into 512 channels/pol/sub-band, but the width of the sub-band can be decreased (e.g. 128 MHz, 64 MHz, 32 MHz, ..., 0.25 MHz).

0.4) Data format

After observation the correlated data are exported to a FITS-IDI standard. For a given observing run, each FITS file contains a single source, with multiple sub-bands per FITS file if these sub-bands are spectrally identical. For spectral-line experiments (where multiple, different sub-band settings are used) multiple FITS-IDI files will be created for each of the different sub-band set-ups. This is due to limitations in FITS-IDI standards.
The FITS-IDI data is hosted at JBO close to the correlator on archive disks. On medium-term scales these data are available for transfer from these machines to local disks for data processing. The long-term archive (data older than 1yr) will consist of cold-media (tapes and/or static disks) as multiple copies for data security. These data will be accessible both at JBCA Manchester and JBO.
You will probably find that a number of different sources have been observed as part of the run; the source name is given as part of the (typically long) filename. These are normally as follows (be sure to locate them all):

- The target source itself. This will normally be the largest file.
- The phase calibrator (PH-cal). This is normally a source close on the sky to the target, which is bright enough to calibrate the telescope gains, each of which is a complex number with an amplitude and phase. These vary with time, in particular the phases which can rotate on timescales of minutes. The default observing cycle consists of 3 minutes on the calibrator, followed by 7 minutes on the target, repeated for the length of the observation.
- One or more observations of an absolute flux calibrator (FX-Cal). Normally 3C286 (1331+305) is used for this purpose. 3C286 has a well-known flux density, and is not variable, but is heavily resolved by e-MERLIN so that only the shortest baselines contain the full flux. However, L- and C-band models are available in the standard distribution of files on the e-MERLIN data reduction webpage, so that all telescopes can be calibrated.
- One or more observations of a bright point-source calibrator (PT-cal). The default sources are 0555+398, OQ208 (1407+286) and 2134+004. These are unresolved by e-MERLIN and thus have the same flux on all e-MERLIN baselines, although they are slowly variable. Their main uses are in calibrating the bandpass response. Narrow-band spectral line obesrvations usually use 3C84 and/or 3C283.
- One or more observations of a zero-polarization calibrator (ZEROPOL-Cal). This will nearly always be 3C84 (0319+415). This is used in calibrating the instrumental polarization offsets if you are interested in polarization imaging. You should also have observations of an absolute position angle calibrator (ANGPOL-Cal), which will nearly always be 3C286 (1331+305).
- Very bright QSO such as 3C84, 3C273 are sometimes used as high spectral resolution bandpass calibrators. They are quite variable and slightly resolved (at the sensitivity of narrow spectral channels).

1. Obtaining your data

- Please consult your local e-MERLIN contact person (or Rob Beswick) in advance of when you intend to reduce your data. This is needed in order to identify and reserve computing resources needed for data processing. This e-MERLIN contact person will then make your e-MERLIN data available for copying. Data will take some hours to transfer and you should double check that you are only transferring the data files you need. Individual data directories can contain data from multiple projects.

In the future an archive system will be implemented, so that users can automatically locate data observed for their projects.

- Note 1: Local access (for designated local users with access privileges: see Rob Beswick if you think this is you!) Data can be accessed using the data finder on the e-MERLIN online system 130.88.9.19:4, for which a vnc password is needed. Once exported, data can be copied by scp to an e-MERLIN offline machine.

2. Loading data from local disk into AIPS

Individual project data files should be organised on disk so that the fits files are placed in a single or multiple directory per data-stamp of observation as recorded by the archive directory. We recommend that this should mirror the structure found on the archive, for example,

```
/scratch/kria_1/USER/myproject/20120501_1/fits

/scratch/kria_1/USER/myproject/20120501_2/fits
```

Also, you should create a folder for writing and reading plots and FITS files, which is the same as that specified in the pipeline input file (if no directory is specified, it will write all plots to the current working directory). The e-MERLIN pipeline will generate useful plots and information relevant to your data and will store them in this directory. At this point, the data can be loaded (including averaging if desired). The e-MERLIN pipeline script will perform the operations of loading and averaging data; it is advised that these steps are done using the latest version of the eMERLIN pipeline. The current version is v0.7, available as a tarball together with other useful scripts for data examination and processing. Download the latest tar bundle containing the eMERLIN parseltongue pipeline and unpackage this into a sub-directory in your working area. You should also have all fits files for your data available on a visible disk (ideally on the machine you are about to use for processing) and you need to have write permissions in the base directory.
If you have some IFs in a different (spectral line) configuration from others, you can use the pipeline for the initial loading etc. of the data described here in Section 2, and you may decide to concatenate the data but (as of 2015 Feb) you must separate the different configurations before any averaging, running SERPent or calibration.
Begin by editing the example `doall_inputs` file which is included in the tarball. The key items that must be set are as follows:

- Set all target/phase reference/flux calibrator/bandpass source names. Use the filenames together with the instructions in section 0.4 to identify these. If this is not clear, ask your e-MERLIN contact person. If you are using the pipeline only to load data, these source lists are not used.
- Set AIPS disk and user number - the AIPS catalogue should be empty!
- Set `fitsdir` (location of fits data to be read)

Next you should set the main control parameters for the pipeline. The pipeline can be run incrementally so each step can be turned on or off. To load the data and make some diagnostic plots, you first need the following settings

- `doload = 1` # 1 = execute, 0 (or anything other than 1) = do not execute
- `dosortfirst = 1` # Sort the data first (see below)
- `doflagmask = 1` # Applies a flag mask of known bad frequencies due to RFI
- `doavg = -1`
- `dodiagnostic1 = 1` # execute simple plotting before flagger - (FG=-1)
- `doflag = -1` # Flag with SERPent (slow)
- `dodiagnostic2 = -1` # execute simple plotting applying flagger result
- `doconcat = -1`
- `docalib = -1`

Then on the command line type to execute the pipeline:
`parseltongue eMERLIN_pipeline.py doall.inputs`
The input values you have chosen will be printed and you will be asked to confirm your intentions before continuing. Once the script is running some logging information and feedback will be printed to the screen and a .log file will be created in your directory which will contain all of what would normally be printed to your AIPS message window. Additionally the script will time how long it takes and at the end return this execution time to the command window, along with a summary of your final AIPS catalogue (PCAT).
The suggested inputs above will load and sort the data, and apply a default flagmask to remove known bad channels. This default flagmask is available on the website and should be present in your current directory. It will also provide some diagnostic plots on disk.

- The DOLOAD part of the pipeline does the following:
  - Checks the data directory for fits files
  - FITLD: Loads all .fits files in this directory (or series of directories, more than one can be specified)
  - RENAME: Changes the name of the file in AIPS to have outnam = source name in SU table (up to 12 characters)
  - UVSRT : OPTIONALLY, sorting of files to TB order (if DOSORTFIRST = 1)
  - INDXR : re-indexes the files
  - UVFLG is used to flag the MKII-Lovell baseline, writing FG table 1
  - Prints the contents of the catalogue to the screen

This recommended procedure (`doload=1`, everything else =0) is useful in case, for example, one of your datasets may be out of order: you can load everything, check the data, and sort what needs sorting, before you run the rest of the stages of the pipeline (or separating narrow-band and wide-band data, if mixed-mode IFs are present).

3. Flagging and averaging the data

Following the initial loading/averaging stage, bad data should be flagged. This is usually done in multiple passes:

- Manual editing of large stretches of data where a telescope is missing.
- Running of the automatic spectral flagging algorithm, SERPENT.
- Manual checking for bad data that has been missed in steps 1 and 2.

3.1. Initial manual editing

Following loading, your data will be in a series of single source multi files within AIPS. At this point your data is not calibrated at all and will probably require significant delay calibration especially on long baselines, and (especially at L-band) may have significant strong RFI and spectral bad data. It is wise, before undertaking any further calibration or editing to remove any significant chunks of time where your telescopes are either off source or not observing for some reason. The reason is that otherwise you risk the autoflagger deriving invalid correlator count levels for individual baselines which can result in the removal of good data.

3.1.1 Plotting the data

Because the data contain multiple dimensions (time, frequency, baseline number and polarization) there are a number of ways of displaying the data using AIPS tasks. In particular, POSSM collapses data in the time domain to plot amplitude and phase as a function of frequency and baseline; VPLOT collapses data in the spectral domain to plot amplitude and phase as a function of time and baseline; and IBLED interactively edits data in the amplitude and phase vs. time domain, allowing you optionally to edit all stokes and all IFs simultaneously. If averaging in the spectral domain, be careful not to over-average; the reason is that delay offsets will introduce phase shifts that vary with frequency, causing decorrelation when averaging spectrally. Finally, SPFLG plots data as a greyscale for each baseline, with frequency on the x-axis and time on the y-axis (so that RFI appears as vertical stripes). It is possible to edit interactively using SPFLG. Interactive help within AIPS on all these tasks can be obtained with the EXPLAIN command, but for further help on SPFLG see section 5.2.
The best place to start is probably to run the `megapossm.py` script which is distributed in the tar file, using the phase calibrator. Instructions for running this script, which is based on a series of command-line arguments, are given in the script. This script produces a table of plots, each one derived from POSSM, at various times during the dataset and on a sample of baselines chosen so that each telescope appears twice. A sample set of plots is shown below. The pipeline also produces a set of POSSM and VPLOT diagnostic plots (dodiagnostic=1 option); these are stored in the PLOT directory. The amplitudes of uncalibrated data amplitudes are between a few 100 μJy and about 1 milliJy.
Note the following:

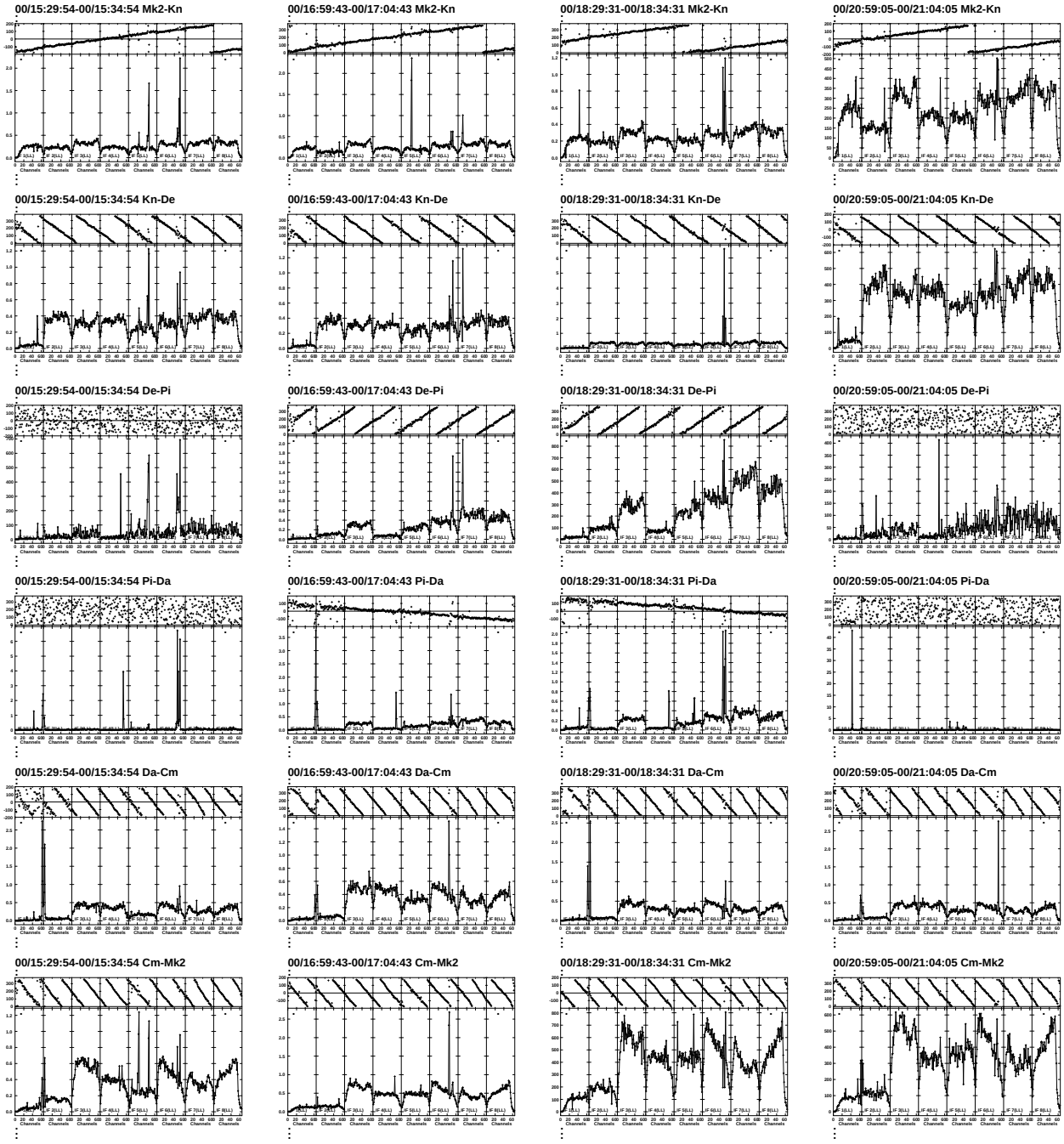

Figure 1: Set of plots for uncalibrated data. Each column of plots represents a different time, and each row corresponds to a different baseline (e.g. Mk2-Knockin in the top row). Within each plot, amplitude (bottom) and phase (top) are displayed as a function of frequency, where the overall band is divided into eight IFs. Note the phase variation as a function of frequency, which will be removed with fringe fitting in section 6. Note also the effect of telescope dropouts (e.g. Defford in the first time interval), and the sharp spikes due to RFI.

- Each plot contains information about amplitude (below) and phase (above). The x-axis of each plot is frequency, divided into the eight IFs (AIPS-speak for sub-bands). You can see the shape of the bandpass in amplitude, and also the variation of phase with frequency caused by delays within the system.
- The script takes by default seven scans on the phase calibrator evenly spread throughout the observations, and the time of each scan is given at the top of each plot together with the names of the telescopes forming the baseline.
- This plot is made using the phase calibrator, so you should see coherent behaviour throughout as the source is bright. The fact that this is not the case on the Pickmere baselines means that the Pickmere telescope was not giving good data at certain times. You should investigate this with further POSSM and/or megapossm plots and remove Pickmere at these times using `uvflg` in AIPS (see next section for an explanation on how to use `uvflg`).
- RFI shows up as sharp spikes in the data and must be removed. This will in principle be done by the autoflagger in the next step, although you can edit data using `uvflg` if you notice consistently bad channels.
- As well as examining the phase calibrator, you should also examine the flux and bandpass calibrators using `POSSM` and/or `VPLOT`. Systematically bad data on any telescope must be flagged using `uvflg` before proceeding.
- In `VPLOT`, you must set the `bchan` to an unflagged channel, e.g. if you have flagged the first 10 channels, set `bchan 11`.

3.1.2 Preliminary flagging of large blocks of data, using UVFLG

`uvflg` flags large blocks of data, keeping a track of the flagging and the reason why you flagged it. In case you realise later-on that you flagged good data, you can use your `uvflg` command and invert them to remove your flags. Use `uvflg` as much as possible, for ease of undoing/redoing and applying phase-ref flags to target.
`uvflg` needs to know the number of the telescopes which are to be affected; you can find this out using the task PRTAN or from the table below:

.Lovell	.Mk2	.Knockin	.Defford	.Pickmere	.Darnhall	.Cambridge
1	2	5	6	7	8	9

Example 1: to flag BL 2-6 / IF4-5(LL) between t = 0/23:30:00 and t = 1/01:15:00, for the PT-Cal, you would:

- run `uvflg` with `getn[ALL.DBCON.1]`; `sources='[PT-Cal]'`; `antennas 2,6,0`; `baseline antennas`; `bif 4`; `eif 5`; `bchan 0`; `echan 0`; `stokes'LL'`; `timerang 0,23,30,0,1,1,15,0`; `opcode'FLAG'`; `reason 'BAD TIME'`; `outfgver 1`

Here, and elsewhere, you should replace [PT-Cal] with the name of your point calibrator.

Example 2: Flagging end channels.

Inspect the POSSM plots for a good time on a bright PT-Cal on a typical baseline, e.g. Mk2-Kn in Fig. 1. You will see that for most or all IFs, the first and last few channels have lower sensitivity. You may want to zoom in by running `POSSM` interactively on the first and last segments of each IF. Note the channels at each end which reach about 90% of the average of the good channels, and have coherent phase. All sources, antennas and polarizations should behave the same, but IFs may behave differently, in particular with more bad channels at the bottom of the lowest IF and the top of the highest. If you set a `reason` you can undo the flags if you feel you have been too brutal.
If you are operating on data averaged to 128 channels per IF, typical settings for the first two IFs might be as follows:

- start with `default uvflg` to initialise parameters.
- run `uvflg` twice for each IF:
  - `getn[ALL.DBCON.1]`; `bif 1`; `eif 1`; `bchan 1`; `echan 10`;`reason 'StartIF1'`; `outfgver 1`
  - `bch 124`; `ech 128`;`reason 'EndIF1'`
  - `bch 1`; `ech 5`;`reason 'StartIF2'`
  - `bch 124`; `ech 128`;`reason 'EndIF2'`
- Repeat for all IFs with suitable channel selections.

If you still have separate files you can run the same selection for each IF in all files taken in the same configuration (but check if they are separated by many weeks). If you have not yet averaged channels i.e. you still have 512 per IF, you will flag more channels, and you should flag exact multiples of 4, if you will eventually average by 4 to 128 channels, or 8 if you eventually average to 64 channels, etc. In `VPLOT`, you must set the `bchan` to an unflagged channel, e.g. if you have flagged the first 10 channels, set `bchan 11`.

Example 3: to flag antenna 6 / IF2(ALL POL) / channels 20-40 at all times, for the PT-Cal, you would:

- run `uvflg` with `getn[ALL.DBCON.1]`; `sources='[PT-Cal]'`; `antennas 6,0`; `baseline 0`; `bif 2`; `eif 2`; `bchan 20`; `echan 40`; `stokes'FULL'`; `timerang 0`; `opcode'FLAG'`; `reason 'BAD CHANNELS'`; `outfgver 1`

To remove a flag, just use the same `uvflg` inputs and change the opcode to 'UFLG'. Sometimes, when the Lovell is present in the array, it is kept on target every other observation scan on the PHS-Cal. Thus, there should be one PHS-Cal scan every 20 min that has bad data for the Lovell. Use IBLED and your scan file (see Section 5.1.b) to determine the bad time ranges and run `uvflg` iteratively with the successive time ranges:

- run `uvflg` with `getn[ALL.DBCON.1]`; `sources='[PHS-Cal]'`; `antennas 1,0`; `baseline 0`; `bif 0`; `eif 0`; `bchan 0`; `echan 0`;`stokes'FULL'`; `timerang [bad range]`; `opcode'FLAG'`; `reason 'BAD SOURCE'`; `outfgver 1`

Notes:

- **Important note about flags.** AIPS has upper limits on the number of flags which can be handled, which is in general task-dependent. If you exceed this number (normally in the thousands) you may find that the flags are not handled correctly. **It is therefore worth keeping an eye on the number of flags in the flag table** (using the task PRTAB with `inext='FG'`; if this exceeds a few thousand, you should run UVCOP (which is able to deal with a very large number of flags) to write a new dataset, applying the accumulated flags with `flagver=n`, where `n` is the FG table in the file header which you have built up.
- If you apply any time- or channel-dependent flags based on examination of the phase calibrator (which affect a significant fraction of a scan or more), you should usually also apply them to the file containing the target.
- After you have manually edited these major stretches of time from your data, APPLY these flags to the data using either UVCOP or SPLAT. Then you should tidy up your AIPS directory so that only the data files you wish to continue with are present.

3.2 Automatic RFI flagging of data

SERPent is an automated flagger which has been developed by the UCL group in collaboration with staff at Onsala, MPIfR, ASTRON and JBCA. This flagger runs a ParselTongue version of the AOflagger algorithm developed for the LOFAR telescope (Offringa et al. 2010, MN 405, 155). The current application of this code can be run either on individual files (with AIPS) or on multiple files simultaneously in order to utilise multiple CPUs available on current machines. You should check that you have the SERPent scripts in your directory before using `doflag=1`. **Note that for C-band data, you rarely need to run SERPent, and the manual inspection described in section 3.1 will often suffice**.
To average the data and then run the autoflagger using the pipeline, the following settings are recommended

```
doload = -1        # Already done in previous section
dosortfirst = -1   # Already done in previous section
doflagmask = -1    # Already done in previous section
doavg = 1
dodiagnostic1 = -1 # execute simple plotting before flagger - (FG=-1)
doflag = 1         # Flag with SERPent (slow; see section 3.2 for further discussion of this option)
dodiagnostic2 = 1  # execute simple plotting applying flagger result
doconcat = 1
docalib = -1
```

After running the script the second time, your AIPS catalogue should look like this:

```
AIPS 1:                          31DEC13 AIPS:
AIPS 1:      Copyright (C) 1995-2012 Associated Universities, Inc.
AIPS 1:             AIPS comes with ABSOLUTELY NO WARRANTY;
AIPS 1:                  for details, type HELP GNUGPL
AIPS 1: This is free software, and you are welcome to redistribute it
AIPS 1: under certain conditions; type EXPLAIN GNUGPL for details.
AIPS 1: Previous session command-line history recovered.
AIPS 1: TAB-key completions enabled, type HELP READLINE for details.
AIPS 1: Recovered POPS environment from last exit
>pcat
AIPS 1: Catalog on disk  1
AIPS 1:  Cat Usid Mapname      Class   Seq  Pt     Last access      Stat
AIPS 1:    1 4654 0639+0553   .UVDATA.    1 UV 20-NOV-2013 15:29:26
AIPS 1:    2 4654 0645+0541   .UVDATA.    1 UV 20-NOV-2013 15:30:16
AIPS 1:    3 4654 1331+305    .UVDATA.    1 UV 20-NOV-2013 15:35:36
AIPS 1:    4 4654 1407+284    .UVDATA.    1 UV 21-NOV-2013 11:14:21
>
```

Notes:

- Before averaging in time, you might want to check the integration time. PRTUV gives the amplitudes, phases and weights of all the polarisation products for every baseline and every integration time. Run `prtuv` with `getn[ALL.DBCON.1]`, `cparm(9)=N`, `docrt=132`; where N is a number formed as 100A+B, A is the number of the first telescope in the baseline and B the number of the second (see section 3.1 for the key to the numbers).
- If averaging data, set:
  - `nchan` to the desired number of channels per sub-band: see the notes in the paragraph below for details of how to set this. If no frequency averaging is required set to 512, i.e., the number of channels in the original data (-1 also works for this purpose). +By default, we recommend to use 128 channels / sub-band, to optimise data resolution and computational time.+
  - `tint` to the desired time averaging in seconds. +If no time averaging is required set tint = -1 (default). The time between successive time stamps then corresponds to the integration time: 1s (by default).+
  - dosort = 1 : to TB sort the data after averaging
- The safe averaging levels are set by consideration of the field of view (FOV) you wish to image. For the spectral averaging, your field of view will be limited to the resolution (50 or 160mas for C- and L-band), multiplied by the frequency, divided by the width of the output channels. For example, at C-band each 128-MHz subband is divided into 512 spectral channels of 0.25MHz; averaging this to 64 channels per subband will give channel widths of 2MHz and a FOV of 50mas x (5GHz/2MHz), or about 2 arcminutes. For the time averaging, the rule of thumb is that if you average the data to n second integration times, your resulting FOV will be about 13000/n resolution elements. However, you should a) probably use a safety margin of a factor of 1.5-2 over these numbers if you wish to ensure reliable imaging of the whole FOV, and b) note that the required FOV may be larger than the extent of your target; particularly at L-band you may need to image, and remove by peeling, other sources within the field. If you wish to image the full primary beam then you will probably not be able to average the data. Averaging the data reduces its size and speeds up subsequent processing.
- After the data have been averaged, the original, unaveraged, AIPS data are deleted to conserve disk space.
- The DOAVG part of the pipeline uses SPLAT to split out the data files out applying:
  - Spectral averaging of data to provide a final number of channels equal to the input `nchan`.
  - IF `TINTO`, time averaging is carried out at the same time. You can find out what integration time has been used by printing out a section of data using the AIPS task PRTUV.
  - Smoothing (if selected in parameters SMOOTHa,b,c in doall_inputs) NOT APPLIED BY DEFAULT
  - A range of sub-bands is extracted between `SBANDL` and `SBANDU` - DEFAULT IS ALL SUB-BANDS
  - Applies any flag tables attached to file at this stage
  - deletes older files.
- The DOAVG part of the pipeline runs UVFIX to re-calculate the u, v, w coordinates. If `DOSORT 0`, UVSRT is used to TB sort data. INDXR is then run on data.
- In a few cases, the phase gradient across the bandpass can be extreme enough that averaging over a significant number of channels destroys coherence. This usually only happens on baselines to Cambridge. To see if this is a problem, run POSSM on a small segment of data on a bright source (e.g. a phase calibrator) on one or two Cambridge baselines. If it does appear that the phase difference is large across the frequency over which you wish to average, you will need to fringe-fit the data (see section 6) before averaging.

- By default the pipeline flags the first minute of data from each scan. If you want to do this manually or remove more data, inspect one or two of the best (non-Lo) baselines on the phase-ref, averaging the central 10-50 channels. You can use IBLED or VPL; check about 1 hr of data at a couple of times during the observations. First, do this with flagver=-1. This will show whether it was correct to flag the first 1 min of each scan. If this is too much, delete the flag table; if too little, add to it using the task QUACK. Note that the QUACK interval is from the beginning of the scan regardless of previous flagging.
- Initially make sure that your AIPS catalogue only contains relevant UV files that you wish to run through the autoflagger. Additionally if you have any existing AIPS tables that you wish to apply (e.g. FG table) make sure these are applied to the data and/or copied (TACOP) to another AIPS disk/number or scratch area.
- **If you have done your own flagging first, be aware that SERPent destroys existing flag tables!** You should apply them first to your data file using UVCOP, producing a new u-v data file to which to apply SERPENT.

3.2.1 SERPent usage

The SERPent software can be obtained directly from http://www.ucl.ac.uk/star/research/stars_galaxies/cobras/technical/rfi as well as documentation detailing the algorithm used and the current recommended parameters.

SERPent is controlled by five main parameters which are listed below along with the current recommended values:

```
aggressiveness_first_run = 10
max_subset_first_run = 2
aggressiveness_second_run = 10
max_subset_second_run = 2
kickout_sigma_threshold = 2.75
```

In general, the main parameter for controlling the aggressiveness of SERPent is the `kickout_sigma_threshold`. This parameter prevents SERPent from flagging data that lies within this threshold level. The default value is 3.0, and lower values give deeper/more aggressive flagging. The majority of baselines work well using values between 2.5 and 3, with 2,75 providing a good compromise. However, some baselines tend to be inherently noisier (especially those to Defford) and may require slightly more aggressive flagging.

The level of observed RFI within the data will also depend on the brightness of the source. In some circumstances a faint target source may require generally more aggressive flagging (or repeated runs of SERPent) in comparison to the brighter calibration sources within the same dataset.

SERPent can be computationally expensive to run especially on larger datasets, so checking that the parameter settings are suitable before performing a full run is advisable. This can be done in one of two ways:

- Use a low aggressiveness run on all baselines and follow this with further runs on selected baselines if necessary.
- Use a small sample of data to test and alter appropriately the parameter settings before running SERPent on the full dataset.

- Further notes for enthusiasts:
  - (DOFLAG) Note that REFLG is an AIPS task that is only available in AIPS 31DEC2012 and newer.
  - (DOFLAG) TBIN has recently been altered to include a large number of Table entries, which is needed. This is in AIPS current MIDNIGHT JOB.
  - (DOFLAG) The SERPENT flagger is likely to create a very large number of individual flag entries. This can be sometimes (for large files) beyond the numbers which can be read into, or handled by AIPS. This can result in a loss of flags created when they are applied to data.
  - (DOFLAG) The flagging operation is VERY computationally intensive on large data sets, and you should expect it to take a number of hours.
  - Running the flagger multiple times on files can produce better (more reliable) flagging results. This is useful but very time consuming.

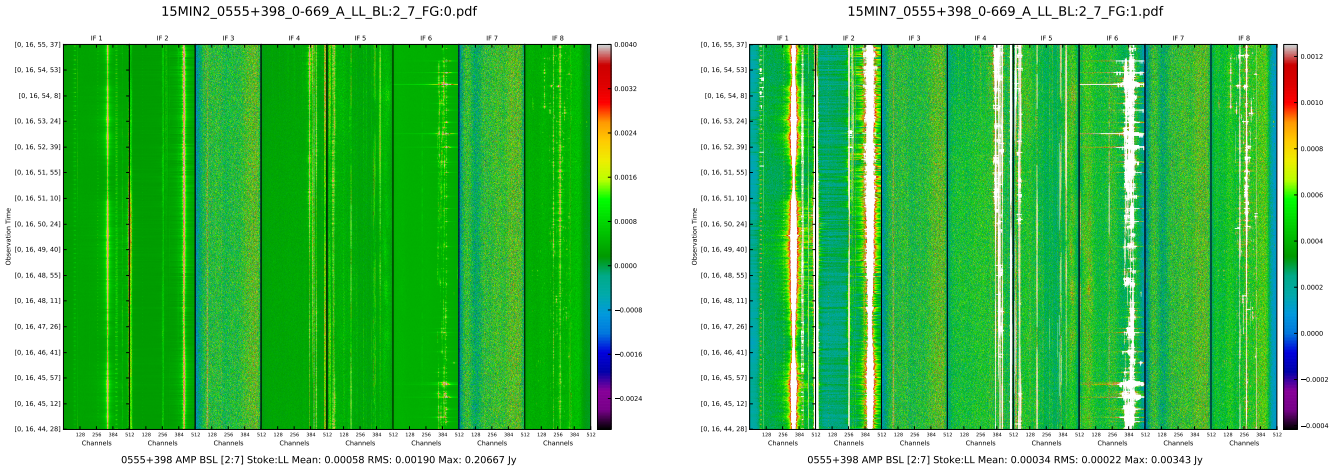

Figure 2: An example of e-MERLIN L-band data from before and after a SERPent run. Each image shows a single baseline over a short time-range for one stokes parameter. Each of the 8 IFs are plotted consecutively across the x-axis with time shown along the y-axis.

3.3 Secondary Manual editing

You should then check the output files from the autoflagger in the same way as for the first stage of editing. The plot below shows the same data, with appropriate flags applied. Note that the RFI is nearly all gone, and that the bad data on Pickmere baselines has been removed. The phase slopes across the bandpass remain, however, as the data have not yet been fringe-fitted. Once you create a new final FG table you should run the task REFLG to condense all flags.

4. Concatenation

At this stage the pipeline should have (with the `doconcat=1` from section 3.2) concatenated into a single file, `ALL.DBCON`, the files corresponding to the individual sources. During the execution of this section of the pipeline, the aips task `quack` should have been run on the final combined file.

- Specifically, the DOCONCAT part of the pipeline:
  - a) Runs TASAV on all files and backs these save files to your scratch disk;
  - b) Applies the highest FG table on each file, using UVCOP;
  - c) Uses DBCON to concatenate all UV-files in AIPS catalogue;
  - d) If DODELETE=1 all original files will be zapped to leave one concatenated multifile, if DODELETE=-1 the pipeline will leave UVCOPed files (ie. with flags applied) in AIPS Catalogue for manual deleting after checks;
  - e) Deletes CL#1 and re-creates it with INDXR, ensuring all sources have entries in the CL table.
  - Following this step, the rest of the pipeline does not care what files are in your AIPS catalogue, as long as there is one file labelled ALL.DBCON.1 - this is the file it expects, and knows what to do with. If, for some reason, the file is called something else and there is more than one file in your catalogue, the script will fail telling you that it does not know what to calibrate.

At this stage, you can run some diagnostic tests, using the following set of parameters. You should in any case run the pipeline again with `dosetflux=1`, as this imports flux densities for the primary flux calibrator, 3C286.

```
dodiagnostic3 = 1  # makes simple diagnostic plots pre-fringfit
dofringfit = 0}    # fringe-fit (done manually in section 6)
dosetflux = 1      # Calibrate the flux of 3C286
dofluxcal = 0      # Calibrated flux scale relative to 3C286
dotherest = 0      # Development section of pipeline (not to be run)
```

5. Confirmatory checks and editing

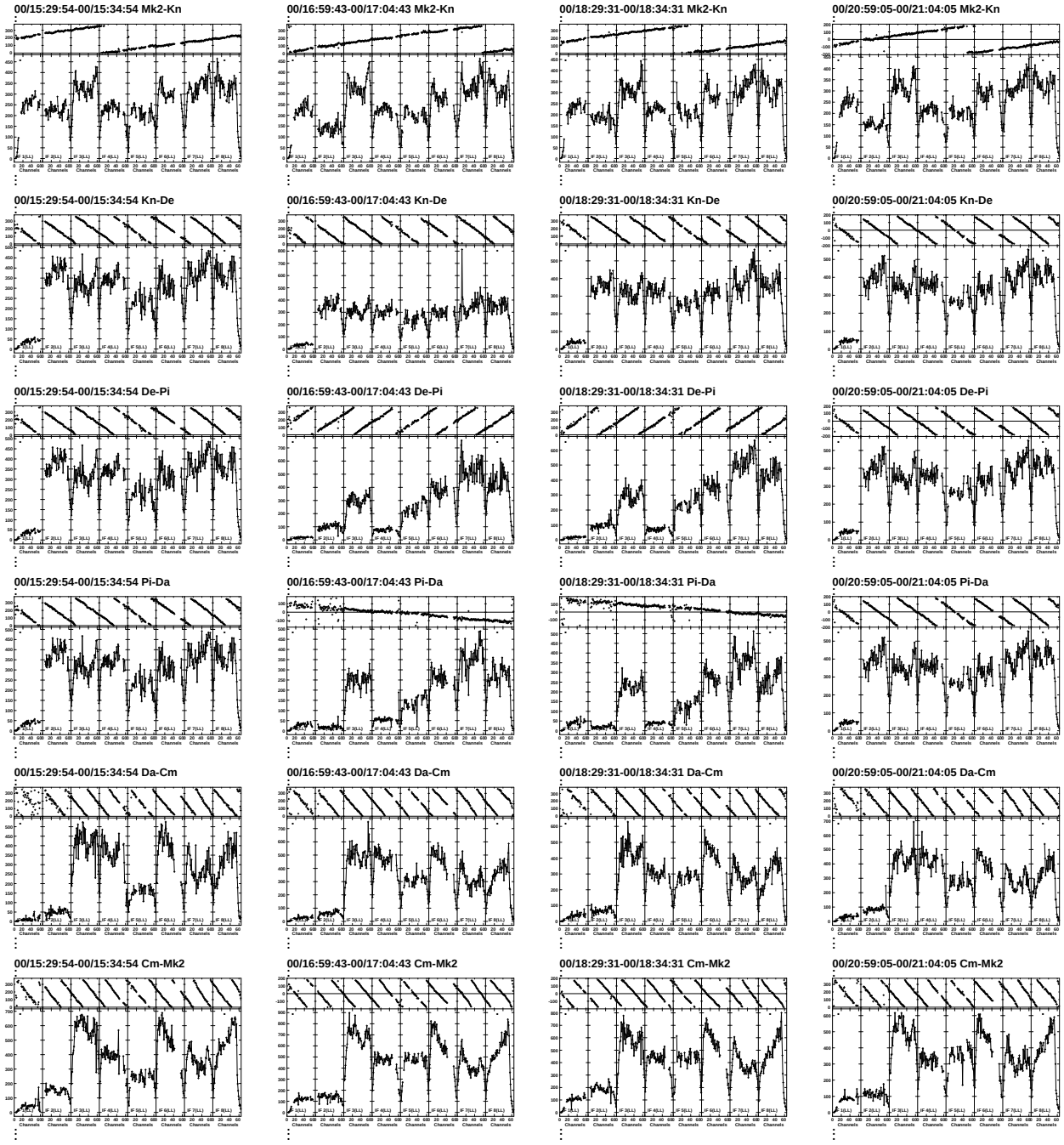

Figure 3: Multiple spectral plots of corrected data

This section describes the editing procedure which you may wish to follow on the multi-source file. If you are absolutely happy that the data have been appropriately flagged, you can skip section 5.2 and proceed straight from 5.1 through to section 6.

5.1. Initial checks

5.1.1 CHECK your catalogue

Use the command pca to print the content of your catalogue of files. You should have a multi-source data file from the previous processing called ALL.DBCON.1 which contains the target and the calibrator sources (FX-Cal, PT-Cal and PHS-Cal). It assumes the observations are continuum observations made with 8 IFs and that channels were averaged such that each IF contains 128 channels (standard configuration for L-Band continuum observations). Update numbers of IFs and channels where appropriate throughout the cookbook.

- Notes:
  - The procedure for spectral line observations and continuum-spectral mixed mode will be released soon;
  - This recipe does not allow calibration of data for which the PT-Cal or the PHS-Cal are resolved by the longest baselines.

5.1.1 CHECK your multi-source file

If you have observed the same source multiple times, you may find that the source table has been written in such a way that multiply-observed sources are written multiple times in the source table, but each time with a different value of the `QUAL` parameter. If this is the case, the source table must be corrected before proceeding further. To check this,

- Run `PRTAB` with `inna='ALL'`, `incl='DBCON'`, `inext='SU'`

Inspect the values in column 3 (`QUAL`). If you have any source listed multiple times, each with a different `QUAL`, then

- Run `DQUAL` with `inna='ALL'`, `incl='DBCON'`.

This may take quite a while, since to correct the problem a new data file is written. You may wish to rename this file back to `ALL.DBCON` before proceeding; check the SU table again to verify that the multiple `QUAL`s have disappeared. Following concatenation it is wise to check your data structure and check that you are happy with all of the flagging that has been made. Note that your data have now had any flags you have made integrated into the data files and the flags are therefore irreversible (unless you go back to the previous step). Additional flagging may be needed, especially since flags have been made on single source files and hence time segments may have been removed on calibrator sources and may have been left on fainter target sources where you are unable to see signal on individual baselines. Use LISTR to print some information about the content of your multi-source file (review section 0 if necessary to translate the following instruction into what you should type):

- Run `LISTR` with `inna='ALL'`, `incl='DBCON'`, `inseq=1`, `optyp='SCAN'`, `docrt=-1` and `outprint='DATA/scan.txt`
- Notes:
  - `DATA` should be a directory which you have previously defined using `setenv`. If you have not done so, exit AIPS, do the definition (e.g. `setenv DATA /scratch/mydir`) and restart AIPS.
  - Rather than setting `inna`, `incl` and `inseq` individually you can inspect your catalogue with `pca` and then type `getn X`, where X is the catalogue number of your ALL.DBCON.1 file. This process will henceforth be denoted getn[ALL.DBCON.1].
  - Note the missing ' at the end of the outprint variable. It allows the use of lower cases for filenames.
  - The variable "outprint" may accept only a limited number of characters, so that you might need to use an environment variable to specify the PATH. Example using the MYAREA environment variable: outprint 'MYAREA:scan.txt
  - If you want to print the result in a terminal, use docrt 132. But, in that case, you'll first need to broaden the width of your terminal to see the whole content of your scan file.
  - Check that all the calibrators are present; that their frequencies and spectral resolutions match that of the target; that you have the expected cycle on phase-cal and target (usually 3/7 minutes), and that all the data you expect are there.

The pipeline automatically generates a flag table, FG 1, attached to your DBCON file. In all cases, this table contains flags corresponding to the 1st min of each scan of your observations (for all sources). If you used SERPent, the table also contains flags corresponding to identified RFI. If you didn't use the auto-flagger, you don't necessarily want to apply the FG table, as the 1st minute of each scan may be not necessarily bad. Delete this flag table, (you will flag the initial times of each scan manually later on) by running `extdest` with `getn[ALL.DBCON.1]`, `inext'FG'` and `invers=1`. The automatic flagging of the 1st min of each scan represents a significant cost. It will be removed in the new version of the e-MERLIN pipeline and replaced by an algorithm that dynamically determines the time to remove for each antenna.

5.2 Further editing

5.2.1 Overview of the whole data set

SPFLG displays your data in the frequency VS time domain, baseline per baseline. Use the interactive menu to select your options (click mouse then press "A" to apply). The time over which the data are averaged is set by dparm(6).

- Run `spflg` with `getn[ALL.DBCON.1]`, `docat=-1`, `dohist=-1`, `stokes='HALF'`, `outfg=1`, `dparm 0`

It is likely to be most efficient to run this multiple times, once for each source (`source=[PHS-CAL]` for example). If you want to set a time interval on the Y axis of anything other than the default (10s) use `dparm(6)`.
In the interactive mode proceed as follows:

- Inspect the amplitudes of all baselines: "LOAD NEXT/LAST BASELINE" + A.
- Don't forget to inspect BOTH POLARISATIONS. You can switch between them by using "DISPLAY STOKES LL" / "DISPLAY STOKES RR" + A
- You can apply math functions to the data in order to improve the contrast (useful when you use several sources at a time): "LOAD LOG", "LOAD SQRT", etc. + A
- Identify any RFI (if you did not use SERPent to remove them) and flag them for ALL SOURCES. For that purpose, make sure that: 1) "ALL-SOURCE" is displayed in the bottom part of the screen. This should be the case by default. In case "ONE-SOURCE" is displayed, use the switch "SOURCE FLAG" in the interactive menu. 2) you flag both polarisations. To do so, click "ENTER STOKES FLAG", press A, type "FULL" in the terminal then press ENTER. 3) you use the "FLAG CHANNEL" command (or "FLAG AREA"). To flag data: click the command, press A, select the channel/area you want to flag, press C. Press D to come back to the interactive menu. The standard PT-Cal sources and 3C286 have negligible spectral features in our bands/resolutions. So spectral spikes are RFI or other artefacts.
- Identify any BAD IFs, BAD TIME RANGES or BAD SCANS. Just note the baseline/antenna, IF, polarisation and times involved for later flagging with UVFLG. You'll especially want to get rid of the early times on -CAL, when some of the antennas may be not pointing at the source while the observations began. These times are easy to identify: the amplitude is almost 0. Phase and amplitude of astrophysical sources change smoothly as a function of time. Any abrupt change is an artefact. Short drop-outs/spikes in time (few min or less) should be flagged. Smooth changes with time may be source structure (not for the PT-cal) and will be faster on longer baselines.
- At C-band, Defford is less sensitive that the other antennas, so do not be worry if intermediate-length baseline data looks noisy, as long as there is some coherence.
- Look at the phases as well: if they become noisier and the amplitude drops, this may be bad weather or low elevation (e.g. at the beginning/end of a run). For the PT-cal, flag as long as there is enough data left (more than 10-15 minutes is recommanded). For the PHS-ref, throwing away 10-20% of data on a particular antenna may be better than adding noise. However, for short runs/low-dec sources, it might be better to leave the data in (at least initially) as long as the phase is visible.
- If a significant amount of data look bad with very low amplitude and/or noisy phase, check with less averaging and check in the other dimension. For instance, if you have a high delay, and average many channels, the data will look like noise when plotting as a function of time, but they can be retrieved by delay calibration.
- If the phase-reference is bad for a single scan or longer, flag the intervening target data as well (unless the target is bright enough that you can inspect and flag it separately).

- A very useful plot for data inspection can be generated by plotting the phase (DISPLAY PHASE followed by LOAD) on the phase calibrator, for Lovell baselines (ANTENNA=1) and using pseudo-colour (TVPSEUDO). This allows instant visibility of which data is and is not phase-coherent, and works well provided that your phase calibrator is stronger than about 100mJy.

5.2.2 Flag small blocks of data

Use SPFLG on each calibrator source independently to get rid of small chunks of data. We do not edit the target data yet because the calibration process may fail for some IFs or timeranges resulting in significant flagging of the target data.

- Run `spflg` with `getn[ALL.DBCON.1]`; `sources='[FX-Cal]'`; `docat=-1`; `dohist=-1`; `outfg=1`; `dparm(6)=1`

and repeat for the other calibrator sources.
Notes:

- You can also use UVFLG if you missed significant chunks of bad data.
- dparm(6) sets the time to which the data are to be averaged. Sometimes, using dparm(6)=1 can saturate the memory. This results in SPFLG diplaying erronous data for the last baselines that are loaded into memory (6-7, 6-8, 6-9, 7-8, ...). Just increase the value of dparm(6) to lower the amount of data: 1 - 5.
- If you pick an averaging time (dparm(6)) which does not divide into a scan (e.g. scan of 100s, averaging 30s) some of the displayed data at the end or beginning of each scan will appear noisier because it has fewer samples (10s in this example). If this happens, choose a shorter averaging time (set timeranges to do the data if necessary).

6. Initial calibration procedure: delay correction and flux density scale

Strategic advice: The pipeline contains options for running calibration automatically (docalib 1) which does sections 6 onwards for you. You may wish to try this, but bear in mind that the decisions that need to be made from this point onwards are likely to need human input, and the map you get by following sections 6-10 manually will almost certainly be better than the automatic output. Hence, the current recommendation is that you should follow steps 6.2 onwards manually. If you have 3C286 as your flux calibrator and you are sure that your PT-Cal is a point (OQ208 or 2136+004), follow step 6.1 as outlined here. If you want to set flux scales manually, refer to Appendix C. Appendix C also provides some guidelines in case the PT-Cal or PHS-Cal are resolved.
Important reminder: Remember that AIPS parameters are common to different tasks in many cases, so you should always either specify the default explicitly, or review the inputs carefully to make sure that none are left over from the last task.

6.1) Check flux density of FX and (optionally) of PT-Cals in SU table (I stokes):

You need to know at least one flux density for a calibrator source. The source used is 3C286, which is heavily resolved by the longer e-MERLIN baselines. 3C286 also has a steep spectral index, so this also needs to be taken into account when setting the fluxes across a wide bandwidth. The pipeline is aware of these effects and should already have set flux densities for you, provided that your main flux calibrator is 3C286; see appendix C if this is not the case. Model images are available for 3C286 at both L and C bands, which will be scaled correctly to the flux densities set.
Make sure that this has worked using the AIPS task PRTAB:

- Run `prtab` with `getn[ALL.DBCON.1];inext'SU'`
- Notes
  - 3C286 should have fluxes listed in Stokes I which decrease with increasing frequency. If not use the `dfluxpy` script which is distributed in the data reduction package and set these values with `SETJY`.
  - 3C286 has alternative names (B)1328+307 and (J)1331+305.
  - Unknown I values should be 1 or 0 and all Q U V values should be 0. This should have been set in the pipeline as part of the flux density scale processing. **If any unknown Q,U,V values are not zero, you must reset them using SETJY with** `optyp'rejy'`.

6.2. FIT and REMOVE delays and rates on ALL calibrators

The e-MERLIN pipeline currently contains a section which will run FRING to make a nominal solution of the delays in the data. This should be run in a similar way to the sections explained above and is listed as `dofringfit`. However, it is important to inspect and edit the solutions, so manual fringe fitting is recommended.

6.2.1 FIT delays and rates on ALL calibrators

You should now assess the level of delay on individual antennas, which can be deduced from the `megapossm.py` plots previously examined. With e-MERLIN, delay offsets on individual telescopes can be up to a few hundred nanoseconds in some cases, particularly on longer baselines, and are larger at C-band than L-band. These delays can vary in time, either as gradual drifts as fibres change temperature and length (note 1 foot of length change corresponds to 1 nanosecond delay), or occasionally as sharp changes originating in the correlator. They cannot therefore be corrected on a single scan, as with JVLA data, but must be corrected for as a function of time using the AIPS task FRING. FRING fits and corrects for delays and rates (standard technique for VLBI obsevations).
If you have already flagged the end channels (as in 3.1.2 example 2) you do not need to set bchan or echan.

- Run `fring` with `getn[ALL.DBCON.1]`; `calsour'-[TARGET]'`; `snver=1`; `bchan 5`; `echan 123`; `refant 2`; `solint 5`; `aparm 0`; `aparm 3 0 0 0 2 1 2.5 0`; `dparm 3,450,0,0,0,0,0,5,1,0`

- Notes:

  - The notation `-TARGET` means to use all sources except the target i.e. all calibrators.
  - Solution (SN) table 1 will be written. This has the same effect as the default (0) which writes a new table with the largest serial number.
  - Channels 5-123 are used, excluding the top and bottom four. Change this accordingly if you have different numbers of channels per IF.
  - Mk 2 (antenna number 2) is generally used as the reference antenna, i.e., that for which phases are set to 0. Your reference antenna should be: Present throughout the observations; Located in the inner part of the array (2,7,8); Have stable phases (without any jump) throughout the observations. If all the antennas exhibit a jump at the same time, antenna 2 is probably the antenna in which the phase jump actually occurs. Opt for another reference antenna and update the refant variable throughout this recipe.
  - `solint 5` determines the solution interval, which is usually slightly longer than the interval needed to cover a phase reference scan.
  - `aparm` is an array of parameters, which here have been set to use a minimum of 3 antennas for solutions; print some information; use at least S:N 2.5 for solutions; and solve for multiband delays.
  - `dparm` is another array, here set to use 3 baseline combinations in initial fringe search; search in a window of 450ns in delay; calculate solutions for delays and phases, **but zero the phase solutions afterwards**. Occasionally we have delays greater than 450ns; if you find many solutions have failed, check the offending antenna in POSSM and, if necessary, run a higher dparm(2) or leave it blank.

Now examine the delay solutions in the SN table produced, with SNPLT on the TV with 9 plots per page:

- Run `snplt` with `getn[ALL.DBCON.1]`; `sources=''`; `optyp'DELA'`; `invers 1`; `dotv 1`; `nplots 9`

- Notes

  - You may find some significant, instantaneous jumps in delays. These jumps are due to correlator resets in the delay correction applied and can happen at any time, and may differ between polarisations on individual telescopes. If a target is too faint to self-cal, then flag the scan between phase ref solutions with a delay jump.
  - Check that delays are of a few tens of nanoseconds and that they evolve smoothly and consistently as a function of time.
  - If you see many erroneous delay solutions for some antennas / IFs, flag the corresponding data with SPFLG (see Section 5.-2), use EXTDEST to delete the SN table produced by FRING, and re-run FRING.
  - Use also these plots to confirm that antenna 2 is a good choice as the reference antenna for the phases.
  - If the point-cal solutions are mostly OK and the delay is stable in time, but many still fail or are bad on the phase-reference, you can solve for the point-cal and fx-cal only and apply the point-cal solutions to the other sources.

  - These default settings of FRING assume that there are no systematic rates (differential of phase w.r.t. time), and all the phases within a solution interval can be averaged coherently before being passed to the fitter in FRING. If more than ∼10% solutions fail or look bad, and all else has failed, try solving for rate also by using dparm(9)=0 and dparm(8)=0. If the phase changes by more than ∼10 deg in 2 min for a significant length of time, on any baselines, you will need to use the revised solution including rates.
  - If all solutions are bad for a significant time period, check whether the reference antenna is at fault. If so, chose another refant near the centre of the array.

If you notice the presence of individual outliers after the previous steps, you need to remove them manually using either the `snedit.py` parseltongue script, distributed with the data reduction package, or with the AIPS task SNEDT. In the first case, the `snedit.py` script performs an automatic edit of SN tables, and will process either delay solutions (which you want to do here) or amplitude/phase solutions (which you will need later). Instructions for use are given in the script. **Note that this overwrites your SN table**, so you may want to use TACOP first to make a copy of the SN table.
Alternatively, you can use SNEDT for more manual editing. In this case:

- Run `snedt` with `getn[ALL.DBCON.1]`; `inext'SN'`; `invers 1`; `dodelay 1`; `crowded 1`; `flagver 1`; `bif 0`; `eif 0`
- Notes
  - This will create a new SN table 2. You should examine this with SNPLT, and keep track very carefully of what is in each SN table. AIPS CL (calibration) tables are made by interpolation of SN tables, so it is important to use the correct SN tables and to use each only once!
  - In case you doubt whether some solutions should be flagged or not, you can simply opt for applying them (see next section) and figure out later on if they give well-calibrated data. However, keep in mind that conserving bad data will affect your calibration solutions and may eventually end up in major calibration errors.
  - A third alternative to `snedit.py` and SNEDIT is that you can use SNSMO with `samptyp'MWF'` to write a second, edited solution table. This usually needs some experimentation with the CPARM parameters in SNSMO; typical values are smoothing times of 5-20 minutes and maximum deviations of about 10ns.

6.2.2 Apply and check delay/rate solutions on ALL calibrators

CLCAL interpolates calibration solutions stored in a SN table. It creates a new CL table, which can be applied to different sources independently (FX-Cal, PT-Cal) or collectively (especially, to pass the PHS-Cal solutions to the target).
By default, AIPS generates CL1, which contains null entries (amp 1, phase 0) for all sources at the requested time intervals. In general in this cookbook, CLCAL will be iterated to generate CL 2, 3, 4, ... successively from SN 1 and the following SN 2, 3, ... to accumulate the corrections additively.

- Run CLCAL with `getn[ALL.DBCON.1];sources '';calsour '';opcode 'CALI';interpol'AMBG'; refant 2; gainver 1; gainuse 2; invers n;snver n`
- Notes
  - `interpol'AMBG'` uses the rate information calculated by FRING to interpolate the phases (because the rate is the differential of the phase with respect to time). This is important because it allows correct interpolation of phase when the phase is changing fast enough to leave you vulnerable to 2-pi phase ambiguities in connecting one solution to the next. If the phases are interpolated incorrectly, this is disastrous and irretrievable.
  - n is the version number of the SN table you wish to interpolate. The syntax is that SN table `snver` up to and including `invers` is used and interpolated into the final calibration. This final calibration is derived by taking CL table number `gainver` and writing out `gainuse`. The defaults for the `gain...` parameters are to use the highest extant CL table and write a new one with a higher version number.
  - `sources '';calsour ''` will take the solutions for each calibration source and interpolate them over the nearest time ranges in the calibration table. In this way, the phase reference solutions are automatically interpolated into the times covering the target scans.

This should interpolate the SN table written by FRING into a CL table, which is an AIPS calibration table for the whole dataset. At this point you should use the megapossm script again, with docalib set to 1, in order to apply the

calibration you have just derived. You should find that all the phase slopes in the data have been removed. POSSM can also be used to visualise individual baselines in your data as a function of frequency/channel. Use it on each calibrator. Press B to move to the next baselines/polarisation/time interval. Check first the mean solutions, averaged over all solution intervals:

- Run possm with `getn[ALL.DBCON.1]`; `sources'[FX-CAL]'`; `antennas 2,0`; `baseline 1 2 5 6 7 8 9`; `stokes'HALF'`; `codetype'PHAS'`; `freqid 1`; `aparm 0`; `aparm(9)1`; `flagver 0`; `docalib 100`; `gainuse 2`; `dotv 1`; `nplots 7`; `solint 0`
- Notes:
  - All baselines to antenna 2 (Mk2) are plotted by these defaults.
  - `gainuse 2` specifies CL table number 2. The default (0) can also be used to apply the highest version number CL table.
  - docalib 100 requires that the CL table (produced by CLCAL) should be applied, but without using weight information
  - aparm(9)1 combines all the IFs into one plot. Once again, remember the AIPS habit of carrying over parameters from one task to another (this is the reason for the aparm 0; you are likely to have these set from previous tasks)
  - As the delay solutions may vary with time, you may want to also check the solutions on each time interval by using solint=-1.
  - Use these POSSM-plots to note the channel range to be used for spectral averaging. The decrease of sensitivity on the edges of each IF results in a higher level of noise in the extremal channels. In order to minimize the contribution of noise in the computation of calibration solutions, you must discard these channels. In addition to this, initial flux calibration requires only the 75% central channels of each IF or even fewer, where the spectral response can be considered as flat (prior to any band-pass correction). From these central channels, mean gain solutions are obtained for each IF, then these are used to derive the flux density of the PT-Cal at different frequencies.
  - Depending on frequency bands and on the current instrument capabilities, IF1 and IF8 can also be affected by side effects of the base-band. In that case, the 30 first channels of IF1 and 30 last channels of IF8 may sometimes exhibit lower sensitivities and should be excluded in further analysis. More details of how to do this are given in section 7 and 8.
  - If you are running the pipeline and it fails here at the POSSM stage, check the sources included in your first CL table - sometimes it contains only one source instead of all of them, causing CLCAL to only write fringe fitted solutions for one source, instead of all sources, hence POSSM fails. To resolve this, delete ALL CL tables (including CL#1), run INDXR, then re-run the pipeline starting at the dofringfit section.

7. Phase and bandpass calibration

CALIB computes antenna-based gain solutions, by solving the phases and amplitudes for the various baselines. After CALIB runs, solutions are stored in SN tables, which must be checked carefully with SNPLT. It is important to get rid of bad solutions.
We will start by solving the phases only (SOLMOD='P'), then we will solve the amplitudes and phases together (SOLMOD='A&P'). This two-step approach turns out to be very efficient for e-MERLIN data, which can sometimes exhibit large phase variations over short timescales. Phases alone are solved for by using at least 3 antennas (aparm(1)=3) and short time intervals (`solint` less than 2-3 mins). The 'weak' constraints (no amplitude, minimum 3 antennas must give consistent solutions, i.e, solutions with no closure errors) ensures that solutions are easy to find, while the use of a short solution interval ensures a good tracking of the phase variations. Then amplitude and phases are solved for by using a minimum of 4 antennas (aparm(1)=4) and longer time intervals. Stronger constraints are applied at this stage to ensure solutions are robust, and long solution intervals guarantee high SNRs (this is especially relevant for amplitudes, which generally do not vary strongly as a function of time). Note that the solution interval can be increased during this step (beyond 2-3 mins) because, in principle, the short timescale variations of the phases were removed by the first step.

7.1 Compute phase solutions on all calibrators

You now need to solve for the phase part of the antenna gains using CALIB on all calibrators. This normally needs to be done twice, once for the FX-Cal and once for the PT-Cal/PHS-Cal together. The reason is that the flux calibrator

is resolved, and cannot be assumed to be a point source. In normal circumstances, your flux calibrator will be 3C286 (1331+305) for which models at both L-band and C-band are available. (If your flux calibrator is not 3C286, you will need to run CALIB twice but with antenna restrictions; see your e-MERLIN contact if in doubt about how to do this). As `setjy` has previously been used to enter the correct flux densities for each IF for 3C286, the model contributes structure and is OK to use for the entire band.
First, read in the model, which we will assume is in a directory with the logical name `DATA` and in a file called `3C286MODEL`:

- Run `fitld` with `datain'DATA:3C286MODEL'`; `outna'3C286'`; `outcl'MODEL'`
- Notes
  - If your filename has non-capital letters, you will need the `datain` command on a line on its own, without the trailing quote.
  - Using `DATA` as a logical variable interferes with subsequent use of LibreOffice - you may want to choose a different name.
  - The catalogue number of the model file you have just read in will hereafter be referred to as `[3C286.MODEL]`.
- Run `calib` with `getn[ALL.DBCON.1]`; `get2n[3C286.MODEL]`; `calsour '[FX-Cal]'`; `anten 0`; `refant 2`; `weightit 1`; `doflag -1`; `docalib 100`; `flagver 0`; `gainuse 2`; `soltype'L1'`; `solmode'P'`; `nmaps 1`; `ncomp 1000 0`; `solint 0.5`; `aparm 3 0 0 0 0 3 5 0`; `snver n`

and then

- Run `calib` with `getn[ALL.DBCON.1]`; `clr2n`; `calsour '[PT-Cal]''[PHS-Cal]'`; `anten 0`; `refant 2`; `weightit 1`; `doflag -1`; `docalib 100`; `flagver 0`; `gainuse 2`; `soltype'L1'`; `solmode'P'`; `nmaps 0`; `ncomp 0`; `solint 0.5`; `aparm 3 0 0 0 0 3 5 0`; `snver n`
- Notes
  - Note the `clr2n` command which gets rid of the 3C286 model and leaves you with an implicit comparison with point sources.
  - `weightit` and `antwt` set in this way forces the data weights to be the inverse square of the error and the antenna relative weights to 1.
  - `soltype'L1'` performs a least-difference rather than least-square solution; this is always needed with e-MERLIN data to reduce the sensitivity to outliers.
  - `n` is the same in each case; solutions are all written to a single SN table number `n`.
  - If you have already flagged the end channels as in 3.1.2 example 2, you can ignor the `ichansel` parameter. Otherwise, read the AIPS help file carefully. `ichansel` is an array with four columns; in the configuration given the meaning is to restrict the channel range to channels 20-108 for all IFs. If you wanted to, in addition, disregard the first 30 channels of IF1 and the last 30 of IF 8 you would use `ichansel 20 180 1 0 30 180 1 1 20 98 1 8` (assuming your IFs have 128 channels; again, modify as necessary if your IFs are differently subdivided).
  - If you have too many bad solutions, delete the SN table(s) just produced with `extdest` as before, and try the following:increase `solint` to 1 or 2 minutes; change the way the data are averaged: use SOLTYP='L1R'; lower the minimal SNR to select data: aparm(7)=5-3-2;keep data with low SNR: APARM(9)=1
- Run `snplt` with `getn[ALL.DBCON.1]`; `invers=n`; `optyp'PHAS'`; `dotv 1`; `nplots 9`

where `n` is the version number of the SN table that you produced in the last two runs of CALIB. If there are obvious outliers or bad sections of phase solution, use SNEDT and SNSMO as necessary. Refer to section 6.2 for instructions on how to do this. In each case note that a different SN table will be written; keep a careful track of which SN table contains the edited/smoothed corrections.

7.2. Write phase solutions to calibration table, and check them

- Run `clcal` with `getn[ALL.DBCON.1]`; `sources=''`; `calsour=''`; `snver=n`; `invers=n`; `gainver=0`; `gainuse=0`;`refant 2`;`interpol''`

to interpolate the solution table n, written in the last stage, to a new CL table. It is possible to check this table also with SNPLT using inext'CL' if you want to verify that this has been successful.
You should now check that the phase solutions, once applied to the calibrators, give close to zero resultant phases for those calibrators that are point sources, or whose corrections have been derived using models. You can use the megapossm.py script for this, or alternatively

- Run possm for getn[ALL.DBCON.1];source'[FX-Cal]'; anten 2,0; baseline 0; stokes'HALF'; codetype'PHAS'; aparm(9)1; docalib 100; solint 0; dotv 1;nplot 6

Check in the same way the other two calibrators. Again, by default we are only plotting baselines to antenna 2 (Mk2, assuming this to be the reference antenna).

- Notes
- If some averages exhibit significant phase shifts/fluctuations
  - check phase solutions baseline per baseline, using nplots 9
  - Once you find the bad baselines, find the bad time intervals by using solint -1
  - Flag the corresponding data (either with SPFLG or UVFLG), they are likely to be not calibratable. In the case of bad solutions for the PHS-Cal, make sure you also flag the data of the previous and next scans on target.

7.3 Initial bandpass calibration

We will now attempt a preliminary calibration of the bandpass; i.e. the variation of sensitivity across the frequency band. To do this, we use the bright PT-CAL calibrator (usually OQ208).

- Run bpass with all defaults apart from getn[ALL.DBCON.1]; calsour='[PT-CAL]'; docal 100; soltype'L1'; refant 2; gainuse 0; bpassprm 0 0 0 0 1 0 0 0 1 3 1

This will apply the highest-number CL table containing delay and phase solutions and produce a bandpass (BP) table attached to the file. This is a first-pass BP table, because we have not attempted to include the intrinsic spectral index of the PT-CAL. In fact, OQ208 has a typically very inverted spectrum at L-band. This will solve per-scan; if your scans are short or many solutions fail, try solty 'L1' and/or average all times with solint =-1.

7.4 Calibration of Amplitudes and Phases (A&P) of calibrators

We now solve for the complex gains, i.e., amplitude and phase simultaneously. We use the phase only solutions derived previously (in 7.1.c) as prior information for the computation of new solutions. The solutions should be computed by using at least 4 antennas (if possible) and data with SNR $\geq$5.0. We are only going to use the table to derive the flux densities, so any dubious solutions should be discarded (SNEDIT), but don't edit the data themselves.

- Run calib with getn[ALL.DBCON.1];get2n[3C286.MODEL];calsour '[FX-Cal'];antennas 0;refant 2; weightit 1;antwt 0;docalib 100;gainuse 0; doband 4; bpver 0; snver n; soltyp'L1'; solmod'A&P'; nmaps 1; ncomp 1000 0; solint 2; aparm 3,0,0,0,0,3,5,0; cparm 10 0;minamper 10;minphser 10

Here n is the next highest version SN table (i.e. the number of SN tables you already have, plus one). Repeat this for the other calibrator sources in a single further pass, as with the phase calibration (section 7.1), using the same snver:

- Run calib with getn[ALL.DBCON.1];clr2n; calsour '[PT-Cal]''[PHS-Cal]';antennas0;refant 2; weightit 1;antwt 0;docalib 100; gainuse 0; snver n; soltyp'L1'; solmod'A&P'; nmaps 0; ncomp 0; solint 2; aparm 4,0,0,0,0,3,5,0; cparm 10 0;minamper 10;minphser 10

You should then inspect the solutions by

- Run snplt with getn[ALL.DBCON.1];source='[CAL]';antennas 0;optyp'X';nplot 6;invers n;dotv 1

where X is either AMP or PHAS to look at amplitude or phase in turn, with 6 plots per page, and with [CAL] set to the names of each of the calibrators in turn.

- Notes
  - cparm 10 0 means a minimum elevation of 10 degrees; you may have to change this on the second pass if your PHS-Cal is at a very low Dec.

- Checking the average gain solutions for each polarisation and each IF, you should obtain similar solutions for both polarisations (within 10-20%), and similar solutions for each IFs (within 10-20%), unless the spectrum of your calibrator has a steep slope. In C-Band, 3C286 and OQ208 have approximately flat spectra.
- If you notice inconsistencies between the average gain solutions of the IFs, check the diagnostic plots, and check that there is no RFI remaining on antennas with inconsistent gain solutions.
- Check that amplitudes and phases evolve smoothly and consistently as a function of time.
- If you see any discontinuity or erroneous solution, remove it from the solutions before applying them using SNEDT or the `snedit.py` parseltongue script. Refer to section 6.2 for details of how to do this.

7.5 Derive fluxes and spectral indices for calibrators, and refine BP solution

- Run `getjy` with `getn[ALL.DBCON.1];sources='[PT-Cal],[PHS-CAL]';calsour='[FX-Cal]';antenna -1,0;snver=n`.
- Notes
  - `n` is the same as the version of the SN table that you produced in section 7.2
  - Keep track of the PT-Cal flux density derived by `getjy`; check that this value is consistent with that you found in catalogues (see Section 6.-1)
  - If the flux densities of your PT-Cal are different from one IF to another (normally the case), then you have to take into account the spectral index of your source.
  - By default, Lovell should be excluded because its high sensitivity affects the scaling. Use the best timerange and antennas in a given dataset if necessary. For example, De might also be excluded at C-band, or Ca if you think phase-ref is resolved, or any other antenna with e.g. bad RFI, as long as there is 10-20 mins of good data on at least 3 antennas.

When you have a good enough frequency coverage (normally the case with e-MERLIN) you can use `sousp` to fit a spectral index and improve the accuracy, on the assumption that the spectral index is linear over the fractional bandwidth.

- Run `sousp` with `getn[ALL.DBCON.1];sources='[PT-Cal]';order 1;dotv 1`

followed by

- Run `sousp` with `getn[ALL.DBCON.1];sources='[PHS-Cal]';order 1;dotv 1`
- Notes
  - Keep track of the best fit parameters (spectral index), produced by `sousp`. Check whether values are consistent with libraries (see Appendix C).
  - In case one (or 2) IF(s) has/have erroneous fluxe(s) falling far off the best fit curve, you can decide to change manually the flux densities of these IFs to avoid biasing the fit. Don't do that if you have reasons to believe that you have more than 1 or 2 erroneous flux densities. This can be done with a `setjy` loop in exactly the same way as section 6.1.c in order to update the flux density values for the PT-Cal. Then re-run `sousp`.

If happy with the best fit of flux densities, you can update the SU table with the fitted flux densities by running `sousp` with doconfrm=1.
Next, you should **use EXTDEST to delete bandpass (BP) table 1, and SN table n which you produced in 7.4**. You can now regenerate them with the better model for the flux density and spectral index of the PT-CAL:

- Run `bpass` with all defaults apart from `getn[ALL.DBCON.1]`; `calsour='[PT-CAL]'`; `docal 100`; `refant 2`; `soltype'L1'`; `solint 0`; `gainuse 0`; `bpassprm 0 0 0 0 1 0 0 0 1 3 1`; `specindx` = whatever you found for the PT-CAL in the run of `sousp`.

If many solutions fail where you think that the data should be OK, try `solty 'L1'` and/or `solint -1` (which averages all data).

- Run `calib` with `getn[ALL.DBCON.1];get2n[3C286.MODEL];calsour '[FX-Cal'];antennas 0;refant 2; weightit 1;antwt 0;docalib 100;gainuse 0; snver n; soltyp'L1'; solmod'A&P'; nmaps 1; ncomp 1000 0; solint 2; aparm 3,0,0,0,0,3,5,0; cparm 10 0;minamper 10;minphser 10;doband 4;bpver 1`

Here n is the next highest version SN table (i.e. the number of SN tables you already have, plus one). Repeat this for the other calibrator sources in a single further pass, as with the phase calibration (section 7.1), using the same snver:

- Run calib with getn[ALL.DBCON.1];clr2n; calsour '[PT-Cal]''[PHS-Cal]';antennas0;refant 2; weightit 1;antwt 0;docalib 100; gainuse 0; snver n; soltyp'L1'; solmod'A&P'; nmaps 0; ncomp 0; solint 2; aparm 4,0,0,0,0,3,5,0; cparm 10 0;minamper 10;minphser 10; doband 4

7.6 Write a&p solutions to calibration table for ALL calibrators (and target)

Allow the application of the amplitude and phase solutions to all calibrators by use of CLCAL:

- Run clcal for getn[ALL.DBCON.1];source'';calsour'';snver n;invers snver;interpol'';gainver 0;gainuse 0

where n is the version number of the SN table you have just created with CALIB. Again you can check using SNPLT with optyp'AMP' that the new CL table you have created has sensible amplitude corrections.

7.7 Check A&P solutions once applied on ALL calibrators (vector average)

As at the end of section 6, you should now check the data with the solutions applied, except that this time you want to apply solutions for both amplitude and phase, for the highest extant CL table that you have just produced with CLCAL:

- Run possm with getn[ALL.DBCON.1]; sources'[FX-CAL]'; antennas 1 2 5 6 7 8 9; baseline antennas; stokes'HALF'; codetype'A&P'; freqid 1; aparm 0; aparm(9)1; flagver 0; docalib 100; gainuse 0; dotv 1;nplots 6;solint 0

It is quicker to specify just those antennas which are present; change the numbers above as needed.
You can also check these with the megapossm.py script.

- Notes:
  - POSSM runs faster if the antennas present (unflagged) are specified; change as required. These should all have good solutions for point sources, or for extended sources where a model has been used.
  - gainuse 0 is used to apply the highest version number CL table.
  - docalib 100 requires that the CL table (produced by CLCAL) should be applied, but without using weight information
  - aparm(9)1 combines all the IFs into one plot. Once again, remember the AIPS habit of carrying over parameters from one task to another (this is the reason for the aparm 0; you are likely to have these set from previous tasks)
  - At this stage, if A&P calibration went all right, all your phases should be zero and your vector-averaged amplitude should equal the flux of your calibrator (i.e., the flux you enter with SETJY in Section 6.-1 or as derived from GETJY)
  - If some averages exhibit significant shifts/fluctuations: 1) check solutions baseline per baseline by using nplots 9; 2) Once you find the bad baselines, find the bad time intervals by changing solint to -1; 3) Flag the corresponding data (either with SPFLG or UVFLG), they are likely to be not calibratable. In the case of bad solutions for the PHS-Cal, make sure you also flag the data of the previous and next scans on target.

8. IMAGING AND SELF-CALIBRATION (PHS-Cal)

At this stage you are ready to begin making images. The strategy is to image the phase calibrator source first. The amplitude and phase calibration can be tweaked by self-calibration on this source, because its flux density is high enough for good incremental corrections to be obtained. Once this has been done, the solutions can be applied to the target source; this will be described in section 10. Section 9.1 can be omitted if the target is bright; in particular section 9.1.b) is not essential to the further analysis if you have an idea of the expected noise level for your observations.

8.1 Phase calibration of the phase calibrator

At this point you should re-correct the phases after the setting of the bandpass performed in the previous section, using the previous corrections together with the bandpass correction. The remaining corrections should be quite small at this stage. We are using docalib=100 (to apply the calibrations in the previous CL table) and doband=4 (to apply the bandpass table previously made). As previously noted, this assumes that the end channels have already been flagged as in 3.1.2 example 2; if not, see comments on ichansel in 7.1.

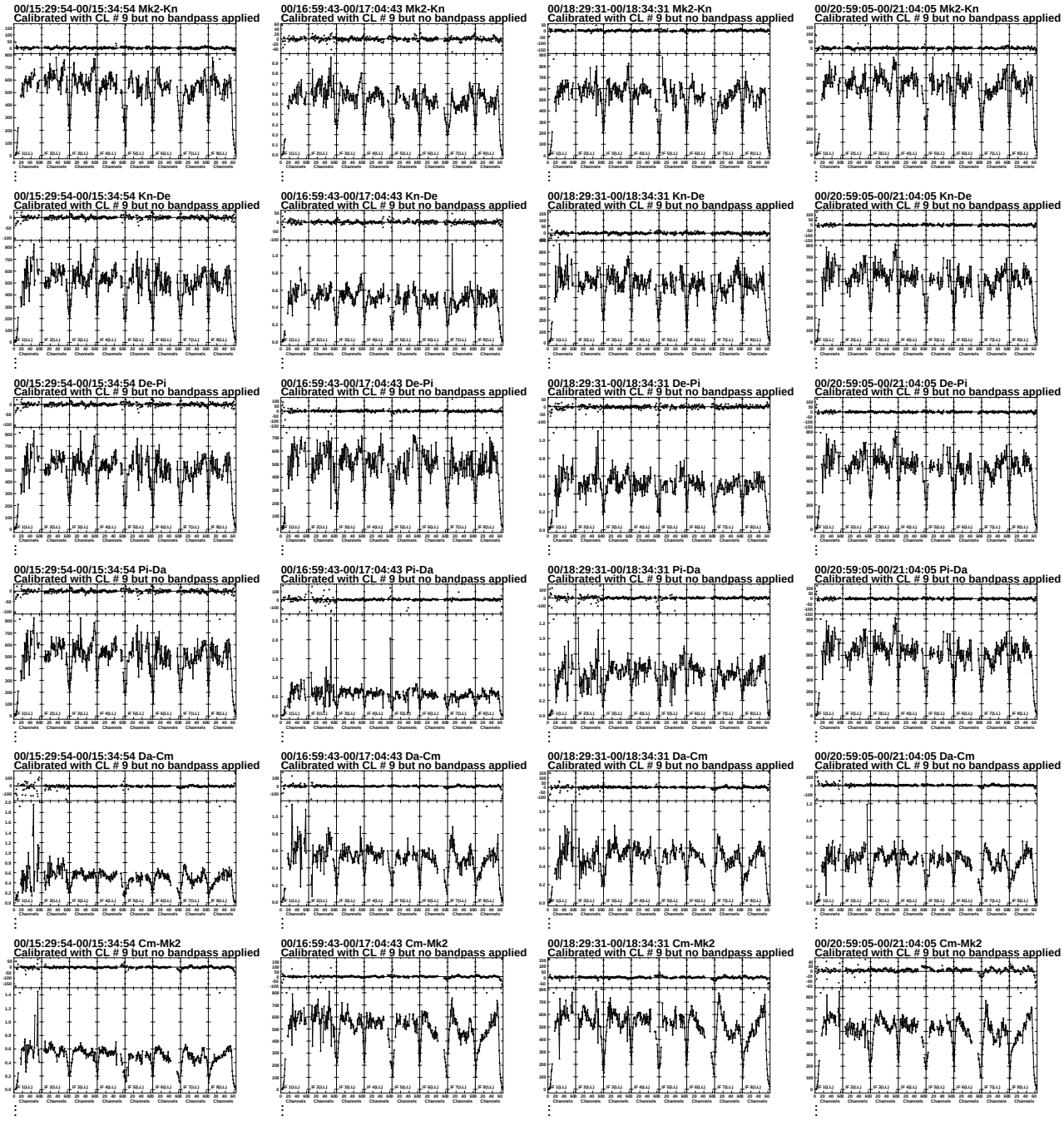

Figure 4: Spectral plot of flagged, fringe-fitted and bandpass-calibrated data.

- Run `calib` with `getn[ALL.DBCON.1];calsour '[PHS-Cal]'];antennas 0;refant 2; weightit 1;antwt 0;docalib 100;gainuse 0; snver 0; soltyp'L1'; solmod'P'; solint 2; doband 4`

- Run `clcal` for `getn[ALL.DBCON.1];source'[PHS-Cal]','[Target]';calsour'';snver n;invers snver;interpol'';gainve 0;gainuse 0`

8.2 Remove outliers

We `CLIP` the data to remove obvious outliers. This should result in a lowering of the noise level. We use `UVPLT` to plot the amplitude (in Jy) as a function of the uv-distance, with out averaging together any channels/IF/polarisation, since `CLIP` operates on unaveraged data. We only plot every 127th point for speed as the aim is to establish the genuine noise scatter, not identify every bad point from `UVPLT`. The plot should be dominated by a scatter about an average value, due to inescapable (instrumental/atmospheric) noise, and we will set the clip level above this noise.

- Run `uvplt` with `getn[ALL.DBCON.1];sources'[PHS-Cal]';stokes'HALF';antennas 0;bparm 0;docal 100;gainuse 0;doband 4;bpver 0;dotv 1;do3col 1;xinc 127`

- Notes

  - All the points should be consistent with a constant average flux density, corresponding here to the flux density of the PHS-Cal. If it is confused or resolved, the shorter baselines may have more flux than the longer ones. Bear in mind that different antennas have different sensitivities;in particular, the Defford baselines at C-band are likely to be noisier. If necessary, use selected combinations of antennas, in order to derive different thresholds for different antennas. As no averaging is done, a significant scatter (several Jy) is seen at all the uv-distances. Note that, for a given uv-distance, the distribution of amplitudes is not necessarily symmetric with respect to the mean value (it is generally extended toward high flux densities).
  - If you see a large clump of high flux densities at some specific uv-distances, you probably still have a bad channel or a bad bit of data at some time. Use `UVFLG` or `SPFLG` to flag these before applying `CLIP` (in order to make sure that noise which happens to be within the average is also flagged).
  - Note the level to apply in `CLIP`; call it `n` Jy.

- Run `clip` with `getn[ALL.DBCON.1];sources'[PHS-Cal]';outfgver 1;aparm n n 0 0`.

See the previous comments about number of flags (section 3). If you have a large number of flags written by CLIP, you may need to do a UVCOP to apply them.

8.3 Imaging/selfcalibration loops

IMAGR inverse Fourier transforms the UV-data and creates images of the real plane. In C-Band, the primary beam size is about 7', which can be covered by a 2048x2048 image with 0.2" pixels. In L-Band, the beam size is about 30', corresponding to 2048x2048 with 0.8" pixels. Full-resolution science images should be made using pixelsize (`cellsize` in AIPS-speak) 0.04 – 0.05" in L-Band and 0.01" – 0.015" in C-Band (smaller values at the higher frequency end). Using a larger pixel size will degrade images considerably.
Self-calibration is a process that calculates telescope gains by comparing the observed data with a model of the sky. These gains are used to provide a corrected dataset, and thence a corrected image, with which the process can be repeated iteratively until convergence. The model is provided by the image produced from the data itself - hence the 'self' - and in particular by the clean components (CC) table attached to the image. The strategy is to reach convergence on the PHS-Cal by iteration, and then apply the accumulated corrections to the target. On each iteration, the image should improve in the sense that artefacts resulting from incorrect telescope gains should become less prominent and the peak/noise ratio should improve.
The CLEAN algorithm treats sources as combinations of points, which can be convolved with a restoring beam into the final image. This is generally less successful for sources with a significant fraction of diffuse emission, in which case you might need to proceed to a multi-scale cleaning of your map (consult a member of the e-MERLIN team in this case).

- Run `imagr` with `getn[ALL.DBCON.1];source'[PHS-Cal]';docalib 100;doband 4;nchav 128;cellsize 0.015;imsize 512;factor -0.3;minpatch 255;minpatch 255;dotv 1;uvwtfn'Na';outname'PHS_0';nboxes 1;clbox x1 y1 x2 y2;niter 300`

- Notes

  - This assumes that you have already flagged the end channels as in 3.1.2 example 2; `nchav 128` will use all the unflagged channels. Modify the number of channels if your file does not have 128 channels per IF.

- This cellsize is suitable for C-band. Use something larger, e.g. 0.05, for L-band. If using a different cellsize, replace 0.015 with your new cellsize in all subsequent instructions.
- The `clbox` parameter specifies the border of a CLEAN window for the algorithm to use. If set, this specifies that only this area of the map should be examined for flux; normally this is the central region if you have a single point source as your phase calibrator. In general, specifying CLEAN windows gives faster convergence, although you can run IMAGR without them. You can reset the windows interactively using `dotv 1`.
- It is worth making a large map, if you suspect there may be significant flux from other sources in the field. Use a bigger `imsize` for this (say 4096 with an increased `cellsize`). For a large map to check most of the primary beam, you are normally aiming for a field size of 20-30 arcminutes at L-band with no Lovell (chose `cellsize` such that `cellsize`×`imsize` $\sim$ 1000-2000).
- If you find other sources in the field with flux densities high enough for their sidelobes to affect the source you actually want ($\sim 5\sigma$ close to the field centre, or a bit more further away), you should avoid this by cleaning them out in multiple 512-pixel fields using the `nfield`, `rashift` and `decshift` arrays to specify the positions of these other sources. Also set `do3dimag=1` to get accurate locations.
- Another file will be produced whose name is the `outname` variable followed by `.ICL001` (or multiple files for multiple fields). If you specify no CLEANing with `niter 0`, the default suffix is `IIM001`.

- Run `ccmrg` and `prtcc` with getn[PHS_0.ICL001].
- Use `tvwin` and `imstat` to measure the noise rms and the peak, off- and on-source.
- Notes
  - We expect that many phase-reference sources will have some extended emission at e-MERLIN sensitivities. The first round(s) of self-calibration are phase-only, and therefore it is important to select CC which represent the structure of the source. If your boxing excluded any sidelobes, there are several strategies: you can go down to the last CC before the first negative, or to 2 sigma rms, or simply use all the CC. If you think that there may be CC for artefacts, you can edit the list, consult staff.
  - Note the believable number of CC `n` and use `ncomp n` in the next round of self-calibration.
  - Set `nmaps 1` or a higher number if using more than one field as below.
  - In the case of multi-sources in the field: consider each source independently (IMAGR creates one image per source). You don't have to restrict your model to CC associated to the PHS-Cal. If the other sources in the field are bright and quite compact, you can constrain further your model by including CC associated with these sources. For example, `ncomp 30,15,20,0` would select 30 CC for source 1, 15 CC for source 2 and 20 CC for source 3. Update with the number of CC you want to use.

And now for the selfcalibration:

- Run `calib` with `getn[ALL.DBCON.1];get2n PHS_0.ICL001;calsour'[PHS-Cal]';refant 2;weightit 1;antwt 0;docalib 100;doband 4;soltyp'L1';solmode'P';solint 2;aparm 3 0 0 0 3 5 0;cparm 10 0;minamper 10;minphser 10;ncomp n;nmaps 1`
- Notes
  - See previous comments on the use of `ichansel` (if you have not flagged the end channels already), `weightit`, `soltyp`, `refant` and `aparm` in section 7.1
  - We are solving for only the phase part of the gains. This makes the procedure more stable initially.
  - Set `ncomp` and `nmaps` according to your examination of the CCs from the previous best image. Be careful not to include sidelobes in the model. If the phase-reference source seems point-like and there are no confusing sources, you can leave these parameters and `in2n` etc. blank, to just use a point model of the flux density set in the SU table.
  - You may need to decrease `cparm 10 0` for very low-Dec sources
  - `solint` Usually, set this to the length of each phase-reference source scan (from `LISTR`). If the phase changes rapidly and there is good signal-to-noise, try e.g. 0.5 min.

– Keep track of the fraction of bad solutions and average closure rms. If you have too many bad solutions (typically more than 20%), inspect the solutions and data. If these are mostly in a particular time interval and/or IF, this might be remaining bad data. If they are mostly one antenna e.g. De at C-band, see App. A. If it is simply a faint, noisy phase reference source you can average the polarisations: aparm(3)=1; you can average the IFs: aparm(5)=1; if the solutions are still poor, you can increase the solution interval: solint 10 mins. In this latter case, you will have to SPLIT your file to obtain a single-source file and re-index the data to have longer scans. Refer to a member of the eMERLIN team.

Next you should, as usual, examine the phase solutions which have been produced.

- Run `snplt` with `getn[ALL.DBCON.1];sources'[PHS-Cal]';inext'SN';invers 0;optyp'PHAS';opcode'ALSI';dotv 1;nplot 7;do3col 1;factor 2`

Check that phases evolve smoothly and consistently as a function of time. Phase corrections should not exceed 20 degrees at this stage, unless one of the antenna is really bad.
Now apply the phase corrections:

- Run `clcal` with `getn[ALL.DBCON.1];sources'';calsour'';interpol'2PT';refant 2;invers n;snver n`
- Notes
  - n is the number of the SN table just created
  - These corrections are written into the solution table for application to the target as well.

If you are sure the phase-reference is an unresolved point, go straight to an `'a&p'` solution. Otherwise, this section should be iterated until convergence of the phase solutions. Every new iteration, will produce a new set of images. If you keep the same name for the successive images (same INNA and INCL), the INSEQ will increment at each iteration. You will be doing either or both of:
Improving the model by including more accurate structure
Improving the solutions by chosing a shorter solution interval e.g. 0.5 minutes
The process has converged when:

- the peak value of the PHS-Cal does not increase further
- the add-on phase solutions stored in the last SN table are less than 1 degree and dominated by noise (i.e. no trend can be identified)
- the image of the phase calibrator looks indistinguishable from that of the previous iteration.

Every time you start a new iteration (CALIB-SNPLT-CLCAL-IMAGR):

- keep track of which INSEQ corresponds to the image you are using for the model. Be sure to `get2n` the correct image for use in CALIB.
- examine this image's CC file in the same way to determine the number of components (`ncomp`) to be included in the model.
- run CALIB with `docalib=100` on the same data file, with `gainuse=0` to pick up the most recent calibration.
- Normally you will want to do one or more iterations using only phase corrections in the CALIB (`solmod'P'`) followed by an iteration using both phase and amplitude (`solmod'A&P'`). Select the number of CC (`ncomps`) with great care. The cumulative flux (right-most column of prtcc output) should correspond to the expected flux density of the phase-reference source. The amplitude solutions should be close to 1. If there is an isolated high point, this corresponds to a low point in the data - flag it. If many solutions are $\ll 1$, probably there is not enough flux included in your CC. Discrepancies can also be caused by poor phase calibration or bad data or previous calibration errors, consult staff.

9. Imaging and selfcalibration (target)

Although you can begin from scratch to build up the model of the target, if you have any previous high-resolution image of your target to use as a starting model, this will make the process quicker.

9.1 Create a single data set with target only

9.1.2 Extract the target only

Create a dataset containing only the target, with all calibrations up to the end of the previous section applied, and all flagging (including the flagging of end channels).

- Run `splat` with `getn[ALL.DBCON.1];sources'[TARGET]';docalib 100;gainuse 0;outname'TARGET'`

9.1.2 Merging data sets (if not already merged with `doconcat=1` in the pipeline)

If you have several days of observations, and they have not already been merged in the pipeline, this is the right moment to merge your data sets with DBCON (as long as the source is not expected to have varied significantly). This will improve the SNR of your target image and thus provide more accurate self-calibration (if possible).

- Run `dbcon` with getn [TAR.DAY1]; get2n [TAR.DAY2]; dopos 1; doarray 1; go
- Notes:
  - This assumes that you have a single source, observed at the same position, with e-MERLIN, in both data sets.
  - Repeat as needed if you have more than two input data sets.
  - Below, TARGET.SPLAT will refer to your single-source file, whatever you have only one day of observation (TARGET.SPLAT) or several days of observation (TARGET.DBCON).

9.2 Clip outliers from the target.

Again we CLIP the data to remove obvious outliers. This should result in a lowering of the noise level. You might want to apply different threshold for different antennas. The threshold to apply can be determined directly from a UVPLT diagram. We use UVPLT to plot the amplitude (in Jy) as a function of the uv-distance, without averaging together any channels/IF/polarisation.

- Run `uvplt` with `getn[TARGET.SPLAT];sources='TARGET';stokes'HALF';antenna 0;dotv 1;do3col 1;xinc 253`
- Notes
  - From now on there is only one source - the target - and so the `sources` parameter will be omitted.
  - The plots should show all the points, although if the target is faint you may not see significant flux. However, you should be able to spot clear outliers in the distribution. As no averaging is done, a significant scatter (several Jy) is seen at all the uv-distances. Note that, for a given uv-distance, the distribution of amplitudes is not necessarily symmetric with respect to the mean value (it is generally extended toward high flux densities).
  - If you see a large clump of high flux densities at some specific uv-distances, you probably still have a bad channel or a bad bit of data at some time. Use `UVFLG` or `SPFLG` to flag these before applying `CLIP` (in order to make sure that noise which happens to be within the average is also flagged).
  - Note the level to apply in `CLIP`; call it `n` Jy.
- Run `clip` with `getn[TARGET.SPLAT];antenna 0;aparm n n 0 0`

9.3 Self-calibration of the target (initialisation)

9.3.1 Create the initial image(s) - Standard CLEANING

- Run `imagr` with `getn[TARGET.SPLAT];source'[TARGET]';docalib -1;doband -1;nchav 128;cellsize 0.015;imsize 512;factor -0.3;dotv 1;uvwtfn'NA';outname'OUT';niter 300`
- Notes
  - This assumes that you ahve 128 channels per IF.
  - Setting `niter` to a negative number causes IMAGR to stop when it reaches the first negative component. This is a cautious approach which may be worthwhile on this initial pass; after the first pass, you can use the CC file to work out the approximate value for `niter` on subsequent iterations.
  - As before, `cellsize` should be larger for L-band. Set `imsize` so that the size of the map (the product of `imsize` and `cellsize`) big enough for your target or at least 512 pixels (to allow effective beam deconvolvution).

- If your target is faint you can measure an approximate noise level off-source by use of `IMSTAT`. If it is dynamic-range limited, you could try simultaneously mapping a blank image 1-5 arcmin away from the target depending on band.
- Check the spatial extension/structure of your target. Depending on the structure of your target, you will opt for 2 different strategies when "cleaning" the map. If it is compact, the standard cleaning procedure (as used so far) is fine, as it will tend to make your target more and more compact. If it exhibits a significant amount of extended/diffuse emission and/or several sub-structures, you might want to tune the IMAGR parameters to proceed to a multi-scale cleaning of the map (seep Appendix E).
- You should increase the imsize as needed, or set several separate fields, if you have a large target and/or confusing sources. This is especially likely at L-band, consult staff if you are unsure what to do.

9.3.2 Define the model(s)

Use PRTCC on the initial image you have just produced to examine the CLEAN components, in conjunction with examination of the image using TVALL. You will have to determine which CCs to include within the model. As a general rule, you should be conservative on a first pass and probably include only CCs until the first negative. Alternatively, if you have generated more CCs than this using a positive value of `niter`, you should select only CCs corresponding to regions containing flux that you believe. Consult an expert if necessary for advice at this point.

9.3.3 Compute new phase calibrations of the target (using a CC model)

You should first consider whether your target source contains enough flux for self-calibration. The criterion for success is that all baselines must individually see S:N of at least 3, in one atmospheric coherence time. As a rule of thumb, for both L-band and C-band, 1 minute coherence time gives minimum target fluxes for self-calibration of a few mJy in compact structure. The requirement for compact structure arises because the flux must be visible on all baselines, including the long ones. If you do not have enough flux, CALIB will fail.

- If you are satisfied that you have enough flux for selfcal, run `calib` with `getn[TARGET.SPLAT];get2n[image-file-name];refan 2;weightit 1;antwt 0;docalib -1;soltyp'L1';solmode'P';solint 10;aparm 3 0 0 0 3 5 0;cparm 10 0;minamper 10;minphser 10;ncomp n;nmaps 1`
- Notes
  - As ever, modify `cparm 10 0` for very low-Dec. targets
  - `ncomp` is the number of CLEAN components in the model, determined in the last step
  - Keep track of the fraction of bad solutions and average closure rms
  - If you have too many bad solutions (typically more than 20%), you have different options to improve the solutions: you can average the polarisations: aparm(3)=1; you can average the IFs: aparm(5)=1;if the solutions are still poor, you can increase the solution interval: solint 10 mins. In this latter case, you will have to re-index the data to have longer scans. Refer to a member of the eMERLIN team. If all of these fail, you may not have enough flux in your target for successful self-calibration.
- Run `snplt` with `getn[TARGET.SPLAT];inext'SN';invers n;optyp'phas';opcode'alsi';dotv 1;do3col 1`

Check that phases evolve smoothly and consistently as a function of time. If the target is resolved, your phases can vary over several 10s of degrees.

9.3.4 Write new phase calibrations to CL table

- Run `clcal` with `getn[TARGET.SPLAT];sources='';` `calsour='';` `snver=n;` `invers=n;` `gainver=0;` `gainuse=0;refant 2;interpol''`

9.4 CLEAN-selfcal iteration.

In exactly the same way as section 8 on the phase-cal, you can now continue with the CLEAN-selfcal cycle. You should begin by `getn` of the file TARGET.SPLAT, and CLEANing it as in part 10.3.a, with `docal 100`; `gainuse 0` except that you can now be slightly less conservative with `ncomp` and with the number of CLEAN components you use in the model. You can then use CALIB again with the improved model, and reduce the solution interval if the signal-to-noise allows. Amplitude self-calibration requires a high signal-to-noise, but might be possible; if in doubt try a long solution interval first. Inspect the solutions and check that the amplitude of the peak does not change drastically and the noise decreases at each calibration round.

Usually, no reweighting is done until after all calibration in order to ensure that the less sensitive antennas are included full in the models and calibration solutions. However, at the top end of C-band where De is very insensitive, it may be necessary to use `WTMOD` in order to get an image good enough to detect all the target structure, see Section 9.5.2.

The process has converged when:

- the peak value and/or structure of the target do not increase/change further;
- the image stops getting better.
- Note about multi-source imaging: When the model of the sources other than your target seem to have converged, you might want to subtract the CC corresponding to these sources if they are bright enough to affect the emission of your target. Refer to a member of the eMERLIN team to proceed. Note that UVSUB can be applied only on a SPLIT file (single-source file). SPLIT files do not support CL tables, which means that incremental calibration solutions can't be done any further. Any new calibration solution will be stored in a new SN table only. The solutions must be applied to the data before new ones are computed with CALIB (iteration of SPLIT). SPLIT must be applied only if you're near the last calibration solution on your target.

9.5 Reweighting the data

In order to use the data efficiently, you should reweight it so that the sensitive baselines (especially those with the Lovell telescope) are weighted up, and others are down-weighted. This does not happen by default, and the data weighting can easily make a difference of a factor of 2 to the noise level in your final map. The required weight is not only a function of baseline, but also of other factors such as channel sensitivity, presence of low-level RFI and a number of other things.
There are two possible ways of reweighting: (1) use of default values for each telescope using the AIPS task `WTMOD` and (2) use of the AIPS task `REWAY` which weights according to the local noise level in the data by determining rms as a function of time and channel. We are currently experimenting with `REWAY` and this is likely to become the preferred method. You may wish to try both and see which gives the lower noise level; please let us know the result.
9.5.1 Reweighting with `REWAY`:

- Run `REWAY` with `getn[TARGET.SPLAT]`; `docalib=-1` and other parameters set to defaults apart from `aparm(4)=2` and `aparm(10)=1` to use boxcar smoothing and keep the WT table.

9.5.2 (alternative to 9.5.1): Reweighting with `WTMOD`:
Here we apply relative weights between the antennas, so that the best antennas have a higher contribution. Initially, we suggest to apply the weights that are given in the MERLIN user guide. These are based on measurements of antenna relative efficiencies in nominal conditions.
The recommended weight arrays are as follows:
antwt 50.0, 1.0, 0., 0., 1.70, 0.10, 1.50, 1.70, 4.80, 0.; $ C-Band weights (+ Lovell assumed to be 50)
antwt 30.3, 1.0, 0., 0., 0.73, 0.61, 0.77, 1.00, 1.74, 0.; $ L-Band weights

- Run `wtmod` with `getn[TARGET.SPLIT]` and `antwt` set to these values.
- Notes:
  - Lovell and Cambridge, which have the largest dishes, are weighted up. Defford, which is a non-solid antenna, often exhibits a larger dispersion of flux densities and is generally weighted down.
  - These weights were computed for the old MERLIN design, and might not be the most adapted for the e-MERLIN. We therefore adivse users to refer to 10.5.b) for a better determination of the e-MERLIN antenna weights.
  - A new file PHS+TAR.WTMOD will be produced. Run INDXR on this file to re-index the data.

Whatever way you have reweighted the data, produce an image, from the data with the new weights:

- Run `imagr` with (e.g., C-band values) `getn[TARGET.WTMOD];source'[TARGET]';docalib -1;doband -1;nchav 128;cellsize 0.015;imsize 512;factor -0.3;dotv -1;uvwtfn'NA';outname'NOISE_N';niter 0;do3dimag 1`

Check (again) with TVALL/TVSTAT that the noise level is what you expect. With the Lovell included in the array, re-weighting the data typically decreases the noise by a factor of 2.
It is possible to re-run `WTMOD`, tuning the parameters iteratively. See Appendix D if you wish to do this.

10. POLARIZATION CALIBRATION

- This section is under test. Preliminary testing indicates that polarization is mappable, although we are currently investigating ways to get high dynamic range in Stokes Q and U. If you are intending to attempt polarization imaging, you are for the moment advised to visit Manchester during your data reduction.

In order to calibrate polarization, you will need two things; a zero-polarization calibrator in order to calibrate the instrumental leakage terms, and a polarized calibrator in order to calibrate the delay between R and L polarizations, and set the absolute polarization position angle. Usually, the zero-polarization calibrator in an e-MERLIN observation is 3C84 (0319+415) and the polarized calibrator is 3C286 (1331+305). If necessary, OQ208 (1407+286) can be used as a zero-polarization calibrator, although it may not give as good results as 3C84.
In order to do the calibration using 3C286, you will need to calibrate the parallel hands of polarization (Stokes LL and RR) for all antennas, rather than only those using the short baselines. There are two ways to do this. One is to get a 3C286 model from your e-MERLIN contact and run CALIB on your 3C286 data, correcting amplitude and phase on a short (1-minute) SOLINT, and apply the resulting SN table using CLCAL. The second is to use the instructions in section 12, using 3C286 as your target, in order to derive an image and model for 3C286, before returning to this section.
Assuming that you now have a calibrated dataset for ANGPOL-Cal (3C286 in nearly every case):

- run `possm` with `source'[ANGPOL-Cal]';stokes'';doband 4;docalib 100;nplot 4;dotv 1;aparm(9)=1;dopol -1`
- Notes
  - You should see well-calibrated bandpasses on the parallel hands (LL and RR), including a spectral slope in the amplitudes and no significant phase slopes. If not, revisit previous sections on bandpass correction and fringe fitting respectively.
  - The cross-hands are likely to show significant amplitude variations (up to 100%), particularly at L-band, and phase drifts. These must be removed.
- run `rldly` with `source'[ANGPOL-Cal]';doband 4;docalib 100`
- Notes
  - This should correct the delays on the cross-hands which manifest themselves as phase slopes across the bandpass on LR and RL correlations in the previous run of POSSM. If you don't have such phase slopes, you don't need to run RLDLY.
  - RLDLY produces another CL table. If you have been using GAINUSE to control which CL table is used for on-the-fly calibration, you will need to increment it; otherwise GAINUSE=0 will pick up the most recent version.
- Run `pcal` with `getn[TARGET.SPLIT];calsour [ZEROPOL-Cal];docalib 100;spectral 1;refant 2`.
- Notes
  - This should produce antenna polarization residuals which are written in new tables (PD and CP) which should appear in the image header if you type `imhead`.
  - Use the Mk2 (telescope 2) as the reference antenna.
  - Check the sizes of the residuals written by PCAL. They should normally be a few percent. If they appear strange (identically zero on a telescope which contains data, or greater than about 10%) check your data carefully. If you have a noisy IF (typically IF1 in L-band data) the corrections may be largely random on this IF, and you should not use it for making polarization maps.
- Run `rldif` with `getn[TARGET.SPLIT];source[ANGPOL-Cal];dopol 1;docalib 100;doband 4;doapply 1;spectral 1`
- Notes
  - This should give you a long printout of R-L phase differences separated by channel. They should be reasonably coherent across IFs.
  - If ANGPOL-Cal is 3C286, you do not need to set the variable POLANGLE, since it is known to RLDIF. If the source is called something other than 3C286 or 1331+305, you should also set POLANGLE (the value is 33 degrees for 3C286).
  - PD and CP tables should now have been created, and you should be able to see this using `imhead`.

When you have done this, you should now have a solution applied to the LR and RL crosshands. Repeat the check with POSSM, but this time with DOPOL=1. Ripples on the amplitudes, and phase slopes, should now have been removed.
Finally, check that the polarization position angle of your ANGPOL-Cal is correct by running RLDIF again, but this time on a per-IF basis:

- Run `rldif` with `getn[TARGET.SPLIT]`; `source[ANGPOL-Cal]`; `dopol 1`;`docalib 100`; `doband 4`; `doapply 0`; `spectral 0`

You should obtain for each IF a matrix with consistent values of twice the polarization position angle (i.e. 66 degrees for 3C286). If you do not, you should issue the command `for i=1:n;clcorprm(i)=66-clcorprm(i);end` and then run CLCOR with `stokes'L';opcode'POLR';dopol 1` to rotate the position angles.

11. Final imaging of the target

In this section, you must choose which situation is best for your target image:
- cleaning by hand with interactive boxing (dotv=1);
- automatic cleaning (dotv=0), but with fine tuning of the IMAGR parameters.
If you want to image several fields, choose `do3dimag=1`, set the parameters `nfield` according to how many fields you have, and specify `rashift`, `decshift` for each field.
At this stage it is worth trying different weighting schemes, including pure natural weighting (uvwtfn: 'NA', 'NS') if your source is extended, and robust weighting (uvwtfn: 'RO', 'RS') with different values of `robust` if your source is point-like. Decide which weighting is optimal to image your target and to match your science goals (sensivitiy / resolution).

11.1) Standard cleaning

- (natural) Run `imagr` with `getn[TARGET-last-file]`;`docalib -1`;`doband -1`;`nchav 128`; `cellsize 0.015`; `imsize 512`; `factor -0.3`;`uvwtfn'NA'`
- (uniform) Run `imagr` with `getn[TARGET-last-file]`;`docalib -1`;`doband -1`;`nchav 128`; `cellsize 0.015`; `imsize 512`; `factor -0.3`;`uvwtfn'RO'`;`robust 0`

11.2) Multi-scale cleaning

- Run `imagr` as for 12.1), but with `ngauss` 1. Refer to 10.3.b to specify the parameters wgauss, fgauss and imagrprm(11).

11.3) Wide-field imaging

An experimental part of the pipeline now exists for wide-field imaging. This takes in the `ALL.DBCON` file, splits off a single target using the existing calibration, and forms multiple small images by iteratively shifting the dataset and using `IMAGR`. It then combines these into a mosaic, applying default primary beam corrections. To use this, run the pipeline again, but with `dowide=1` and all other `do`-parameters set to -1. Within `dowide`, you will need to set the `widetarget` in the `doall.inputs` file to the name of your target source. It is recommended to leave all the other subsidiary parameters to 1, although you may want to set `dosad=0`. This will attempt to use the AIPS task `SAD` to detect sources, and this takes a long time.

Appendices A-E follow this point. They mostly concern lists of suggested inputs for more abstruse and/or experimental procedures referred to in the text. Please, ask for assistance from an e-MERLIN team member if needed.

Appendix A: Further robust/conservative A&P solution computations

+The precision of data calibration depends upon the individual performances of the set of antennas used in CALIB to compute the gain solutions. Using (a) noisy antenna(s) can result in less precise amplitude and phase calibrations. In the case of e-MERLIN, Defford, which is a mesh dish, is less efficient than the other antennas (mainly in C-Band). In C-Band, the amplitude of the signal in Defford is between 20 and 50% smaller than in the other antennas, and the noise level is higher.
Also, e-MERLIN has the specificity to involve a rather heterogeneous set of antennas, when the Lovell is included in observations. All the baselines involving the Lovell inherits much higher weights. While a proper use of the weights

of e-MERLIN data is still under study, we advise the e-MERLIN users not to use the Lovell baselines to compute (initial) gain solutions. Instead, we suggest to compute calibrations solutions in 2 steps: 1) compute the solutions for all the antennas excluding Defford and Lovell; 2) bootstrap the solutions of Defford and Lovell using the other antenna solutions as a prior.+

A.1) Compute solutions excluding the Lovell and Defford:

default CALIB;
getn ALL.DBCON;
calsour '1331+305',";
antennas 2,7,8,0; dofit 0; refant 2;
weightit 1; antwt = 0;
doflag -1; flagver 0;
docalib 100; gainuse 3; snver = 3; $ Use previous phase solutions
soltype 'L1'; solmode 'A&P'; cmethod 'DFT'; solint=2;
aparm = 3, 0, 0, 0, 0, 3, 5, 0;
cparm = 10, 0; minamper 10; minphser 10;
If you have not already flagged the end channels, in each execution of CALIB also add a suitable selection such as: ICHANSEL(1,1) = 20; ICHANSEL(2,1) = 108; ICHANSEL(3,1) = 1; ICHANSEL(4,1) = 0;
Keep track of the average gain solution for each IF:
default CALIB;
getn ALL.DBCON;
calsour '1407+284',"
antennas -1,-6,0; dofit 0; refant 2;
weightit 1; antwt = 0;
doflag -1; flagver 0;
docalib 100; gainuse 3; snver = 3; $ Use previous phase solutions
soltype 'L1'; solmode 'A&P'; cmethod 'DFT'; solint=2;
aparm = 4, 0, 0, 0, 0, 3, 5, 0;
cparm = 10, 0; minamper 10; minphser 10;
Keep track of the average gain solution for each IF:
default CALIB;
getn ALL.DBCON;
calsour '0645+0541',"
antennas -1,-6,0; dofit 0; refant 2;
weightit 1; antwt = 0;
doflag -1; flagver 0;
docalib 100; gainuse 3; snver = 3; $ Use previous phase solutions
soltype 'L1'; solmode 'A&P'; cmethod 'DFT'; solint=10;
aparm = 4, 0, 0, 0, 0, 3, 5, 0;
cparm = 10, 0; minamper 10; minphser 10;
Keep track of the average gain solution for each IF:

- Notes:

If you check the content of SN 3 with SNPLT, you won't see 1 and 6;
If you need to edit the solutions, use SNEDT.

A.2) Apply solutions for all the antennas apart the Lovell and Defford:

default CLCAL;
getn ALL.DBCON;
sources '1331+305',";
calsour sources;
opcode 'CALI'; interpol 'SIMP'; refant 2;
gainver 3; gainuse 4; invers 3; snver invers;
$cutoff 120 $ Don't extrapolate/interpolate beyond 120 minutes?
default CLCAL;
getn ALL.DBCON;
sources '1407+284',";
calsour sources;
opcode 'CALI'; interpol 'SIMP'; refant 2;
gainver 3; gainuse 4; invers 3; snver invers;

$cutoff 120 $ Don't extrapolate/interpolate beyond 120 minutes?
default CLCAL;
getn ALL.DBCON;
sources '0645+0541','0639+0553', '';
calsour '0645+0541','';
opcode 'CALP'; interpol 'SIMP'; refant 2;
gainver 3; gainuse 4; invers 3; snver invers;
$cutoff 120 $ Don't extrapolate/interpolate beyond 120 minutes?

A.3) Derive the solutions for the Lovell and Defford

default CALIB;
getn ALL.DBCON;
calsour '1331+305','';
antennas 0; dofit 1,0; refant 2;
weightit 1; antwt = 0;
doflag -1; flagver 0;
docalib 100; gainuse 4; snver = 4; $ Use solutions from other antennas
soltype 'L1'; solmode 'A&P'; cmethod 'DFT'; solint=2;
aparm = 4, 0, 0, 0, 0, 3, 5, 0;
cparm = 10, 0; minamper 10; minphser 10;
Keep track of the average gain solution for each IF:
default CALIB;
getn ALL.DBCON;
calsour '1407+284','';
antennas 0; dofit 1,6,0; refant 2;
weightit 1; antwt = 0;
doflag -1; flagver 0;
docalib 100; gainuse 4; snver = 4; $ Use solutions from other antennas
soltype 'L1'; solmode 'A&P'; cmethod 'DFT'; solint=2;
aparm = 4, 0, 0, 0, 0, 3, 5, 0;
cparm = 10, 0; minamper 10; minphser 10;
Keep track of the average gain solution for each IF:
default CALIB;
getn ALL.DBCON;
calsour '0645+0541','';
antennas 0; dofit 1,6,0; refant 2;
weightit 1; antwt = 0;
doflag -1; flagver 0;
docalib 100; gainuse 4; snver = 4; $ Use solutions from other antennas
soltype 'L1'; solmode 'A&P'; cmethod 'DFT'; solint=10;
aparm = 4, 0, 0, 0, 0, 3, 5, 0;
cparm = 10, 0; minamper 10; minphser 10;
Keep track of the average gain solution for each IF:

- Notes:

If you check the content of SN 4 with SNPLT, you will see that all antennas have a gain of (1,0deg) apart 1 and 6.

A.4) Apply solutions for all the antennas

default CLCAL;
getn ALL.DBCON;
sources '1331+305','';
calsour sources;
opcode 'CALI'; interpol 'SIMP'; refant 2;
gainver 4; gainuse 5; invers 4; snver invers;
$cutoff 120 $ Don't extrapolate/interpolate beyond 120 minutes?
default CLCAL;
getn ALL.DBCON;
sources '1407+284','';
calsour sources;
opcode 'CALI'; interpol 'SIMP'; refant 2;

gainver 4; gainuse 5; invers 4; snver invers;
$cutoff 120 $ Don't extrapolate/interpolate beyond 120 minutes?
default CLCAL;
getn ALL.DBCON;
sources '0645+0541','0639+0553', '';
calsour '0645+0541','';
opcode 'CALP'; interpol 'SIMP'; refant 2;
gainver 4; gainuse 5; invers 4; snver invers;
$cutoff 120 $ Don't extrapolate/interpolate beyond 120 minutes?
Then tidy up your list of SN and CL table so that you can keep using this recipe and its incremental order of SN and CL table:
default EXTDEST;
getn ALL.DBCON;
inext 'SN'; invers 4;
default EXTDEST;
getn ALL.DBCON;
inext 'CL'; invers 4;
default TACOP;
getn ALL.DBCON; geton .DBCON;
inext 'CL'; invers 5; outvers 4;
default EXTDEST;
getn ALL.DBCON;
inext 'CL'; invers 5;

APPENDIX B: Using PT-Cal and PHS-Cal to calculate a BP correction

This sections applies only if you could not derive the BP tables of some of the antennas present in your PHS-Cal+target run. The idea is to use the observations on the PHS-Cal to derive the spectral response of the missing antennas. One method could consist in applying the current BP table (that derived from the PT-Cal observations) to the data and then re-run BPASS on the PHS-Cal. To apply calibration or BP tables to data, you have to use SPLAT or SPLIT (single source file only). Unfortunately, uncalibrated data can't be passed through these tasks, which means that if there are asbolutely no data in your PT-Cal scan, you can't apply this method. An alternative to the previous methods consists in taking advantage of some AIPS functionalities regarding the computation of successive BP tables. The combined BP table is created in 2 steps (including a loop), using the different sources and weights.

B.1) Compute BP-table for all antennas from PHS-Cal

default BPASS;
getn ALL.DBCON;
calsour '0645+0541','';
antennas 0; refant 2;
bpassprm(5) = 1; $ do no normalization before determining the solutions
bpassprm(9) = 1; $ interpolate over flagged channels
bpassprm(10) = 3; $ use ICHANSEL to select best channels for normalising the gain and setting the average phase to 0
ICHANSEL(1,1) = 25;
ICHANSEL(2,1) = 128-25;
ICHANSEL(3,1) = 1;
ICHANSEL(4,1) = 0;
$bpassprm(10) = 1; $ use all channels to normalise bandpass
bpassprm(11) = 1; $ make weights given to the solution be independent of channel.
flagver 1; outvers 2; $ We create a new BP table, so that we can compare with the previous one
docal = 100; gainuse 4;
soltyp = 'L1'; solint = -1; specindx = ??;
bpassprm(2) = 1; minamper 7; minphser 7; $ report closures 7%/7d
= you get initial solutions, for all antennas present in the PHS-Cal run

B.1) Refine BP solutions for antennas that have data on PT-cal

To improve the solutions for the antennas that are present in the PT-Cal, you simply re-run BPASS on the same BP table, using the data of the PT-Cal. The previous data are kept by AIPS, so that only the data in common will be updated. AIPS averages 2 existing BP tables when merging them into 1. After several iteration, the content of the combined BP-table converges toward the solution you would obtained with the PT-Cal only (while keeping the BP

tables for the missing antennas).
tget BPASS;
calsour '1407+284',";
for i = 1 to 10; go; wait; end for; $ 10 iterations seem to be enough for convergence.

- Notes:
  - You can run the loop with weightit 1, so that the PT-Cal observations have higher weights. This should make the convergence quicker.
  - You can check that the BP solutions converged towards the good solution by comparing BP 1 and BP 2.

default POSSM;
getn ALL.DBCON;
sources ";
flagver 1; bpver 2;
dotv 1; nplots 2;
solint 0; $ average over whole time
aparm 0, 1, 0.5, 1.4, -20, 20, 0, 2, 1, 0; $ Plot BP, with amp/ph ranges
bparm 0;
= Once you checked that BP 1 and BP 2 are the same for all the antennas present in the PT-Cal scan, you can delete BP 1 and replace it by BP 2:
default EXTDEST;
getn ALL.DBCON;
inext 'BP'; invers 1;
default TACOP;
getn ALL.DBCON; geton .DBCON;
inext 'BP'; invers 2; outvers 1;
default EXTDEST;
getn ALL.DBCON;
inext 'BP'; invers 2;

Appendix C: Resolved calibrators and manual setting of flux densities (alternative section 6.1).

C.1) Resolved calibrators: identification and treatment

You can run `uvplt` on your PT-Cal and PHS-Cal with `getn[ALL.DBCON.1]`; `sources'[PT-Cal]'`; `bparm 0`; `xinc _some_random_number_around_100`; `dotv 1` to check the distribution of amplitudes as function of uv-distance. If the amplitudes are not constant as a function of the uv-distance, your calibrator is resolved / confused and you will have to restrict the uv-range when using CALIB to the range of uv-distance over which the amplitudes are constant.

- Case of resolved PT-Cal (other than OQ208 and 2136+004):

Ask a member of the e-MERLIN team to provide you another calibrator. Standard calibrators are regularly observed on a daily basis. Alternatively, if your PHS-cal is bright enough and unresolved, you can use it as a PT-Cal.

- Case of resolved PHS-Cal (very rare):

Contact a member of the e-MERLIN team.

C.2) Library of calibrators

- C.2.a) FX-Cal: 3C286

1331+305	=.J2000	13h31m08.287984s 30d30'32.958850"
1328+307	=.B1950	13h28m49.657700s 30d45'58.640000"
.BAND	.FLUX(Jy)	.UVmax (kl)
=. 20cm (L)	=.15.00	=. 610
=. 6cm (C)	=.7.47	=. 165
=.1.3cm (K)	=.2.59	=. 40

(coordinates and fluxes are from the JVLA calibrator manual)

- Notes:

  - In L (C) band, you should restrict the UV-range to 0-610 kl (0-165 kl, resp.). You can run `uvplt` with `getn[ALL.DBCON.1]`; `sources'[FX-Cal]'`; `bparm 6,7,0`; `xinc 127`; `dotv 1` to check the uv-coverage. Use bparm 6,7,1,-610,610,-610,610,0; xinc 127 if you want to see the inner uv-coverage.
  - In this figure, each baseline is represented by a pair of tracks, symmetric with respect to the origin of the UV-domain. When source elevation is low, the inner part of the uv-domain is more filled.
  - Ensure you have at least 1 or 2 baselines (2 is preferable) covering the inner part of the uv-domain.

- C.2.b) PT-Cal: OQ208, 2136+004 (low declination targets, TBC)

OQ208 is a point source at all e-MERLIN bands and frequencies, and therefore requires no UV-range restrictions.

1407+284 =.J2000 14h07m00.394410s 28d27'14.689980" (a)
1404+286 =.B1950 14h04m45.615100s 28d41'29.235000" (a)

.BAND	.FLUX(Jy)	.UVmax (kl)	Spectral index (nu+alpha)
20cm (L)	=.0.80 (a)	=. unresolved by e-MERLIN	alpha = 2.3/5.7 (b)
6cm (C)	=.2.80 (a)	=. unresolved by e-MERLIN	alpha = -1.1/-0.7 (b)

(a) from the JVLA calibrator manual
(b) from Xie et al. 2005, SSA/FFA+beaming model (http://cdsads.u-strasbg.fr/abs/2005ApJ...621L..13X)

- Notes:
  - The accuracy of the flux calibration greatly depends on a good derivation of the spectral index of OQ208. Check that the spectral index you found in 8.1.c is consistent with the values listed in the above table. Please, report to e-MERLIN staff the flux densities and spectral index you found for OQ208, as we try to keep a record of these values.

- C.2.c) 3C84

3C84 is used as a zero-polarisation calibrator.

0319+415 =.J2000 03h19m48.160102s 41d30'42.103050"
0316+413 =.B1950 03h16m29.567300s 41d19'51.916000"

.BAND	.FLUX(Jy)	.UVmax (kl)
=. 20cm (L)	=.23.9	=. unresolved by e-MERLIN
=. 6cm (C)	=.23.3	=. unresolved by e-MERLIN

(coordinates and fluxes are from the JVLA calibrator manual)

- C.2.d) VLBI Catalogues

If your calibrators are not among the above, check out:

- http://www.vla.nrao.edu/astro/calib/manual/csource.html
  - http://www.vla.nrao.edu/astro/calib/manual/boot.html

to know restrictions that apply.
You can also find some useful information on calibrators in the VLBI catalogue:

- http://www.vlba.nrao.edu/astro/calib/vlbaCalib.txt
  - http://www.merlin.ac.uk/user_guide/collect_cal_flux.txt

C.3) Update flux densities in SU table

The SU table contains the flux densities of all the calibrators. To update the flux densities of a calibrator (example is given here for 3C286), update the source name and corresponding flux densities and run the following procedure:

```
default SETJY; getn [ALL.DBCON.1];
sources '1331+305', '';
optype '';
proc LOOP_SETJY
array tab_zerops(8);
tab_zerops 7.530 7.468 7.407 7.347 7.289 7.231 7.175 7.119;
for i = 1 to 8;
bif i; eif bif;
zerosp tab_zerops(i),0,0,0;
```

go;
wait;
end;
FINISH
LOOP_SETJY

- Notes:
  - For relative flux calibration, it is not required to update the flux densities of the PT-Cal. You can simply use 1.0 Jy as the reference flux for your PT-Cal. Then, use GETJY to derive the flux density of your PT-Cal.
  - For a non-polarised calibrator, only the "I stokes" column of your scan file should have non-zero values.

Appendix D - tuning the antenna weights.

Here, we explore the parameter space of the antenna weights. Reweighting antennas will inevitably affect the resolution and sensitivity to specific spatial scales: weighting up antennas involved in short baselines (Lovell, Mk2, Pickmere) will result in an increasing of the beam size and sensitivity to extended structure at the expense of resolution; conversely, weighting up antennas involved in long baselines (Cambridge) will result in a higher resolution at the expense of sensitivity to extended structure. Therefore, users should always modify the weights in accordance with their science goals. However, we insist that, here, the main purpose of re-weighting the data is to take into account the performances of antennas relatively to each other. Data weighting for science purposes (sensitvity VS resolution) will (later) be done when imaging (using the uvwtfn and robust parameters in `imagr`).
To determine the antennas and intervals of variations that will be tested, we suggest to use the task `uvhgm`. `uvhgm` can plot the distribution of flux densities for specific antennas/baselines.

- Run `uvhgm` with `getn[TAR.WTMOD];source'[Target]'; stokes'HALF'; antennas 1,0; doall 1; axtype'H';opcode'CLIN';do 1`. Repeat for the other antennas (`antennas 2.0` etc).

- Notes
  - This task will fit the distribution of points and will return the statistical dispersion of the distribution (i.e., the noise for all the baselines associated with a specific antenna).
  - Note the statistical dispersions associated to each antenna. Normalise them to the statistical dispersion associated to MKII. The relative weights that should be applied, based on the fits done by UVHGM, are the square roots of the normalised statistical dispersions.
  - However, UVHGM uses Gaussian fits, while, in most cases, the distribution is not Gaussian (extended wings can be seen). So the relative weights determined by this technique are not necessarily accurate. At least, by comparison with the nominal weights, they give a clue in which direction (up or down) the relative weights of the antenna should be adjusted. In general, adjusting only the following weights should produce some significant improvement in the noise level: Lovell, spanning the range [5:30/50], by step of 5; Cambridge, spanning the range [1.74/4.8:10], by step of 2; lowering the weight of any antenna for which a lot of data were flagged.

Explore the parameter space by running the following procedure. Do a "recat" of your catalogue and simply copy/paste the code.
Make sure the source name corresponds to your target and adjust the intervals and steps of weights you want to span (see comments)
default NAMEGET;
default WTMOD; inclass 'SPLAT'; NAMEGET; tput;
default IMAGR;
source '[TARGET]',";
docal -1; gainuse 0; doband -1; bpver -1;
freqid 1; flagver 0;
nchav 128;
cellsize 0.015; imsize 512;
minpa 255; factor -0.3; dotv -1;
nfield 1; do3dimag 1; rashift 60,0;
uvwtfn 'NA'; niter 0;
tput;

```
PROC WEIGHTS
FOR i = 0 to 50 by 5; $ LOVELL weights, adjust interval and step if necessary
FOR j = 0 to 10 by 2; $ Cambridge weights, adjust interval and step if necessary
tget WTMOD;
antwt 5+i, 1.0, 0., 0., 1.70, 0.10, 1.50, 1.70, 2+j, 0.; $ C-Band weights, adjust interval and step if necessary
$ antwt 5+i, 1.0, 0., 0., 0.73, 0.61, 0.77, 1.00, 2+j, 0.; $ L-Band weights, adjust interval and step if necessary
outname 'LO'!!char(5+i)!!'_CA'!!char(2+j);
go; wait;
default NAMEGET;
default INDXR;
inname 'LO'!!char(5+i)!!'_CA'!!char(2+j); inclass 'WTMOD'; NAMEGET;
go; wait;
default NAMEGET;
tget IMAGR;
inname 'LO'!!char(5+i)!!'_CA'!!char(2+j); inclass 'WTMOD'; NAMEGET;
outname inname;
go; wait; recat;
END;
END;
FINISH
WEIGHTS
```

- Notes
- This may take quite a long time, so you can have a break !
- Check the noise levels in all maps and determine the best set of parameters.
- Delete all the WTMOD files (and associated maps) apart your best one, as they take quite a lot of space

Appendix E: multi-scale cleaning

MULTI-SCALE CLEANING is adapted for extended and/or patchy sources. You define a set of beams, each with a different size, to deconvolve the dirty image with. The different spatial scales that characterise the emission of your target are then better recovered. This option produces as many images as the number of beams you specified, i.e., one image for each resolution. Only the image labelled .ICL001 is a multi-scale cleaned image. NGAUSS is the parameter that turn the multi-scale cleaning option on. WGAUSS specifies the different beam sizes and FGAUSS the different thresholds. The only other parameter you have to worry about is imagrprm(11), it controls the switch from one beam to another, by weighting the peaks in the various images. imagrprm(11) must be chosen in the range [0,1] and depends on the source structure: the higher imagrprm(11), the smaller the weights of the extended emission. This parameter should be tuned experimentally by testing different values and keeping the one that give the highest total flux density for the target. imagrprm(11) = 0.5 represents a good starting value to proceed to the SC of the target. Just keep in mind that you'll have to tune it for your very final imaging.

- 1) By observation of the image of your target, define the various scales over which you want to clean. For instance, if you want to clean the image on the highest resolution scale (taken as 0.05") , then 3 times and 6 times this resolution, you would choose `wgauss 0,0.15,0.3,0;ngauss 3`.
- 2) Determine noise level at each resolution by running IMAGR with `getn[TARGET.SPLAT];docalib -1;doband -1;cellsize 0.015;nchav 119;imsize 512;factor -0.3;dotv -1;uvwtfn'NA';outname'[choose name]'` and using TVALL/TVSTAT to measure the noise level.
- 3) Clean your map with IMAGR and the same parameters as for single-scale CLEANing, including multiple fields as necessary, but also with `imagrprm(11) 0.5`; `ngauss 3`;`wgauss 0,0.9,1.8,0` and `fgauss` updated with `n,n,n,0` where `n` is 2.5 times the rms found above.