

# User guide to the e-MERLIN data reduction pipeline

Megan Argo  
email: argo@astron.nl

January 10, 2012

## 1 Overview

This document is intended only as a guide to using the e-MERLIN data reduction pipeline (and associated scripts), it is not meant to be a data reduction tutorial. For more detailed information on calibrating data from interferometers, see the AIPS cookbook, the old MERLIN User Guide<sup>1</sup> or any of the other standard radio interferometry texts.

The pipeline scripts are written in Python and use ParselTongue to interface with AIPS<sup>2</sup>. To run them, you will therefore need working installations of AIPS, Python and ParselTongue. The scripts have been tested with AIPS 31DEC11 and Python 2.7. For the gory details on ParselTongue, including installation instructions, see the JIVE wiki<sup>3</sup>.

Currently, there is no all-in-one end-to-end pipeline since it is strongly recommended that the user edit their visibilities prior to averaging. There are in fact three scripts which should be executed in sequence: the first loads the data from disk (or directly from the e-MERLIN archive if you are on the JBCA computer system), the second averages and concatenates your data files to create one UV database, the third is the calibration pipeline. It is recommended that you edit spectrally (e.g. with SPFLG) between the first and second steps, and temporally (e.g. with IBLED) between steps two and three.

---

<sup>1</sup>The MUG: [http://www.merlin.ac.uk/user\\_guide/](http://www.merlin.ac.uk/user_guide/)

<sup>2</sup>See <http://www.aips.nrao.edu/>

<sup>3</sup><http://www.jive.nl/dokuwiki/doku.php?id=parseltongue:parseltongue>

Note that the sections in the old MERLIN User Guide which refer to the `dprogs` are now obsolete for e-MERLIN data (although still required for processing old MERLIN archive data). Data from the new e-MERLIN correlator are written to the archive in FITS format which can be loaded straight into your favourite data reduction package. This pipeline uses Classic AIPS.

It is assumed that you are doing a fairly simple experiment using standard setups. If this is not the case, or if you have particular requirements, you may have to carry out some of the procedures described here by hand. In any case, it is highly unlikely that the images produced by the pipeline will be of publication quality and it is recommended that you re-image the data once you are satisfied with the calibration.

## 2 Loading your data

The e-MERLIN correlator produces one file per source, per setup, per directory. More than one directory may be created per calendar day, depending on file sizes. The directories are located on the archive at JBCA. In order to process your data, it needs to be loaded into AIPS and a few standard procedures run before any calibration is attempted. You can, of course, do this by hand, but the scripts described here will take care of it automatically and should free the user to work on other things while the data are loading.

There are two load scripts, select the appropriate one for your location. Both scripts load the selected data and carry out some necessary housekeeping. In either case, your files can be called anything you like since the scripts will rename them based on the source name as specified in the header.

If you are using the JBCA computer system directly (either onsite or remotely) and will be loading data directly from the archive, then you should use `eMERLIN_loadlocal.py`. The script will ask you for your AIPS number and which AIPS disk to load to. It also checks that it can see the archive; if it cannot, it exits with an error message. You can specify dates to load either by giving a text file containing a list of dates as an argument when invoking the script:

```
argo@dop250> parseltongue eMERLIN_loadlocal.py dates.txt
```

where `dates.txt` contains a list of dates in `yyyymmdd` format, one per line. If no file is specified, the script will ask you for dates interactively in the same format. The script searches the e-MERLIN archive on the Jodrell system for

the specified dates and loads anything with the extension `.fits` to the AIPS number and disk specified by the user. For each file found, the script will run FITLD, UVFIX, MSORT and INDXR, tidying up the AIPS disk when loading is complete.

If you are not at JBCA or using the JBCA computer system, then you should use `eMERLIN_load.py` instead. This script will load the specified data and carry out the same housekeeping procedures as described above, but can be used on any computer. The script will ask you for your AIPS number and which AIPS disk to load to. For this script to work, your data should be in one or more directories, named as they are on the archive (e.g. `/scratch/jaglan_1/M51/20110613.1/`). Put the script in the base directory where your data are located (e.g. `/scratch/jaglan_1/M51/`) and invoke as:

```
argo@dop250> parseltongue eMERLIN_load.py
```

which will load any `.fits` files that it finds.

Due to AIPS issues, these scripts will not be able to load files with names longer than 46 characters. They will log all messages to `eMERLIN_load.log` (clobbering any messages from previous runs).

Note that both of these scripts will load all `.fits` files that they find in the given directories. If you have data taken with more than one setup, it would be a good idea to separate it on disk so that you only load data from one setup at a time. If you load everything to one AIPS number at this point, it will cause problems when you come to run the DBCON script later.

Following the completion of either of the loading scripts, you should flag your data before proceeding. It is a good idea to examine each file in SPFLG in order to find and remove any RFI. Use your favourite editing tasks to examine the data and determine if there were problems for any particular telescope, polarisation or time range.

### 3 Averaging and concatenating

At this point you should average your data in time and, unless you are doing spectral line work, also in frequency. Again, you can of course do this by hand, but a script exists to automate the averaging and combining of eMERLIN datasets to make life slightly more pleasant. Before running this script, your data should have been UVFIX-ed, MSORT-ed, INDXR-ed (e.g. by using one of the load scripts described above) and flagged.

The script, `eMERLIN_dbcon.py`, will ask you for your AIPS number, the

AIPS disk where your data are located, the desired number of spectral channels in your final database and how much time averaging you want. If you want no time averaging, enter `-1` at the appropriate prompt. If you want no spectral averaging, enter `-1` at the appropriate prompt. You should generally average to somewhere between 128 and 32 channels, depending on the field of view required.

The script queries the specified AIPS catalog and exits if any stale files from a previous run are found, so delete any pre-existing files of class “SUB SP” or “DBCON” before you run it. It will also examine the headers of each file in your AIPS catalog and will exit if it finds files with headers which do not match in terms of either the observing frequency, or the number of Stokes parameters, frequency channels or IFs, since a mis-match in any of these would cause DBCON to fail.

Assuming everything is ok, the script will average in time with UVAVG (if requested), average spectrally with AVSPC (if requested), then combine the resulting files into a single multi-source database in a logical way with DBCON, finishing with a final run of MSORT and INDXR. It will log all messages to `eMERLIN_dbcon.log`, clobbering any messages from previous runs. To save on disk space, the unaveraged files are deleted during the process, as are the intermediate DBCON databases, leaving just the final DBCON database. It applies any flag tables, so if you want to keep them you should back them up outside of AIPS **before** running this script.

Following the successful completion of this script, your data should be ready for calibration.

## 4 The calibration pipeline

The pipeline, `eMERLIN_pipeline.py`, will apply standard calibration procedures to your dataset, including amplitude calibration, phase referencing, bandpass calibration, optional self-calibration cycles and imaging. It requires the file `eMERLIN_tasks.py`, which contains various task definitions, to be present.

The script can either load a file from disk, or process a file already loaded in which case it assumes that your AIPS catalogue contains only one file, the flagged output from the `eMERLIN_dbcon.py` script, and will complain if it finds other files.

There are various options which are controlled via an input file which

should be specified as a parameter on the command line. Details of the inputs (both required and optional) are given below. After checking the inputs, the pipeline first deletes all SN, PL, BP and CL(>1) tables, so if you have already done some processing and want to keep your tables you are advised to back them up outside of AIPS before running the script. The pipeline then runs an initial fringe fit before calibrating using standard procedures.

The script carries out the following procedures: (optionally) load a multi-source fits file from disk, (optionally) MSORT and INDXR, output some pre-calibration diagnostic plots, fringe fit the data, calibrate the gain and phase, perform a bandpass calibration, (optionally) iterate selfcal on the phase calibrator, and finally map both the phase calibrator and source. It also makes various diagnostic plots along the way which it will plot to disk.

The pipeline requires a number of bits of information which it takes from a text file specified on the command line. An example input file (`calib.inputs`) with explanations is located with the scripts and in section 4.6. Most parameters are optional (although recommended) and will be set to sensible defaults if they are missing from the inputs file. You **must** specify your AIPS number, disk, sources and pointcal fluxes. Note that, unlike for the old MERLIN pipeline, there is no longer an option to specify continuum/wide-field/spectral imaging. Averaging should have been carried out by the `eMERLIN_dbcon.py` script (or by hand if you prefer), the pipeline carries out a bandpass calibration on all datasets and no further averaging is done here.

## 4.1 Before running the pipeline

By this point, your data should have been loaded, UVFIX-ed, MSORT-ed, INDXR-ed (e.g. by one of the load scripts described above, or by hand if you're feeling masochistic), flagged in frequency (using your favourite methods), AVSPC-ed, DBCON-ed, MSORT-ed, INDXR-ed (the DBCON script carries out these procedures), optional final IBLED/EDITA/whatever and be ready for calibration. Note that, while averaging is entirely optional, things will go **much** faster if your data have been averaged in time and/or frequency.

## 4.2 Running the pipeline

Once your data have been prepared and flagged, invoke the pipeline with the following incantation:

`argo@dop250> parseltongue eMERLIN_pipeline.py calib.inputs`  
where `calib.inputs` is a plain text file giving a set of input parameters for the script (the file can be called anything you like). There is an example inputs file in section 4.6. Only a few parameters are required, everything else is optional and will be set to sensible defaults if they are missing from the inputs file.

The script will first check the specified AIPS catalogue for existing files from a previous run, exiting with a complaint if it finds more than one file. Assuming it finds just one file, it then removes stale calibration tables (any SN, BP, PL tables, and any  $CL > 1$ , so make a backup if you want to keep your existing tables) before proceeding with calibration.

The basic outline of the process is as follows:

- (optionally) load DBCON-ed data file from disk
- (optionally) MSORT and INDXR
- output some pre-calibration diagnostic plots
- initial fringe fit on everything but the target(s)
- CLCAL the solutions
- plot some diagnostics
- SETJY on the pointcal
- CALIB (phase only on the phase cal)
- CLCAL the solutions
- CALIB (amp and phase)
- GETJY
- CLCAL the solutions
- bandpass calibration
- (optionally) iterate selfcal on the phase calibrator
- map the phase cal and source

There is no option for continuum/wide-field/line imaging since it is assumed that the data contain more than one channel. A bandpass calibration is carried out and wide-field imaging is carried out by default.

The pipeline generates numerous diagnostic plots as it runs and it would be a good idea to check them to make sure things are proceeding in a sensible way. If you do not specify a location for them, they will be written to the current working directory.

### 4.3 Calibrator information

Note that the e-MERLIN system currently lists 3C286 and OQ208 by their IAU designations (1331+305 and 1407+284 respectively). The name of each source given in the inputs to the pipeline must match those in the source table of your FITS file.

For channel 1 in IF 1 at 4128 MHz on the Mk2–Pi baseline we (currently, Summer 2011) expect the following:

IF1	3C286 =	7.705	Jy	OQ208 =	2.49	Jy
IF2		7.558			2.48	
IF3		7.416			2.47	
IF4		7.279			2.44	

Table 1: e-MERLIN calibrator information. See e-MERLIN staff for the most recent numbers.

### 4.4 Pipeline inputs

Once your data have been prepared and flagged, invoke the pipeline with the following incantation:

```
argo@dop250> parseltongue eMERLIN_pipeline.py calib.inputs
```

where “calib.inputs” is a plain text file giving a set of input parameters for the script (the file can be called anything you like). There is an example inputs file below. Only a few parameters are required, everything else is optional and these will be set to sensible defaults if they are missing from the inputs file.

Required inputs:

```
userno = 667                # AIPS user number
indisk = 1                  # AIPS disk to use
targets = M82              # comma-separated lists of target source(s)
phsrefs = M82A            # comma-separated list of phase cal(s)
fluxcals = 1331+305       # Flux cal(s)
pointcals = 1407+284      # Point cal
bpasscals = 0555+398     # Bandpass cal
pointflux = 2.49,2.48,2.47,2.44 # Point cal flux list for each IF (Jy)
```

Optional inputs:

```
refant = Mk2          # Reference antenna - this defaults to Mk2, but
                      # obviously this is not always possible. The pipeline
                      # will check if the antenna is present, if not it will
                      # select one for you. Choices are: Mk2, Pi, Da, Kn,
                      # De, Cm, Lo.

solint = 10          # Specify the solution interval in minutes
dosort = 1           # Do we need to do a sort? Runs MSORT and
                      # INDXR. Defaults to 1. If you are sure your data
                      # are already sorted, turn this off (dosort = 0) to
                      # speed up processing.

msgkill = -5         # How chatty do you want AIPS to be? Defaults
                      # to -5. Messages are printed to the terminal and
                      # AIPS messages are written to eMERLIN.pipeline.log
                      # in the current working directory.

doselfcal = 0        # Do selfcal on phase calibrator? 0=no, any
                      # other positive integer = this many rounds of
                      # selfcal. Default is 0.

imsize = 1024        # Parameters for imaging. The imsize
                      # parameter only applies to your target field,
                      # not the calibrators.

cellsize = 0.02      # Parameters for imaging. Note that the
                      # cellsize will be reset if you set it outside a
                      # sensible range for the frequency given in the
                      # FITS header.

plotdir = /my/dir/   # Where to dump the plots if not in
                      # interactive mode, defaults to the current
                      # working directory if not specified.

fitsfile = data.fits # If your data require loading, name of
                      # fitsfile on disk.

fitsdir = /my/dir/   # If your data require loading, location
                      # of fitsfile on disk. Defaults to the current
                      # working directory.

fring_snr = 9        # SNR threshold for FRING, defaults to 9.
```

## 4.5 Things you need to know

The scripts and inputs should be fairly self-explanatory and will mostly run without user interaction. They should be fairly robust, as long as you do not try using the same AIPS number while any of the scripts are running, but if there are any problems then check here first.

1. The file `eMERLIN_tasks.py` contains numerous AIPS task definitions and is required by `eMERLIN_pipeline.py`.
2. When running `eMERLIN_pipeline.py`, if there is more than one file in the specified AIPS catalogue, the script will complain and quit. If you requested that the pipeline load a fits file from a directory on the filesystem, it will quit if the specified AIPS disk is not empty.
3. If you have tables (e.g. from a previous calibration) that you want to keep, make a backup outside of AIPS before running `eMERLIN_pipeline.py` because it will delete old tables before starting.
4. Diagnostic and contour plots from the pipeline will (if output directory is not specified) be dumped to the current working directory. It is a good idea to check them.
5. If you have more than one target/phscaf pair, you can specify them as comma separated lists in `calib.inputs`.

## 4.6 Example `calib.inputs`

```
#####  
# Example inputs for the e-MERLIN calibration pipeline. #  
#####  
  
#####  
# Required inputs #  
#####  
  
# AIPS info  
userno = 667  
indisk = 1  
  
# Lists of sources to process. For multiple sources you can  
# specify comma-separated lists of phsrefs and targets.
```

```

targets = 1512+470
phsrefs = 1500+478
fluxcals = 1331+305
bpasscals = 0555+398
pointcals = 1407+284

# List of fluxes to use for your pointcal defined above
pointflux = 2.4885, 2.4705, 2.4580, 2.4305

#####
# Optional inputs # - recommended, but with sensible defaults
#####

# Reference antennas. Choices are: Mk2, Pi, Da, Kn, De, Cm, Lo
# The default refant is Mk2 (if in the array)
refant = Mk2

solint = 10

# Do we need to do a sort?  Runs MSORT and INDXR, defaults to 1.
# If you are *sure* your data are already sorted, turn this off
# (dosort = 0) to speed up processing.
dosort = 0

# How chatty do you want AIPS to be?  Defaults to -5.
msgkill = -5

# Do selfcal on phase calibrator? 0=no, any other positive
# integer = this many rounds of selfcal. Default is 0.
doselfcal = 3

# Parameters for imaging. Note that the cellsize will be reset
# if you set it outside a sensible range for the frequency
# given in the FITS header. The imsize parameter only applies
# to your target field.
imsize = 2048

```

```
cellsize = 0.02

# Where to dump the plots if not in interactive mode.
# Defaults to the current working directory
plotdir = /home/user/e-MERLIN/data

# Directory containing fitsfile to load if your data are not
# already loaded in AIPS. If the file is in the current working
# directory, you only need to specify fitsfile.
fitsfile = myfitsfile.fits
fitsdir=/home/user/e-MERLIN/data/

# snr for fring, defaults to 9
fring_snr = 9
```