LBWG memo 20

Pre-processing in AIPS

Neal Jackson, 2019.04.23

Introduction. AIPS is not the normal pre-processing system; you are encouraged to use the official long-baseline pipeline. In some cases - particularly if you are familiar with AIPS, or want to fiddle with the parameters of the fringe-fit - you may want to use AIPS for some tests. This is a brief set of suggestions for the best way to do this, including workarounds for things that otherwise do not work, and frequently-encountered bugs.

It is assumed that you have started with a measurement set from which you have made FITS files containing averaged data around one or more sources. See the long-baseline tutorial, or the NDPPP documentation, for more details on how to do this. Note that if you are doing your own parallelisation you need `numthreads=1` in the NDPPP parset.

Procedure

1. Read the data into AIPS with `FITLD`. You will need to create CL tables for each file using `INDXR`. The defaults for this task are totally unsuitable for LOFAR LB data, since they assume that the atmosphere varies slowly and create by default CL tables with 5-minute intervals. I recommend using 5-second intervals, i.e. use `CPARM(3)=5.0/60.0` .

2. If your bandwidth is more than about 20 subbands, you will need to split all the data into IFs, because AIPS does not do dispersive delays. Use `MORIF` for this. A full dataset will need to be split into about 12 IFs, or more if the ionosphere is bad.

3. Do the fringe-fit on the best calibrator, using the phased superterp as reference antenna. If you have a model, read it in and use this, otherwise use a point source and be prepared to repeat the fringe-fitting and calibration steps. Typically you will need a delay solution with a solution interval of 1, or a few, minutes. I recommend `APARM(7)=2` (SNR of 2 in the solutions), `DPARM(2)=500` (500-ns fringe search, should be enough for most conditions) and `DPARM(8)=5`. The latter will zero the rates and phases - i.e. we are calibrating in two steps.

4. You will almost certainly need to smooth the delay solutions with `snsmo`. I have found a smoothing time of about an hour and a clipping of 100-150 ns to work well (these are set with `BPARM` and `CPARM`).

5. Interpolate the delay solution witn `CLCAL` **and check the resulting phases with SPFLG and docalib=1** for each of the baselines to the reference antenna. The phases will be incoherent from one IF to another, but they should be flat across an IF.

6. Run `CALIB` to deal with the phases, using `docalib=1` and a solution interval of about 30 seconds. Look at the solutions, and if happy, interpolate with `CLCAL`. **Again check with SPFLG** - now your phases should have structure only from the calibrator source (or be flat if the calibrator is a point source). *Note that for SPFLG, you will need to delete the .SPFLGR. file before re-running SPFLG on the same dataset, otherwise you get the same plot again.*

7. If your calibrator is not a point source, make an image to use as a model, and repeat steps 2-6.

8. Copy the CL table with your best calibration to any other MORIF'd datasets with TACOP. Then use SPLIT to make single-source files for each source/direction.

9. Plot each dataset as needed with UVPLT (all defaults) and edit any high-amplitude points with CLIP, producing a flag file.

10. Use UVCOP to apply the flags (with FLAGVER=n where n is the highest-number flag file). This is important because AIPS does not by default write regular FITS files (they do not have the same number of baselines per time stamp). This causes CASA or other import programs to refuse to read them into measurement sets. UVCOP appears to write a file which is then written out as a regular fits file. If it doesn't work there is a Parseltongue script called checkregular.py on github.com/nealjackson which can be used instead.

11. Write out with FITTP and read into a measurement set with CASA.

12. Run clearcal in CASA to create a corrected-data column.

13. AIPS writes the antenna tables with '1', '2' etc. as antenna names instead of the station names. In order to correct this (and be compatible with further pipeline steps) you are likely to need the following from within CASA:

```
tb.open('mydata.ms',nomodify=False)
tb.putcol('NAME', tb.getcol('STATION'))
tb.close()
```