

Author: S.J. Wijnholds	Date of issue: 2011-Jan-03 Kind of issue: limited	Scope: LOFAR project Doc.nr.: LOFAR-ASTRON-MAN-049	
	Status: draft Revision nr.: 0.1	File:	

Station Calibration Cookbook

Stefan J. Wijnholds

Verified:			
Name	Signature	Date	Rev.nr.
TBD!	0.1

Accepted:		
Work Package Manager	LOFAR Operations Manager	LOFAR Project Manager
S.J. Wijnholds	A. Polatidis	M.P. van Haarlem
..... <i>date:</i> <i>date:</i> <i>date:</i>

©ASTRON 2011
All rights are reserved. Reproduction in whole or in part is prohibited without the written consent of the copyright owner.

Author: S.J. Wijnholds	Date of issue: 2011-Jan-03 Kind of issue: limited	Scope: LOFAR project Doc.nr.: LOFAR-ASTRON-MAN-049	
	Status: draft Revision nr.: 0.1	File:	

Distribution list:

Group:	For Information:
LOFAR operations A. Asgekar M.A. Brentjens W. Frieswijk G. Kuper R.A. McFadden R. Pizzo A. Polatidis J.J. Sluman Y. Tang	LOFAR project M.P. van Haarlem H. Munk R.J. Nijboer C. Vogt - van Haarlem M. Wise

Document revision:

Revision	Date	Section	Page(s)	Modification
0.1	2011-Jan-03	-	-	Creation

Author: S.J. Wijnholds	Date of issue: 2011-Jan-03 Kind of issue: limited	Scope: LOFAR project Doc.nr.: LOFAR-ASTRON-MAN-049	
	Status: draft Revision nr.: 0.1	File:	

Abstract

This station calibration cookbook provides a detailed description of how to do station calibration observations, reduce the resulting calibration data and produce station calibration tables. The reader is assumed to be familiar with the station calibration strategy adopted for initial LOFAR operations, so this guide will explain how station calibration is done, but not why it is done that way.

Author: S.J. Wijnholds	Date of issue: 2011-Jan-03 Kind of issue: limited	Scope: LOFAR project Doc.nr.: LOFAR-ASTRON-MAN-049	
	Status: draft Revision nr.: 0.1	File:	

1 Introduction

This document provides a step-by-step description that should allow the reader to do station calibration observations, reduce the resulting calibration data and produce station calibration tables. The reader is assumed to be familiar with the station calibration strategy adopted for initial LOFAR observations as described in [2]. This guide will therefore explain how things are done, but not why they are done that way.

The station calibration pipeline was developed in Matlab. The code is available in the `StationCal` toolbox developed by the author. This code is also compiled to the C++ shared library, that is used by the `CalServer` at the stations. Experience has learned that porting this code to C++ or Octave results in a significantly slower reduction process (a factor 5 – 10 in execution time). Porting may also result in slightly different results due to small differences in the configuration of underlying algorithms like the EVD making comparison between on-line and off-line calibration results more difficult. It is therefore advised and assumed, that Matlab is available on the machine on which the data are reduced.

The remainder of this document is split in three sections. The next section explains how station calibration observations are done. Section 3 describes how the station calibration data are organized in the station calibration data archive and how the data can be conveniently copied to and from this archive. The last sections explain how the data can be reduced and how problematic data can be analyzed.

2 Making station calibration observations

2.1 LBA station calibration

The calibration table for the LBA station array is based on a 24-hour monitoring campaign in which the station correlator continuously sweeps over all 512 subbands with one second integration time per subband [2]. During these observations, the autocorrelation spectra of the individual signal paths are stored with ten seconds integration for monitoring and diagnostic purposes. These observations are done in RCU modes 1 and 3, except for the European stations, which do not use RCU modes 1 and 2 and therefore do not require calibration in RCU mode 1. A detailed description is provided below to do these observations by hand. The observers have developed a number of scripts to set up these observations reproducibly and conveniently over multiple stations. Information on these scripts and potential future updates can be found on the LOFAR operations wiki [1].

Before starting the observation, we need to configure the `CalServer` to store the intra-station visibility data to a file. This is done by editing the `CalServer.conf` file in `/opt/lofar/etc`. In this file the `CalServer.DataDirectory` tag should point to the directory in which the data should be stored (the default is `/localhome/data`), and the `CalServer.WriteACCToFile` tag should be set to 1 (the default is 0). Once the observation is finished, these tags should be set to their default values. One should do this shortly after the observation to stop the `CalServer` from producing more data, otherwise the hard disk will be filled completely. The data directory should exist, before the observation is started.

To restart the `CalServer` and ensure a proper and reproducible state of the system, reset the station by

```
swlevel 1
swlevel 3
rspctl --regstat
```

Author: S.J. Wijnholds	Date of issue: 2011-Jan-03 Kind of issue: limited	Scope: LOFAR project Doc.nr.: LOFAR-ASTRON-MAN-049	
	Status: draft Revision nr.: 0.1	File:	

After entering the last command, wait until all RSP boards are properly initialized before setting the RCU mode (say 1) and enabling the RCUs by

```
rspctl --rcumode=1
rspctl --rcuenable=1
```

. The entire system should now be running in the specified RCU mode. To verify this, check the output of the following commands:

```
rspctl --rcu
rspctl --stati
rspctl --status
```

We can now start capturing the autocorrelation spectra by entering the following line:

```
nohup rspctl --stati --integration=10 --duration=86400
--directory=/localhome/data/rcumode_1_20101215 &
```

The directory name is, of course, just mentioned as an example. The `nohup` program ensures that the `rspctl` process will keep running if the connection to the station is closed or lost. The `CalServer` should already have been running since we stepped to software level 3 and should produce its first data file within nine minutes. To check that all processes are running, move to the data directory and verify that two types of time stamped data files are being written: files containing the subband statistics for each RCU (`*sst_rcu*.dat`) and files containing intra-station visibilities for each subband or array covariance cubes (`*acc*.dat`).

Once the observation is completed, the `CalServer.conf` file should be restored to its original state, as is the station. The latter is done by issuing the commands

```
swlevel 1
swlevel 6
```

after restoring `CalServer.conf`.

2.2 HBA station calibration

The HBA station calibration observation is very similar to the LBA station calibration observation with the added complexity that we want to have a strong source in the main beam of the tiles [2], i.e., we need to let the tiles track a source during the 6-hour calibration observation. As for the LBA station calibration, the observers have developed a number of scripts to run these observations reproducibly and conveniently on many stations [1]. Note that the station calibration observations should be done with the ring splitter turned off for all stations including the core stations, i.e., the station should act as a single HBA tile array. Below, we will describe the procedure as one would do by hand at the station.

Before starting the observation, reconfigure the `CalServer` by editing the `CalServer.conf` file as explained in the previous section. Once this is done, restart the system by

Author: S.J. Wijnholds	Date of issue: 2011-Jan-03 Kind of issue: limited	Scope: LOFAR project Doc.nr.: LOFAR-ASTRON-MAN-049	
	Status: draft Revision nr.: 0.1	File:	

```
swlevel 1
swlevel 3
rspctl --regstat
```

and wait until all RSP boards are properly initialized. Since we want to track a bright source, we need to set the tile beams using `beamctl`. This is done by running `station_HBA_cal_setupBeam.sh` in `/globalhome/lofarsystem`. One may need to edit this script to ensure tracking of the right source in the right RCU mode. Once the beams are set, all tiles should operate in the right RCU mode and receive a clear signal. To verify the proper operation of the system, check the output of the following commands:

```
rspctl --rcu
rspctl --stati
rspctl --stati=beamlet
rspctl --status
```

We can now start capturing the autocorrelation spectra by entering the following line:

```
nohup rspctl --stati --integration=10 --duration=21600
--directory=/localhome/data/rcumode_5_CasA_20101215 &
```

The directory name is, of course, just mentioned as an example. The `nohup` program ensures that the `rspctl` process will keep running if the connection to the station is closed or lost. The `CalServer` should already have been running since we stepped to software level 3 and should produce its first data file within nine minutes. To check that all processes are running, move to the data directory and verify that two types of time stamped data files are being written: files containing the subband statistics for each RCU (`*sst_rcu*.dat`) and files containing intra-station visibilities for each subband or array covariance cubes (`*acc*.dat`).

Once the observation is completed, the `CalServer.conf` file should be restored to its original state, as is the station. The latter is done by issuing the commands

```
swlevel 1
swlevel 6
```

after restoring `CalServer.conf`.

3 Transferring data from the stations to the archive

The station calibration data archive is currently located under `/caldata` on `lcs010`. Every station has its own data directory, which is named after the station, e.g., `CS001`. Each of these directories contains the data from all calibration observations for that particular station. Each observation is stored in a subdirectory, whose name contains the station name, the RCU mode and the date at which the observation was started. In the case of an HBA calibration observation, the name also includes the name of the tracked source. Examples of these directory names are thus `CS001_mode1_20100607` and `CS001_mode5_CasA_20101008`.

Author: S.J. Wijnholds	Date of issue: 2011-Jan-03 Kind of issue: limited	Scope: LOFAR project Doc.nr.: LOFAR-ASTRON-MAN-049	
	Status: draft Revision nr.: 0.1	File:	

The `/caldata` directory also contains the scripts `copy.sh` and `copy4reduction.sh` to facilitate copying the data from the station to the station calibration data archive and from this archive to the data reduction machine. The data are copied using `scp`, so distribution of `ssh` public keys will speed up the data transfer considerably by removing the user interaction (entering passwords).

The `copy.sh` script facilitates copying data for identical station calibration observations from multiple stations. Here follows a listing of this script:

```
#!/bin/sh
# copy files from calibration observations to archive
# SJW, 14 Oct 2010
station_name="RS406"
data="/localhome/data/rcumode_3_20101209/*"
suffix="_mode3_20101209"

for name in $station_name; do
  mkdir ${name}/${name}${suffix}
  scp -rp wijnholds@${name}C:${data} ${name}/${name}${suffix}/
done
```

The variable `station_name` should contain a space separated list of stations from which data should be copied, the `data` variable should specify the data files at the stations and the `suffix` variable should specify which suffix should be appended to the station name in the station calibration data directories.

The `copy4reduction.sh` script facilitates the transfer of specific data sets to the computer used to reduce the data. This script looks as follows:

```
#!/bin/sh
# copy files from calibration observations to archive
# SJW, 14 Oct 2010
station_name="DE602 DE603 FR606"
suffix="_mode5_CasA_20101202"
targetmachine="dop147.astron.nl"
targetdir="/dop147_0/wijnholds/data/caldata/"

for name in $station_name; do
  cd ${name}
  scp -rp ${name}${suffix} wijnholds@${targetmachine}:${targetdir}
  cd ..
done
```

The variable `station_name` should contain a space delimited list of stations for which data should be copied, the `suffix` specifies the suffix added to the station name of the data sets to be copied and `targetmachine` and `targetdir` specify the name of the machine and the directory on that machine to which the data should be copied.

Author: S.J. Wijnholds	Date of issue: 2011-Jan-03 Kind of issue: limited	Scope: LOFAR project Doc.nr.: LOFAR-ASTRON-MAN-049	
	Status: draft Revision nr.: 0.1	File:	

4 Reducing the data

Once the data are available on a data reduction machine with Matlab installed, the data can be inspected and reduced using the tools provided by the `StationCal` package. If this package is installed and found within Matlab's search path, a complete overview of all available functions can be obtained by typing `help StationCal` at the Matlab command prompt.

In the following two subsections, I give a detailed description of the regular process of extracting calibration information from the data and making calibration tables. When the calibration attempt is unsuccessful while nothing obvious went wrong during the observation itself, one may want to take a close look at the data or the calibration result to find the cause of this problem. The last subsection therefore suggests a number of steps that may provide a good starting point.

4.1 Starting the calibration pipeline

The calibration pipeline discussed in [2] is implemented in the function `calibrateData`. This script takes a struct variable as input, which should contain the following fields:

`dirname` the directory containing the calibration data
`bandstart` start frequency (in Hz) of the band pass filter. In RCU mode 6, for example, this value should be $170e6$.
`bandstop` stop frequency (in Hz) of the band pass filter. In RCU mode 6, for example, this value should be $230e6$.
`startfreq` start frequency (in Hz) of the Nyquist zone. In RCU mode 5 this is $100e6$ if spectral inversion is turned on, which should normally be the case if the BeamServer is used to track a source. However, if a non-typical observation is set up by hand and spectral inversion is not turned on, the start frequency should be $200e6$. The `calibrateData` function can deal with both situations.
`stopfreq` stop frequency (in Hz) of the Nyquist zone. Note that for an observation in RCU mode 5, this value ($100e6$) may be lower than the start frequency.
`srcsel` vector containing the 3C numbers of the sources that should be included in the calibration model. If one wants to include the Sun, one should put a zero in this vector. The first source mentioned will be used as flux reference and will have unit flux in the calibration solutions. For LBA observations, this value is typically set to `[324, 283, 88, 179, 0]`, which specifies a model containing Cas A, Cyg A, Tau A, Vir A and the Sun. For HBA observations, this value is typically a scalar specifying the source tracked during the calibration observation.
`posfile` antenna position file (`AntennaArrays.conf`) of the station
`arrayname` name of the array used during the calibration observation in the `AntennaArrays.conf` file. For regular station calibration observations, this is either `LBA_INNER` (RCU mode 3), `LBA_OUTER` (RCU mode 1) or `HBA` (RCU modes 5, 6 and 7).
`uvmask` mask to flag specific entries of the array covariance matrices by hand, for example to ensure that visibilities that are affected by crosstalk, are not used in the calibration.
`RFITol` tolerance for the RFI detector. Typical values are 1 for LBA observations and 5 for HBA observations.
`outfile` name of the file in which the calibration results will be stored. The standard naming convention in the calibration archive is the string `CalTable_` followed by the station number and the RCU mode and source tracked when applicable. Examples are `CalTable_001_mode1.mat` and `CalTable_501_mode5_CasA.mat`.

This struct is composed in the script `calconfig.m`, which also produces the command string `cmd` that can be used to start the calibration pipeline. This script should be edited to configure the pipeline.

Once the variables in the script `calconfig.m` are set to appropriate values, start a screen session. This will allow us to close the connection to the data reduction machine without interrupting the pipeline, which may run for more than a day when calibrating an LBA calibration observation from a European station. Once in screen, start Matlab using

Author: S.J. Wijnholds	Date of issue: 2011-Jan-03 Kind of issue: limited	Scope: LOFAR project Doc.nr.: LOFAR-ASTRON-MAN-049	
	Status: draft Revision nr.: 0.1	File:	

```
matlab -nodisplay
```

The `-nodisplay` option ensures that Matlab does not make a connection to the graphical interface (X11). That connection will break upon closing the connection with the data reduction machine, even if Matlab is running in a `screen` session.

In Matlab, the calibration observation can now be started by

```
calconfig
eval(cmd)
```

Matlab should now start producing progress information looking similar to this:

```
configuration found, parsing...
Processing file number 1 of 42
working on subband 1 of 235
optimization stopped after 2 iteration(s).
Elapsed time is 0.183551 seconds.
working on subband 2 of 235
optimization stopped after 2 iteration(s).
Elapsed time is 0.065810 seconds.
...
```

If you see an error message, use that feedback to check the setup specified in `calconfig`. If things are running smoothly, you can disconnect from the `screen` session (usually by `ctrl-A-ctrl-D`) and close the connection to the data reduction machine. If multiple CPU cores are available, multiple reduction processes can be started in parallel `screen` session. Appropriate changes have, of course, to be made to the `calconfig.m` script before starting another calibration process.

4.2 Making calibration tables

Once the data from a calibration observation is reduced, we are ready to produce a calibration table. First load the calibration result to the Matlab workspace by, e.g.,

```
load CalTable_001_model.mat
```

Such a file contains a struct `calres` with the following fields:

```
calx0 calibration factors per run, per subband and per element for the array of x-dipoles before solution based flagging
caly0 corresponding result for the array of y-dipoles
calx calibration factors per run, per subband and per element for the array of x-dipoles after solution based flagging
caly the corresponding result for the array of y-dipoles
sigmax estimated calibration source powers per run, per subband and per calibration source for the array of x-dipoles
sigmay corresponding result for the array of y-dipoles
```

Author: S.J. Wijnholds	Date of issue: 2011-Jan-03 Kind of issue: limited	Scope: LOFAR project Doc.nr.: LOFAR-ASTRON-MAN-049	
	Status: draft Revision nr.: 0.1	File:	

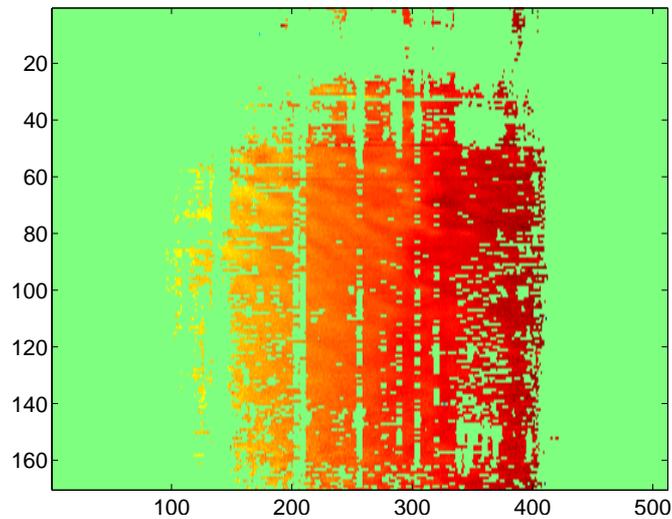


Figure 1: A typical (t, f) -plane for the phase solution of a single element. The data look smooth over time (vertical axis) and frequency (horizontal axis) and is not too severely flagged.

`AoverTx` estimate of A/T towards Cas A for the array of x -dipoles. This assumes that the first source in the source model is Cas A.

`AoverTy` corresponding result for the array of y -dipoles

`t_obs` reconstructed observing times for each snapshot observation

`freq` center frequencies for each snapshot observation

Before starting the construction of a calibration table, it is advisable to plot a time-frequency-plane of the phase solutions for a few antennas to check the quality of the solutions. Such a plot can be made using standard Matlab functionality:

```
imagesc(angle(calres.calx(:, :, 20)))
```

This should show a result that is smooth over time and over frequency and does not show too many outliers such as shown in Fig. 1. If a lot of data is flagged or if fringes are visible, one should do a closer inspection of the results and the data to judge whether the results are suitable for making a calibration table.

If the calibration results look ok, the `makeCalTable` script can be used to extract the values needed for the calibration table. The script needs to be executed once for each polarization, which can be specified in the script where it reads

```
% specify polarization ('x' or 'y')
pol = 'x';
```

Once the polarization is specified, the script can be started from the Matlab command prompt. This script will try to fit an amplitude and a phase model to the complex valued gain solutions by calling the `fitAmplModel.m` and `fitPhaseModel.m` scripts respectively. The first script, produces output that looks like this:

```
fit for antenna 1 based on 25883 data points
```

Author: S.J. Wijnholds	Date of issue: 2011-Jan-03 Kind of issue: limited	Scope: LOFAR project Doc.nr.: LOFAR-ASTRON-MAN-049	
	Status: draft Revision nr.: 0.1	File:	

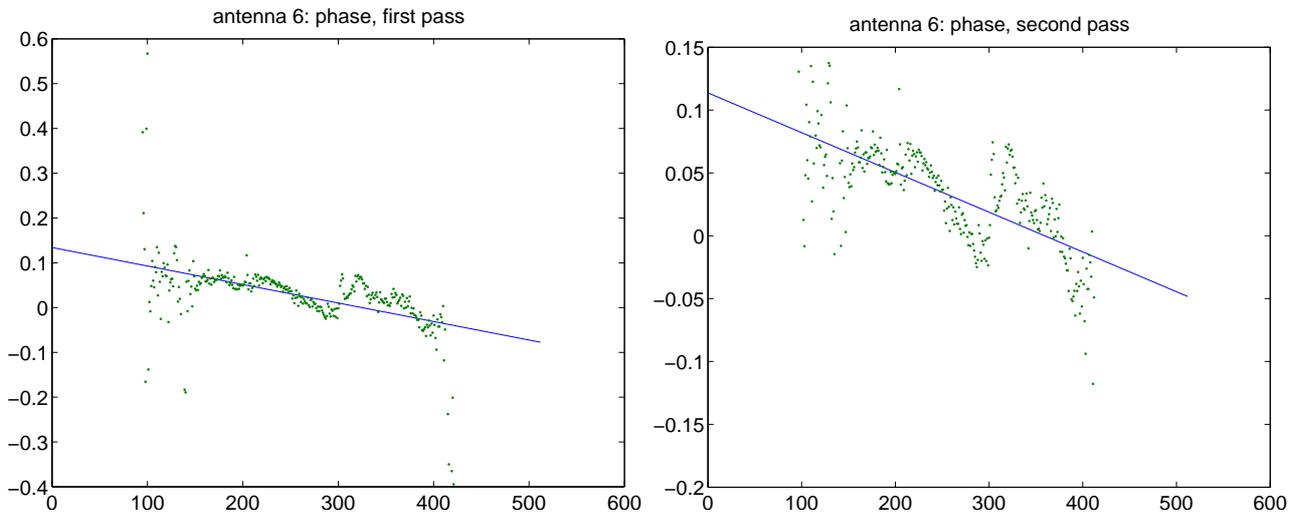


Figure 2: Example of a plot produced by `fitPhaseModel` showing the phase data used (dots) and the fitted phase model (solid line) before (left) and after (right) outlier removal.

```

fit for antenna 2 based on 26056 data points
fit for antenna 3 based on 26102 data points
fit for antenna 4 based on 26425 data points
fit for antenna 5 based on 26296 data points
fit for antenna 6 based on 26093 data points
fit for antenna 7 based on 26194 data points
fit for antenna 8 based on 26171 data points
fit for antenna 9 based on 26176 data points
fit for antenna 10 based on 26120 data points
fit for antenna 11 based on 0 data points
...

```

This shows how many data points are available for fitting the amplitude and phase model for each signal path over the entire calibration observation. A relatively low number may indicate a problem with a specific signal path, a zero implies that that signal path did not produce any useful result at all. The latter case certainly indicates a technical failure. The first case may do so as well and requires careful attention during the phase model fitting.

The script `fitPhaseModel.m` fits a phase model to the data by averaging the phase data over time for each subband and then fitting a phase model to these (at most 512) data points. After the first fitting attempt, it shows the fit and the data as shown in the left panel of Fig. 2. It waits for the user to press any key to confirm the correctness of the fit. It then attempts to improve the fit by removing outliers and shows the fit and the data without the outliers for confirmation as shown in the right panel of Fig. 2. Although these plots show that there are significant differences between the fitted linear phase model and the actual data, the deviations are quite small and therefore acceptable. The main goal of the current station calibration tables is to take care of residual delay effects and phase offsets, which account for most of the phase differences between the antennas within a station.

If all goes well, the user only has to confirm all the plots and the `makeCalTable.m` script will produce a matrix `calx` or `caly`, depending on the polarization specified, that can be used for the calibration table. However, sometimes the phase unwrapping works unsuccessful and needs manual removal of the points causing the problem. If this is needed, do the

Author: S.J. Wijnholds	Date of issue: 2011-Jan-03 Kind of issue: limited	Scope: LOFAR project Doc.nr.: LOFAR-ASTRON-MAN-049	
	Status: draft Revision nr.: 0.1	File:	

following steps

1. Abort the `makeCalTable.m` script using `ctrl-c`.
2. Set the `antstart` variable to the current antenna number
3. Use the plot and Matlab's zoom function to find the subband indices of the points causing the problem and collect these points in a vector `sbflag`.
4. Run `fitPhaseModel.m` to continue.
5. If more antennas need manual flagging, abort `fitPhaseModel.m` using `ctrl-c` and repeat steps 2 – 4.
6. Execute the last two lines of `makeCalTables.m` manually.

If a fit is bad or unreliable due to some problem that cannot be solved by flagging a few points, write down the antenna number and proceed to complete the script. Once the script is finished, these solutions can be flagged just like any other signal paths for which no or insufficient calibration data were available, namely by setting all calibration factors to one. This can be easily done from the Matlab command prompt. For example, to flag antenna 10 for the x -polarization, type

```
calx(10, :) = 1;
```

Once `makeCalTable.m` is completed for both polarizations, one should have two matrices, `calx` and `caly`, in the Matlab workspace that do not contain any Not-a-Number values. These matrices can be supplied as arguments to the `writeCalTable` function. This function writes a properly formatted calibration table that is directly usable at the station. The current naming convention in the station calibration table archive is a combination of the string `CalTable`, the station number and the RCU mode separated by underscores with extension `.dat`. Examples are `CalTable_001_model.dat` and `CalTable_406_mode5.dat`. In the end, each station requires five calibration tables, one for each of the RCU modes 1, 3, 5, 6 and 7.

4.3 Data inspection

If the station calibration fails, we generally want to know why. The `StationCal` package contains a the functions `acm2skyimage` and `inspectData` that are specifically included to support such efforts. Based on practical experience, the following steps typically help to find the problem:

- Examine the phase solutions over time and frequency. This can be done by, e.g.,

```
imagesc(angle(calres.calx(:, :, 20)))
```

If you see fringes, this may indicate the presence of a source that is not included in the model. It could also indicate a mistake in the observing time, but this is not likely, since all timestamps are produced and processed automatically. Finally, it could indicate an ill response of the RSP boards while the `CalServer` tries to scan over all subbands. If you see extreme phase wrapping, i.e., large phase gradients over frequency for a large number of antennas but not for all, this may indicate that the cable delays are not compensated directly. This may be caused by either not reading the `CableDelays.conf` or by a wrong version of `CableDelays.conf` at the station.

- If fringes are found in the phase solutions, a logical step is to image the visibilities for a few frequency scans. This can be done using the DFT imager `acm2skyimage`. Are Cas A and Cyg A mapped to the right locations? Are there additional sources? If so, where? This information generally provides useful clues to zoom in on the problem.

Author: S.J. Wijnholds	Date of issue: 2011-Jan-03 Kind of issue: limited	Scope: LOFAR project Doc.nr.: LOFAR-ASTRON-MAN-049	
	Status: draft Revision nr.: 0.1	File:	

- If unexpected sources are present or many data are flagged by the RFI flagger, the function `inspectData` may be used to inspect the subband statistics. This function will search for subband statistics files in a specified directory and plot (t, f) -planes for each file consecutively for inspection by the user. This provides useful information on the band passes and the RFI situation, including electric fences, lightning strokes, etc.
- If the aforementioned steps do not provide a solution, one may want to try and calibrate a few frequency sweeps by hand. This can be done by calling the functions used by the `calibrateData` function directly from the work space. This has the advantage, that all intermediate results can be inspected, which helps to localize the problem.

References

- [1] LOFAR project. LOFAR operations wiki. www.lofar.org/operations/doku.php.
- [2] Stefan J. Wijnholds. Overview of the Initial Production Version of the Station Calibration Pipeline. Technical Report LOFAR-ASTRON-MEM-263, ASTRON, Dwingeloo (The Netherlands), January 2011.