Data Processing School :: Exercise 08

Source directory	/data/lofarschool/data/Exercise-08
Contact person	Joris van Zwieten (zwieten -at- astron.nl)

Context

The goal of this exercise is to acquaint you with the calibration of LOFAR data using the BBS software package. BBS is intended to operate as an integral part of the standard imaging pipeline. Though it can be run manually, it was originally designed to be run in an automated fashion. You will probably be reminded of this fact now and then in the course of this exercise.

We encourage you to explore the software and try out things other than those presented here if you feel like it. We very much welcome any feedback and/or suggestions you may have. Just send me an email at zwieten -at- astron.nl. Good luck with the exercise!

Prerequisites

Basic knowledge of Python and the `ipython` shell, MeasurementSet format, `casaviewer` or any available `fits` image viewer, and a notion radio data reduction.

Description

The goal of this exercise is to get hands-on experience with the reduction of LOFAR data in general and with the BBS software package in particular. In the course of the exercise you will:

- Inspect an uncalibrated LOFAR image
- Perform a simple simulation with BBS
- Perform rudimentary self-calibration of LOFAR data using BBS
- Inspect calibration solutions
- Flag data based on the quality of calibration solutions
- Inspect a calibrated LOFAR image

Files & Directories

The following is a list of the files present in the data directory for this exercise (`/data/lofarschool/data/Exercise-08/`) along with a short description of each file.

```
/data/lofarschool/data/Exercise-08/
  |
  |-- image-plane-cal.parset    # BBS parameter set file (image-plane
calibration)
  |-- instrument                # Initial instrument parameter database
  |-- lioffen.clusterdesc       # Description of the offline cluster
  |-- make-image.sh             # Script to make an image
  |-- plot.py                   # Script to plot calibration solutions
```

```
  |-- simulation.parset         # BBS parameter set file (simulation)
  |-- sky.in                    # Initial sky model
  |-- solflag.py                # Script to flag data based on solution
quality
  |-- solplot.py                # Helper script for plot.py
  |-- uv-plane-cal.parset       # BBS parameter set file (uv-plane
calibration)
  |-- L2007_03463_SB10_compressed.MS    #
  |-- L2007_03463_SB11_compressed.MS    #
  |-- L2007_03463_SB12_compressed.MS    # LOFAR measurement sets
  |-- L2007_03463_SB13_compressed.MS    #
  |-- L2007_03463_SB14_compressed.MS    # Each set contains a single
  |-- L2007_03463_SB15_compressed.MS    # subband. You will only need
  |-- L2007_03463_SB16_compressed.MS    # one of these measurement sets
  |-- L2007_03463_SB17_compressed.MS    # for this exercise.
  |-- L2007_03463_SB18_compressed.MS    #
  |-- L2007_03463_SB19_compressed.MS    #
```

Step-by-step instructions

This section starts with a step-by-step guide to correctly setup the environment for this exercise. Next is a short description of the scripts that will be used regularly in the course of this exercise. Below that are the step-by-step instructions for the exercise itself. The topics 'Observed data' up to and including 'Image-plane clalibration' are self contained, so you can pick the ones that interest you most to save time.

Setting up the environment

Initialize the LOFAR software environment.

```
source /app/lofar/dev/lofarinit.csh
```

Initialize `pyrap` (the Python wrappings for `casacore`).

```
source /app/scripts/doPyrap
```

Create a private directory below `/data/lofarschool/users`. **NB.** Please *do not use spaces or special characters* to avoid quoting issues later.

```
cd /data/lofarschool/users
mkdir <your directory>
```

Copy all files needed for the exercise. Several measurement sets can be found under `/data/lofarschool/data/Exercise-08`. You'll only need a single measurement set for the exercise, so *just pick one*. Later you can look for participants who have reduced a different measurement sets and create an averaged (deeper) output image. The name of the measurement set used in these instructions is always `L2007_03463_SB10_compressed.MS`. Please replace it with the name of the measurement set you picked (starting with the copy command below).

```
cd <your directory>
cp /data/lofarschool/data/Exercise-08/* .
cp -r /data/lofarschool/data/Exercise-08/instrument .
cp -r /data/lofarschool/data/Exercise-08/L2007_03463_SB10_compressed.MS .
```

Create a measurement description (VDS) file.

```
makevds lioffen.clusterdesc L2007_03463_SB10_compressed.MS
combinevds L2007_03463_compressed.vds L2007_03463_SB10_compressed.MS.vds
rm L2007_03463_SB10_compressed.MS.vds
```

Scripts

This section contains a short description of two scripts that will be used regularly in the course of this exercise: `make-image.sh` and `calibrate`.

The `make-image.sh` script will be used to create images. The script takes three arguments: the measurement set name, the column name (one of `DATA`, `MODEL_DATA`, `CORRECTED_DATA`), and the output image name, respectively. The script will create both a `.img` (`casa`) image and a `.fits` image. Imaging should take approximately 5 minutes.

The `calibrate` script will be used to start BBS. If you run this script without arguments, it will print a short help text. Try it if you like. The following options should always be specified:

```
--cluster-desc lioffen.clusterdesc
--db cepmaster0
--db-user postgres
```

The `lioffen.clusterdesc` file contains a description of the offline processing cluster. It should be located in the same directory where you start the calibrate script, or you have to specify a path to it. The –`db` options specify the database server and user to use.

The `calibrate` script starts processes via `ssh`. When you start it for the first time, `ssh` may report that it cannot confirm the authenticity of the host and will ask if you want to continue connecting. Answer 'yes' to continue.

Running the `calibrate` script generates a large number of log files. You may want to delete them

from time to time as you work your way through the exercise.

Should you need to terminate the `calibrate` script (i.e. by hitting Control-C), make sure you also kill any remaining BBS processes:

```
killall GlobalControl KernelControl SolverControl
```

Observed data

The measurement set you picked is one of the 36 subbands (38-60 MHz) of the measurement called `L2007_03463.MS`. This measurement was performed on July 27, 2007. It is a 24-hour measurement (2861 timeslots of 30.2 s. each) using 16 single LBA (Low Band Antenna) *micro-stations*, targeted at the North celestial pole. The individual subband measurement sets were `RFI` flagged and compressed to a single frequency channel.

We will start by imaging the raw visibility data using the `make-image.sh` script.

```
./make-image.sh L2007_03463_SB10_compressed.MS DATA observed
```

This will create `observed.img` (a `casa` image) and `observed.fits` (a `fits` image). You can use `casaviewer` to view `observed.img`, or you can use a `fits` viewer (e.g. `ds9`) to view `observed.fits`.

Simulation

Next, we will simulate visbility data using BBS. The sky model, `sky.in`, contains the two strongest radio sources at LOFAR frequencies: Cassiopeia A and Cygnus A. The sky model is just a flat `ASCII` file, so you could inspect it with a text editor.

BBS expects a *parameter set file* that describes the processing steps that will be performed. A parameter set file is simply a text file that contains a list of `key = value` pairs. The file we will use is called `simulation.parset`. If you like, open it in a text editor and try to figure out what is does. An overview of the keys recognized by BBS can be found in the parset documentation.

Run BBS using the `calibrate` script as follows:

```
calibrate -f --cluster-desc lioffen.clusterdesc --db cepmaster0
--db-user postgres --instrument-db instrument L2007_03463_compressed.vds
simulation.parset sky.in /data/lofarschool/users/<your directory>
```

Make an image of the `MODEL_DATA` column:

```
./make-image.sh L2007_03463_SB10_compressed.MS MODEL_DATA simulation
```

Have a look at either `simualtion.img` or `simulation.fits`. You should see an all sky map, centered on the North celestial pole, containing two strong point sources.

UV-plane calibration

Ok. Time to calibrate the visibility data. Open `uv-plane-cal.parset` in a text editor and figure out what it does. Two separate operations will be performed on the data. First, the (uv-plane) complex antenna gains will be fitted. Second, the data will be corrected for these gains. The calibration should take approximately 5 minutes. Of course, if many people are running at the same time, it may take longer.

Run BBS using the `calibrate` script as follows:

```
calibrate -f --cluster-desc lioffen.clusterdesc --db cepmaster0
--db-user postgres --instrument-db instrument L2007_03463_compressed.vds
uv-plane-cal.parset sky.in /data/lofarschool/users/<your directory>
```

Make an image of the `CORRECTED_DATA` column:

```
./make-image.sh L2007_03463_SB10_compressed.MS CORRECTED_DATA uv-plane
```

Have a look at `uv-plane.img` or `uv-plane.fits`. The image you've just created should look similar to the result of the previous step. Of course, the images will not be exactly the same. There may be calibration errors. For instance, if gain amplitude has gone to zero during the fitting process, then correcting for it will create an image with a very prominent ringing structure across the entire map. Furthermore, the real sky contains many sources, so a sky model that consists of CasA and CygA is only an approximation.

Image-plane calibration

In this exercise we will subtract CasA and CygA to have a look at a number of fainter radio sources. Open `image-plane-cal.parset` in a text editor and figure out what it does. Three operations will be performed on the data. First, the complex antenna gains in the direction of CasA and CygA will be fitted. Second, CasA and CygA will be subtracted from the visibility data (each corrupted with their own complex gains). Third, the visibility data will be corrected for the complex gains in the direction of CasA. The calibration should take approximately 10 minutes. Of course, if many people are running at the same time, it may take longer.

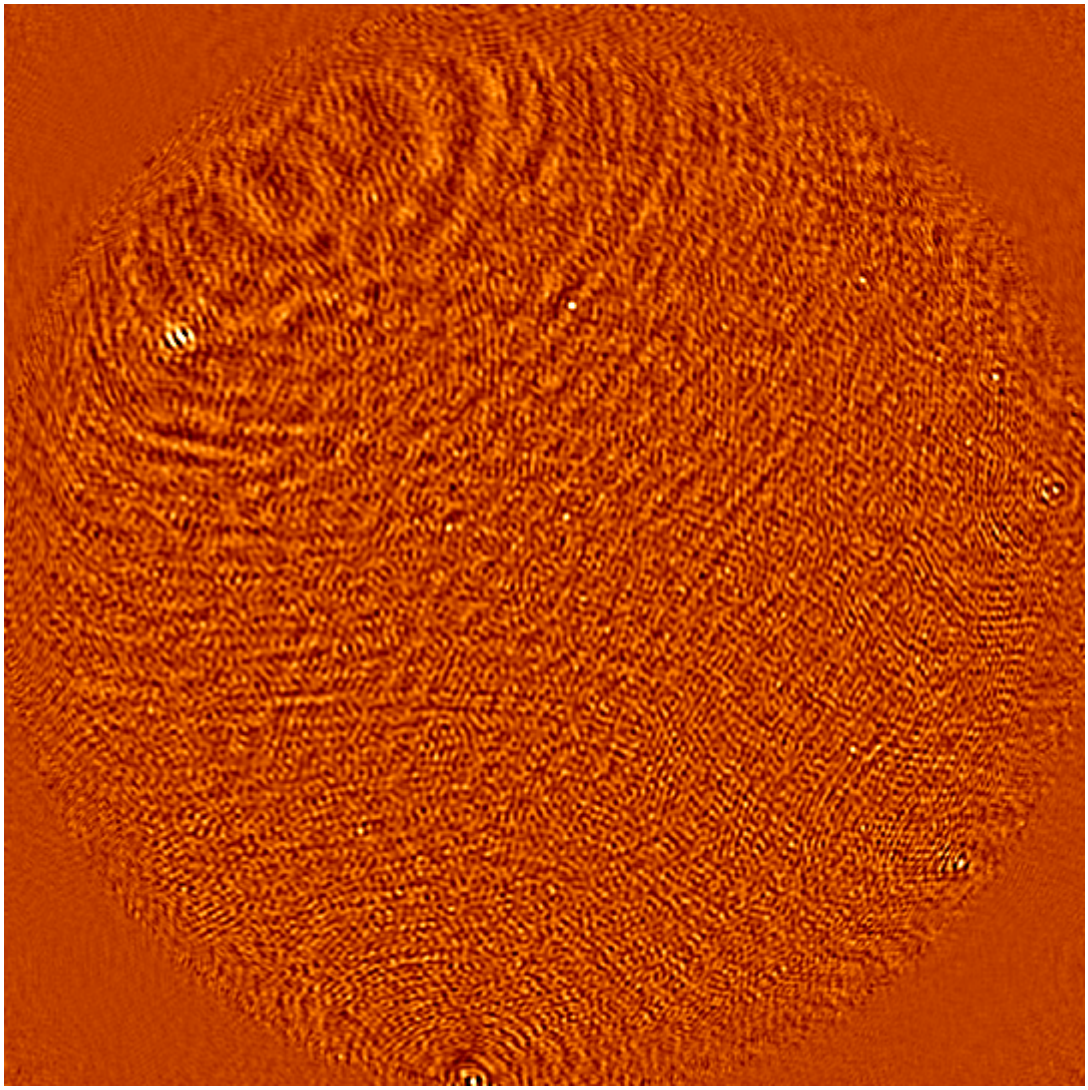
Run BBS using the `calibrate` script as follows:

```
calibrate -f --cluster-desc lioffen.clusterdesc --db cepmaster0
--db-user postgres --instrument-db instrument L2007_03463_compressed.vds
image-plane-cal.parset sky.in /data/lofarschool/users/<your directory>
```

Make an image of the `CORRECTED_DATA` column:

```
./make-image.sh L2007_03463_SB10_compressed.MS CORRECTED_DATA image-plane
```

Have a look at `image-plane.img` or `image-plane.fits`. As an example, for `L2007_03463_SB13_compressed.MS` the calibrated image looks like this (**NB**. Note the remnant of CygA in the upper left corner):



Calibration solutions

Did the image look reasonable? If not, it may that the fitting has failed for some time samples. In this exercise we will inspect the complex gains obtained during the previous step. The fitted model parameters are stored in a parameter database. By default, this database is stored inside the measurement set as a `casa` table called `instrument`.

To plot the gains we will use a simple Python script (`pylab` is used for the plotting). First, run the `ipython` interpreter in its `pylab` aware mode:

```
ipython -pylab
```

The first time you run `ipython` you may be presented with a welcome screen. Just press enter to continue. Now let's plot the complex gains for the x-dipole in the direction of CasA and CygA:

```
import plot
help(plot.plot)    # Use 'q' to exit help

plot.plot("L2007_03463_SB10_compressed.MS/instrument", "CasA")
```

```
plot.plot("L2007_03463_SB10_compressed.MS/instrument", "CygA")
```

You can plot the complex gains for the y-dipole using `parm="Gain:22"` as an extra argument.

```
plot.plot("L2007_03463_SB10_compressed.MS/instrument", "CasA",
parm="Gain:22")
plot.plot("L2007_03463_SB10_compressed.MS/instrument", "CygA",
parm="Gain:22")
```

The help text of the `plot.plot()` function shows a number of options we have not used yet. If you like, try different settings and have a look at the plots.

Solution based flagging

In this exercise we will try to improve the image by flagging visibility data that has become corrupted due to bad fitting solutions for the antenna gains. In particular, amplitudes close to zero have a large impact on the image (because correcting for such gains creates visibilities with very large amplitudes in the residuals).

Before you start, copy your measurement set. This is important because it is not trivial to undo flagging.

```
cp -r L2007_03463_SB10_compressed.MS L2007_03463_SB10_compressed_solflag.MS
```

The script we will use (called `solflag.py`) does two things: It flags visibility data for which the corresponding gain amplitudes fall outside a given range and it flags visibility data based on the (windowed) median of the absolute distance to the median of the gain amplitudes.

The range of valid amplitudes must be supplied by the user. Therefore, we will have another look at the gain plots to pick suitable upper and lower amplitude thresholds. The same thresholds are applied to the gains for both directions (CasA and CygA), so make sure you choose thresholds that are reasonable for both plots.

If you left the ipython interpreter, restart it. Picking thresholds is easiest when the amplitude plots are not stacked (offset).

```
import plot

plot.plot("L2007_03463_SB10_compressed_solflag.MS/instrument", "CasA",
stack=False)
plot.plot("L2007_03463_SB10_compressed_solflag.MS/instrument", "CygA",
stack=False)
```

Pick a suitable upper and lower threshold. Typical values are 0.0008 for the upper and 0.000075 for the lower threshold. Substitute the values you picked in the `solflag.flag()` command below.

```
import solflag
help(solflag.flag)    # Use 'q' to exit help
```

```
solflag.flag("L2007_03463_SB10_compressed_solflag.MS",
    "L2007_03463_SB10_compressed_solflag.MS/instrument",
    half_window=5, threshold=3, cutoffLow=<your lower threshold>,
    cutoffHigh=<your upper threshold>)
```

The flagging script outputs the percentage of timestamps it flagged for each antenna. After the script finishes, make an image of the CORRECTED_DATA column.

```
./make-image.sh L2007_03463_SB10_compressed_solflag.MS CORRECTED_DATA
image-plane-solflag
```

Have a look at `image-plane-solflag.img` or `image-plane-solflag.fits`. Did you see any improvement in the image? You can try to fiddle with the settings of the flagging script to see if you can improve the image further. Remember to start from a clean copy of your measurement set.

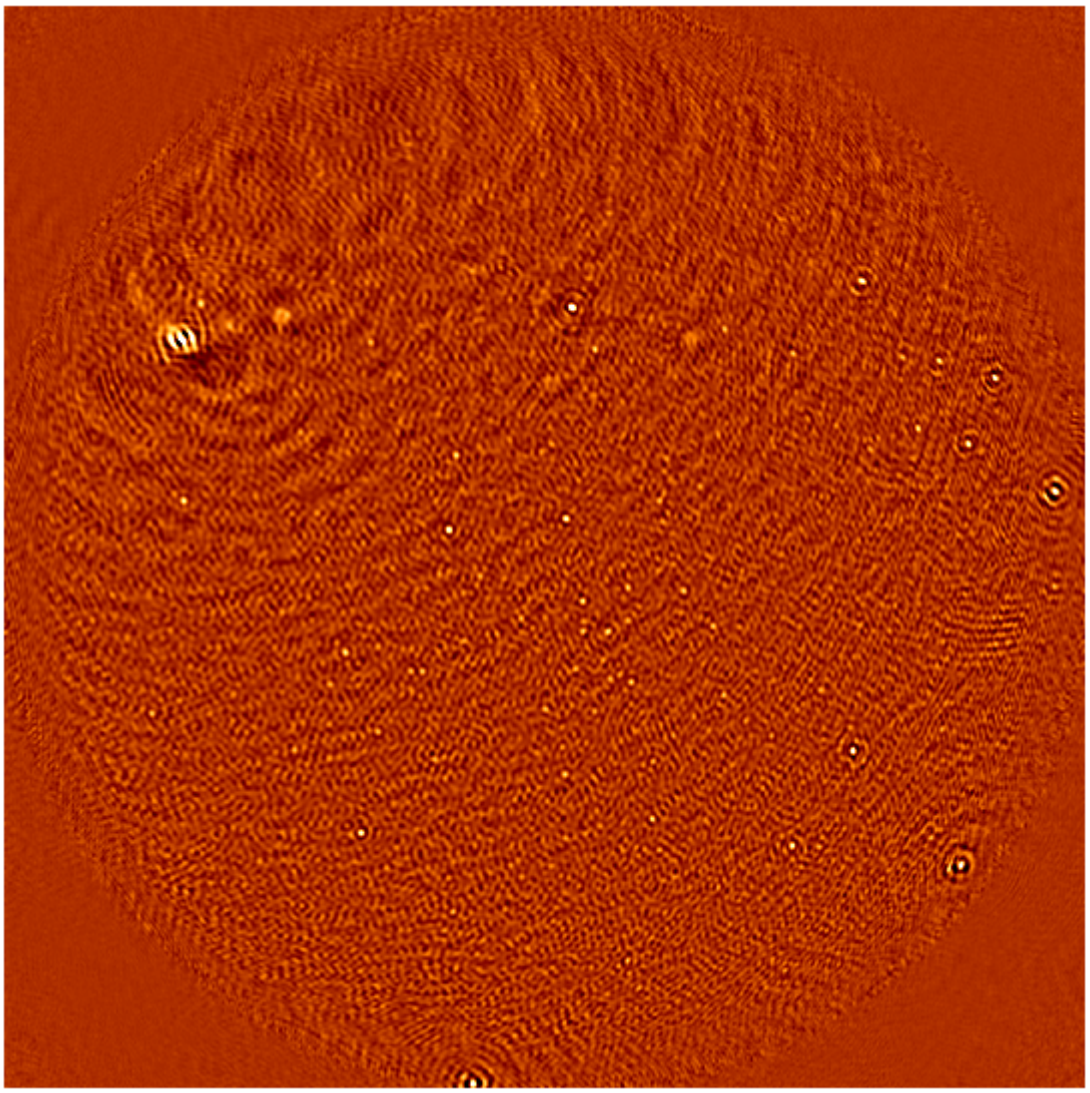
A deeper image

If everything went well, you should now have a quite decent subband image. As a last step, you could try to find other participants who reduced different subbands and average all images to obtain a deeper image.

The `average.py` script can be used as a starting point. Open it in a text editor and set the list of images to the images you have available. Then run it as followes:

```
python average.py
```

Inspect `averaged.img`. If you averaged images of all 10 measurement sets the output will look like (without solution based flagging):



Clean up

If you want to keep (part of) your work, make sure you copy it to an external machine (for instance using `scp`). Once this is done, please remove your directory:

```
rm -rf /data/lofarschool/users/<your directory>
```