# Long Baseline Pipeline for LOFAR HBA Surveys

(genericpipeline implementation)

**C. Coughlan, A. Drabent**

**Aims of the pipeline:** This is a generic pipeline implementation of the LOFAR long baseline pipline. It is optimized for the use with LOFAR HBA Surveys data which already has run through prefactor.

The pipeline can be used in two ways:

1. It can perform standard amplitude calibration using a calibrator source, or
2. it can use the output from the prefactor pipeline - both amplitude, clock and direction independent phase solutions can be applied to the target field.

Regardless of the means of selection, the dataset will be shifted onto each source in turn before averaging, flagging and forming a tied station. It will use the current version of the LBCS calibrator list to identify the sources the dataset will be shifted onto. The data are then converted to circular polarisation and written out as a FITS file.

**What do you need:** If you use data which has been processed with prefactor you need

- to specify the directories transfer_amp_clock_sols_store and phase_sol_input_path in the lofar_lb_gpipeline.parset file,
- the amplitude and clock solutions to be placed in transfer_amp_clock_sols_store, usually called caldata_transfer_amplitude_array.npy and fitted_data_dclock_caldata_transfer_1st.sm.npy, and
- the instrument tables of the direction-independent phase-only self-calibration solutions to be placed in phase_sol_input_path,
  usually called instrument_directionindependent.

The solution tables should be renamed according to the target file names, e.g., if the parmdb of prefactor is stored as L192004_SB009_uv.dppp.MS/instrument_directionindependent, you may call it L192004_SB009.table. The pipeline will compare the subband numbers of the filenames to match the solutions with the target data.

**Setting up the pipeline:** Download the following files from the github repository:

- lofar_lb_gpipeline.parset
- bin-directory

- `plugins`-directory

Before you actually can run the pipeline you need to set up the running environment. These parameters are stored in the pipeline configuration file, called `pipeline.cfg`. It is shipped with your LOFAR installation by default and is usually located in

`$LOFARROOT/share/pipeline/pipeline.cfg`.

You need to alter the following lines:

- `runtime_directory` – Choose a local directory with r+w privileges. The genericpipeline will put mapfiles and the statefile here.
- `working_directory` – Choose a local directory with r+w privileges and sufficient disk space. The genericpipeline will put the processed data there.
- `recipe_directories` – Add the directory in the list, where you store the `plugins` folder from the repository. Avoid spaces here!
- `clusterdesc` – Point it to the appropriate cluster description file.
- `log_file` – Choose a local directory with r+w privileges. The genericpipeline will put the logfiles here.
- `xml_stat_file` – Choose a local directory with r+w privileges. The genericpipeline will put the XML status files here.

At the end of the configuration file you may add:

**[remote]**
**method = local**
**max_per_node = 24**

if you are running the pipeline on a single node machine. Adjust the number of `max_per_node` according to the number of CPU threads your system provides (type `ncproc` in your shell). The cluster description files are usually located in

`$LOFARROOT/share/*.clusterdesc`.

On a single node machine your cluster descripton file should look like that:

```
# Clusterdesc file to do parallel processing on a local machine.

ClusterName = Local
```

```
NNodes = 1
Node0.NodeName = localhost
Node0.NodeFileSys = localhost
Node0.NodeMountPoints = /
```

**Editing the parset:** Before you can run the pipeline, please alter the variables in the header of `lofar_lb_gpipeline.parset`.

- `shift_avg_timestep` – averaging time step after shifting and phase up. Make sure that this is a sensible value!
- `shift_avg_freqstep` – averaging frequency step after shifting and phase up.
- `working_directory` – should be the same as specified in the `pipeline.cfg`.
- `results_directory` – directory where the results of the pipeline will be stored.
- `target_input_path` – full directory of the location of the high-resolution LB-data.
- `target_input_pattern` – a string which matches the filename of your high-resolution LB-data, e.g, L18194*.MS.
- `phaseup_command` – credentials for phasing up in curly brackets {`tied station name`:'`antenna regular expression`'}, see LOFAR wiki or cookbook for details.
- `filter_command` – baselines/stations to filter (antenna regular expression), usually set to the same baselines/stations used for phasing up the tied station.
- `manual_targets` – you should set it to `False` in the beginning, unless you specify your targets manually at some point.
- `number_SBs_per_group` – should be same as for `prefactor` run. This is also the number of SBs per AIPS IF. Make sure that this is a sensible number.

If you are using `prefactor`, you need to specify the following parameters:

- `phase_sol_input_path` – full directory of the direction-independent solution tables (parmdbs) `prefactor` provides you.
- `phase_sol_input_pattern` – a string which matches the filename of the solution tables
- `transfer_amp_clock_sols_store` – full directory of the `.npy` files `prefactor` provides you
- `amp_sols_basename` – this should be the same as in your `prefactor` run. The default value is `caldata_transfer`.

The path to the scripts needs also to be adjusted. The easiest way is to keep all scripts in the `bin`-directory downloaded from the repository. Afterwards, you only need to replace `/mnt/home_cr/coughlan/lofar_software/lofar-lb/surveys_processing/lofar_lb_gpipeline/bin` to the actual `bin`-directory. You should stick to `lin2circ` as a `circ_converter_choice`, since the beam is applied to the target data during the pipeline. Download the `lin2circ` script into your `bin`-directory. The flagging strategy is usually installed in `$lofarroot/share/rfistrategies/HBAdefault` if you have a local LOFAR installation.

Do not modify the list of steps, unless you know what you do. Leave the second steps line commented if your data has been processed with `prefactor` before.

**Running the pipeline:** To run the pipeline you need to type the following command:

```
$ genericpipeline.py <parset_file> -c <config_file>
```

The "verbose mode" (`-v`) is recommended.

**Not yet implemented:**

- using AIPS `FRING` via ParselTongue.

## List of pipeline jobs

**Preparation pipeline:**

- `createmap_target` – generate a mapfile of all the target data.
- `ndppp_prep_target` – runs `NDPPP` on the target to flag bad data.

**Applying solutions from prefactor:**

- `transfer_amp_clock_sols` – transfers the solutions from `prefactor` to the target data.
- `is_amps_gains` – adds amplitude solutions to the international stations through scaling the solutions from the core stations.
- `createmap_ps` – generates a mapfile of the phase solution tables.
- `copy_sols` – copies the phase solution tables to the `working_directory`.
- `createmap_pstwo` – generates a mapfile of the copied phase solution tables.

- `match_files` – finds appropriate target measurement sets to match with phase solutions. If no exact match is found, the nearest measurement set will be used to provide antenna info.
- `is_add_phase` – modifies the phase solution tables in-place to add international stations with unity gain and zero phase.
- `make_group_map` – generates mapfile where the target data are sorted into groups (or AIPS IFs).
- `expand_mapfile` – generates mapfile where the target data are matched with the corresponding phase solution tables.
- `ndppp_apply_cal` – runs `NDPPP` to apply the amplitudes, the clock, and the phase solutions from `prefactor` as well as the beam to the target data. Solutions are written into the `CORRECTED_DATA` column.

**The LBCS pipeline:**

- `main_loop` – initializes the amount of loops, depending on the number of suitable long baseline calibrators in the field.
- `prep_dirs` – downloads `LBCS` data, identifies suitable calibrators according to the `nP` parameter (quality criterion), and determines directions and file names.
- `sortmap_tar` – generates mapfile to sort the target data into groups for concatenation.
- `dppp_phaseup` – runs `NDPPP`, performs shifting to a long baseline calibrator, averaging, flagging, adding the tied station, filtering, and concatenating into groups.
- `make_circ` – converts the concatenated data to circular polarization in-place.
- `maptosingle` – generates mapfile which merges all groups to a single file.
- `fits` – virtually concatenates all groups and converts it to FITS. Data is now ready to be imported into AIPS.