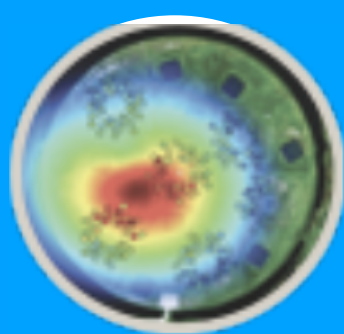


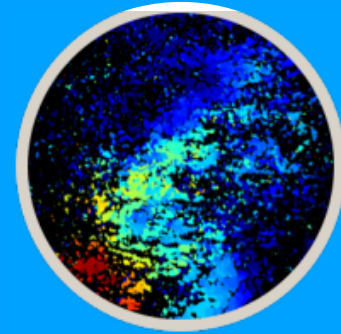
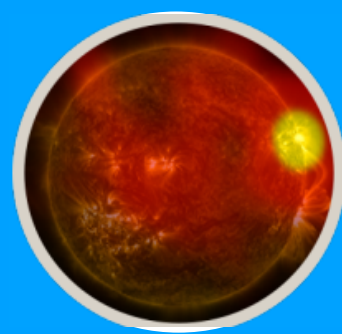
# PulP: hands-on

Vlad Kondratiev  
(ASTRON)



6th LOFAR Data  
School

March 22-26, 2021



# Curriculum:

0

- First things first:  
make sure you have working docker/singularity environment and you have downloaded at least minimally required data to work with.

1

- Following the PulP steps:
  - you will manually run PulP commands to understand them better for one of the beamformed (BF) observation with the setup usually used for pulsar timing observations (CS\_XXYY).
  - compare the results with those from automated PulP
  - optional: play with other PRESTO/dspsr options
  - optional: add extra processing into the mix, e.g.:
    - converting to 8-bit
    - single-pulse analysis
    - RRATs analysis

2

- Pulsar flux calibration

3

- Explore processing steps for other input data for the different observing setups

4

- Easier way to retrieve your «PulP'ed» BF data from the LTA

# Hands-on prerequisites

0

1. Docker / singularity container — [psr-lds2021.\[sif | tar\]](#) (~1.7 GB)
  - Follow the link to download from the instructions that were sent earlier
2. Download the raw input and PulP-processed data for **t4-pulp** following the link in the same instructions
  - **NOTE — the total disk volume is very LARGE ~ 800GB!**
  - However, if you have limited disk space, it is not a show-stopper, not all data are needed at once and for XXYY data (the largest) you can get by by downloading smaller tarballs with pre-processed data (dspsr step).
3. Minimum requirements:
  - very small disk space: CS\_XXYY\_light\_noraw.tgz (~203 MB)
  - modest disk space: CS\_XXYY\_raw\_light.tgz (17.5 GB)
  - large disk space: CS\_XXYY\_raw\_p[0-3].tgz (~350 GB)
  - PulP-processed data: CS\_XXYY\_pulp\_noraw\_8bit.tgz (~800 MB) [Optional]
4. Extra data to explore different observing setups [raw tgz / pulp-processed tgz]:
  - Coherent sum of the stations, Stokes I: CS\_I\_raw.tgz / CS\_I\_pulp.tgz [Optional]
  - Both coherent & incoherent sum of the stations, Stokes I:  
CS\_IS\_I\_raw.tgz / CS\_IS\_I\_pulp.tgz
  - Coherent sum of the stations, Stokes IQUV: CS\_IQUV\_raw.tgz / CS\_IQUV\_pulp.tgz
  - Coherent sum of the stations, 6 rings of TABs: CS\_6rings\_raw.tgz / CS\_6rings\_pulp.tgz
  - Fly's Eye observation, Stokes I: FE\_I\_raw.tgz / FE\_I\_pulp.tgz

# Test the container / software

1. If you are working on CEP3, follow the separate instructions on how to log in to a CEP3 working node and how to start your Singularity or Docker container there.

2. In general:

→ Docker:

- `docker run --net=host -w ${HOME} -e ${USER} -e ${HOME} --rm -it psr:lds2021 [-u <uid>:<gid>]` - you can even run it as root with `-u 0:0` (be careful though!)  
-v <datadir>:<datadir> - make sure you have directory with the data that you've downloaded for the hands-on session
- If you have problems with opening up GUI applications from dockers, try:
  - `xhost +local:root` (outside docker before starting it)
  - Add following options to your docker run command:
    - `-e DISPLAY -e QT_X11_NO_MITSM=1 -v /tmp/.X11-unix:/tmp/.X11-unix:rw`

[simple, but not safe;  
There are other solutions  
as well]

→ Singularity:

- `singularity shell --bind <your/path/to/school/folder>, psr-lds2021.sif`

3. Test the software:

→ Run, e.g.:

- `dspsr -h`
- `pdmp -h`

→ You should get the list of command-line options. If you get some errors instead, something is wrong. Let us know to help you out.

# Explore PulP processing (1)

1

1. In this hands-on session you will try to execute different PulP commands manually, explore the output data, and compare with the results from the automated pipeline. In this part of the tutorial the focus is on the first dataset CS\_XXYY, you can explore other datasets yourself later in the week or after the school.

2. The data:

→ 5-min observations of the pulsar B0809+74 with different observing setups, namely:

- 1 tied-array beam (TAB), complex-voltage data (CV) — CS\_XXYY
- 1 TAB, Stokes I data — CS\_I
- 1 incoherent beam (station beam) + 1 TAB, Stokes I — CS\_IS\_I
- 1 TAB, Stokes IQUV — CS\_IQUV
- 127 TABs (6-ring setup), Stokes I — CS\_6rings
- Fly's Eye observation, Stokes I — FE\_I

→ Both **raw** data and automatically processed by PulP (**pulp**)

→ We assume the data are here:

- Raw: <your/path/to/school/folder>/t4-pulp/raw = <t4-pulp>/raw
- PulP: <your/path/to/school/folder>/t4-pulp/pulp = <t4-pulp>/pulp

# Explore PulP processing (2)

1

3. In this hands-on session we will be processing CS\_XXYY observation L667450

→ Untar the data with:

- `tar xvfz CS_XXYY_raw_p[0-3].tgz` for each tarball - if you have ALL raw data
- **OR:** `tar xvfz CS_XXYY_raw_light.tgz`
- **OR:** `tar xvfz CS_XXYY_light_noraw.tgz`
- also untar the corresponding PulP-processed tarball:
  - `tar xvfz CS_XXYY_pulp_noraw_8bit.tgz`

→ Raw data will be in:

- **(i)** `<t4-pulp>/raw/CS_XXYY` — if you have ALL raw data
- **(ii)** `<t4-pulp>/CS_XXYY_raw_light` — if you have raw data for only 1 frequency part P000
- **(iii)** `<t4-pulp>/CS_XXYY_light_noraw` — if you have only pre-processed data by DSPSR

4. create the working directory:

- `cd <t4-pulp>`
- `mkdir -p <tutorial>` [you can name it whatever you want]
- `cd <tutorial>`

5. copy parfile from «pulp» directory to the current directory:

- `cp <t4-pulp>/pulp/CS_XXYY/L667494/pulp/cs/0809+74.par .`



# Explore Pulp processing (3)

1

6. If you do not have raw data (iii):

→ copy \*.ar files in the current directory:

- `cp ../CS_XXYY_light_noraw/*.ar .`

→ or make the links:

- `ln -s ../CS_XXYY_light_noraw/*.ar .`

→ skip steps 7-8, continue with the step 9.

7. First let's dedisperse/fold the data for a single (1st) frequency part:

→ `dspsr -b 1024 -A -L 5 -E 0809+74.par -O B0809+74_L667450_SAP0_BEAM0_PART0 -U minX1 -t 1`

`<t4-pulp>/data/raw/CS_XXYY/L667450/cs/L667450_SAP000_B000_S3_P000_bf.h5` [as input you give only

ONE .h5 file for a given part. It can be any out of four]

- in case (ii) use `<t4-pulp>/CS_XXYY_light_noraw/L667450_SAP000_B000_S3_P000_bf.h5` as input file

→ Inspect the metadata (header) of the output ar-file with *psredit* command:

- `psredit B0809+74_L667450_SAP0_BEAM0_PART0.ar`

→ Were the data dedispersed?

→ You can plot the profile using e.g. *pav* command:

- `pav -DFTp B0809+74_L667450_SAP0_BEAM0_PART0.ar`
- Or polarimetric profiles: `pav -SFT B0809+74_L667450_SAP0_BEAM0_PART0.ar`
- To plot the spectrum: `pav -GTP B0809+74_L667450_SAP0_BEAM0_PART0.ar`
- To remove dispersion delay between channels you can add «d» option to *pav*:
  - `pav -GTpd B0809+74_L667450_SAP0_BEAM0_PART0.ar`

→ In case (ii) skip the step 8 and continue with the step 9.

# Explore PulP processing (4)

1

8. In case (i) run *dspsr* for all parts (except the 0th which was already done):

- for part in `seq 1 1 19`; do echo "PART=\$part" ; dspsr -b 1024 -A -L 5 -E 0809+74.par -O B0809+74\_L667450\_SAP0\_BEAM0\_PART\${part} -U minX1 -t 1  
<t4-pulp>/data/raw/CS\_XXYY/L667450/cs/L667450\_SAP000\_B000\_S3\_P`printf "%03d" \$part`\_bf.h5 ; done
- Using «-t 1» (number of threads=1) usually a good idea here as to give room for other users on this cluster. In automated pipeline we also use this as Slurm does not expect processes in PulP to use large number of cpus.

9. Add frequency parts together:

- psradd -R -m time B0809+74\_L667450\_SAP0\_BEAM0\_PART\*.ar -o b0809.ar
- psredit b0809.ar
  - Was bandwidth increased?
- pav -DFTpd b0809.ar
  - Do you see the profile?
- pav -GTpd b0809.ar
  - What can you tell about the spectrum?

10. Zapping RFIs:

- clean.py -F surgical -o b0809.paz.ar b0809.ar
- pav -DFTpd b0809.paz.ar
  - How does it look?
- pav -GTpd b0809.paz.ar
  - And now?

11. Extra manual RFI excision:

- pazi b0809.paz.ar [save changes pressing «S» button]
- OR: psrzap b0809.paz.ar
- Move new .pazi or .zap file to b0809.pazi.ar



# Explore PulP processing (5)

## 12. Dedispersion:

- `pam -D -e dd.ar b0809.pazi.ar`
- `psredit b0809.pazi.dd.ar`
  - What is changed?

## 13. Create different diagnostic plots using *pav*

- Make use of the PulP lecture or read *pav* manual: `pav -h`
- Compare diagnostic plots with plots from automated PulP, look for *status.png* file

## 14. Optimize period and DM:

- `pdmp -mc 100 -mb 128 b0809.pazi.dd.ar`
- How much DM and period have changed?
- Correct the archive file updating the DM and period:
  - `pam -D -e pdmp.AR -d <new DM> --period <new topo Period> b0809.pazi.dd.ar`
  - Inspect with *psredit* and *pav*
  - Did S/N improve?
  - **Note, pdmp optimizes P/DM based on higher S/N which is not always the correct! Be mindful!**

## 15. Optional: do single-pulse analysis for this CV observation.

- *digifil* for every part
- *sigproc\_splice*
- *prepdata*
- *single\_pulse\_search\_lotaas.py* OR simply: *single\_pulse\_search.py*

## 16. Extra:

- Run *prepfold* on input \*.fil data (from *digifil* and *sigproc\_splice*).
- Run similar *dspsr* command as before but using .fil file as input.
  - How does the profile look like? Same as before? What is different?

# Pulsar flux calibration (intro)

In general (see e.g. Lorimer & Kramer 2005):

$$\Delta S_{\text{sys}} = \frac{T_{\text{sys}}}{G \sqrt{n_p t_{\text{obs}} \Delta f}} = C \sigma_p,$$

$C = \text{SEFD}$

**LOFAR**

▶  $\Delta f / f \sim 0.5$  (huge)

## Contributing factors

- ▶ Beam shape has strong dependence on AZ, EL, and frequency, and thus the gain,  $G$
- ▶  $\text{Gain}(f) \neq \text{const}$
- ▶  $T_{\text{sys}} = T_{\text{sky}} + T_{\text{inst}}$
- ▶  $T_{\text{sky}}(f) \sim f^{-2.55}$
- ▶  $T_{\text{inst}}(f) \neq \text{const}$
- ▶  $T_{\text{src}}(f) \neq \text{const}$  (ignore for now)
- +
- ▶ Broken tiles (~5%)
- ▶ Coherence scaling  $S/N \sim N^{0.85}$ ,  $N$  — number of 48-tile stations
- ▶ Radio frequency interference (RFI), on average 25-30% (MSP data, 1 ch/sub, normally — much less)

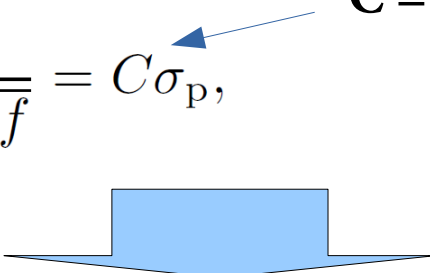
# Pulsar flux calibration (intro)

2

In general (see e.g. Lorimer & Kramer 2005):

$$\Delta S_{\text{sys}} = \frac{T_{\text{sys}}}{G \sqrt{n_p t_{\text{obs}} \Delta f}} = C \sigma_p,$$

$C = \text{SEFD}$



$$\text{SEFD} = \frac{2\beta k [T_{\text{inst}}(f) + T_{\text{sky}}(f, \text{GL}, \text{GB})]}{N_s^\gamma A_{\text{eff}}(f, \text{EL}) [1 - \xi] \sqrt{n_p [1 - \zeta(f)] \left(\frac{T_{\text{obs}}}{\text{nbins}}\right) \Delta f}}$$

$\beta$  — digitization factor = 1

GL, GB — Galactic longitude and latitude

$\gamma$  — coherence factor  $\approx 0.85$

$N_s$  — number of stations used

$n_p$  — number of polarizations (2)

$A_{\text{eff}}$  — effective area of a 48-tile station

$\xi$  — average fraction of bad/flagged dipoles/tiles

$\zeta$  — RFI fraction

nbins — number of bins in the profile

$T_{\text{obs}}$  — observation length (s)

$\Delta f$  — frequency channel width (Hz)

# Pulsar flux calibration (intro)

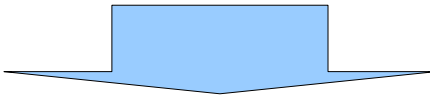
2

In general (see e.g. Lorimer & Kramer 2005):

$$\Delta S_{\text{sys}} = \frac{T_{\text{sys}}}{G \sqrt{n_p t_{\text{obs}} \Delta f}} = C \sigma_p,$$

$C = \text{SEFD}$

New preliminary  
coherence factor  
for Cobalt:  
 $\gamma = 0.816$


$$\text{SEFD} = \frac{2\beta k [T_{\text{inst}}(f) + T_{\text{sky}}(f, \text{GL}, \text{GB})]}{N_s^\gamma A_{\text{eff}}(f, \text{EL}) [1 - \xi] \sqrt{n_p [1 - \zeta(f)] \left(\frac{T_{\text{obs}}}{\text{nbins}}\right) \Delta f}}$$

$\beta$  — digitization factor = 1

GL, GB — Galactic longitude and latitude

$\gamma$  — coherence factor  $\approx 0.85$

$N_s$  — number of stations used

$n_p$  — number of polarizations (2)

$A_{\text{eff}}$  — effective area of a 48-tile station

$\xi$  — average fraction of bad/flagged  
dipoles/tiles

$\zeta$  — RFI fraction

nbins — number of bins in the profile

$T_{\text{obs}}$  — observation length (s)

$\Delta f$  — frequency channel width (Hz)

# Beam models

2

1) **“arts”**, improved Hamaker model, provides full EM simulations of a 24-tile HBA sub-station, including edge effects and grating lobes (Hamaker's model is based on an infinite array of elements).

In practice →

Table of 91 ELs \* 361 AZs \* 29 frequencies

- AZ, 0 — 360 deg, 1-deg step
- EL, 0 — 90 deg, 1-deg step
- Frequency, 110 — 250 MHz, 5-MHz step

Note! When calibrating, for a given EL  $A_{\text{eff}}$  is averaged over all azimuths, as the stations are randomly rotated.

2) **“arisN”**, maximum theoretical value of  $A_{\text{eff}}$  ( $A_{\text{max}}$ ) is scaled as  $\sim \sin(\text{EL})^{1.39}$  as in Noutsos et al. (2015). For HBA,  $A_{\text{max}} = 48 * 16 * \min\{\lambda^2/3, 1.5625\}$ .

3) **“hamaker\_carozzi”**, maximum theoretical value of  $A_{\text{eff}}$  ( $A_{\text{max}}$ ) is corrected by a corresponding factor calculated from the Carozzi's implementation of the Hamaker model. In practice, we use functions from the “mscorpol” package (on Github) written by Tobia Carozzi that calculate Jones matrices for a given HBA station, date/time and frequency (there is also a standalone script `antennaJones.py` to do that). Unlike “arts” model, this model is based on a real station (it uses coordinates, cable delays and time deltas). We used CS001, the difference for other stations is much smaller than the nominal flux error.

$A_{\text{eff}}$  is scaled by  $B(\text{PSR})/B(\text{CasA})$ , where  $B = 0.5 * |J_{xx} \times J_{xx}^* + J_{xy} \times J_{xy}^* + J_{yx} \times J_{yx}^* + J_{yy} \times J_{yy}^*|$ . The value of  $B(\text{PSR})$  is normalized by reference value of the CasA observation  $B(\text{CasA})$  used in Wijnholds & van Cappelen for A/T measurements. Although, for all freqs the value for CasA is almost 1.0 (changing in 2-3 digits after decimal point).

# Pulsar flux calibration (intro)

2

In general (see e.g. Lorimer & Kramer 2005):

$$\Delta S_{\text{sys}} = \frac{T_{\text{sys}}}{G \sqrt{n_p t_{\text{obs}} \Delta f}} = C \sigma_p,$$

**C = SEFD**



$$\text{SEFD} = \frac{2\beta k [T_{\text{inst}}(f) + T_{\text{sky}}(f, \text{GL}, \text{GB})]}{N_s^\gamma A_{\text{eff}}(f, \text{EL}) [1 - \xi] \sqrt{n_p [1 - \zeta(f)] \left(\frac{T_{\text{obs}}}{n_{\text{bins}}}\right) \Delta f}}$$

For all three beam models all ingredients are the same except for the value of  $A_{\text{eff}}$

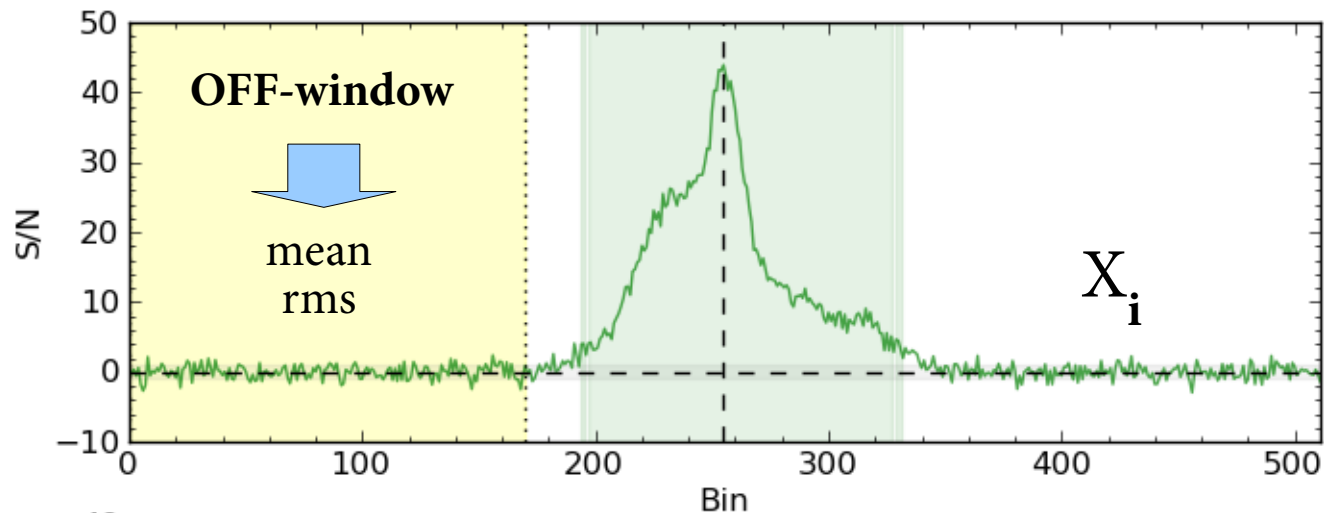


# Other literature/presentations:

- More info about the LOFAR pulsar calibrations in the following papers:
  - «A LOFAR Census of MSPs», Kondratiev et al. 2016, A&A, 585, 128
  - «A LOFAR Census of non-recycled pulsars:...», Bilous et al. 2016, A&A, 591, 134
- and several LOFAR Status Meeting (LSM) presentations:
  - Kondratiev, LOFAR MSP Population. Pulsar flux calibration, Jan 7, 2015
  - Kondratiev, Bilous, LOFAR Pulsar Flux Calibration, Oct 14, 2015
  - Kondratiev, van Amersfoort, New stations' coherency test, May 9, 2018
- all LSM presentations can be found online here:
  - <https://www.astron.nl/LofarSlides/index.php>

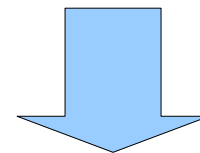
# Pulsar flux calibration (profile, S/N)

2

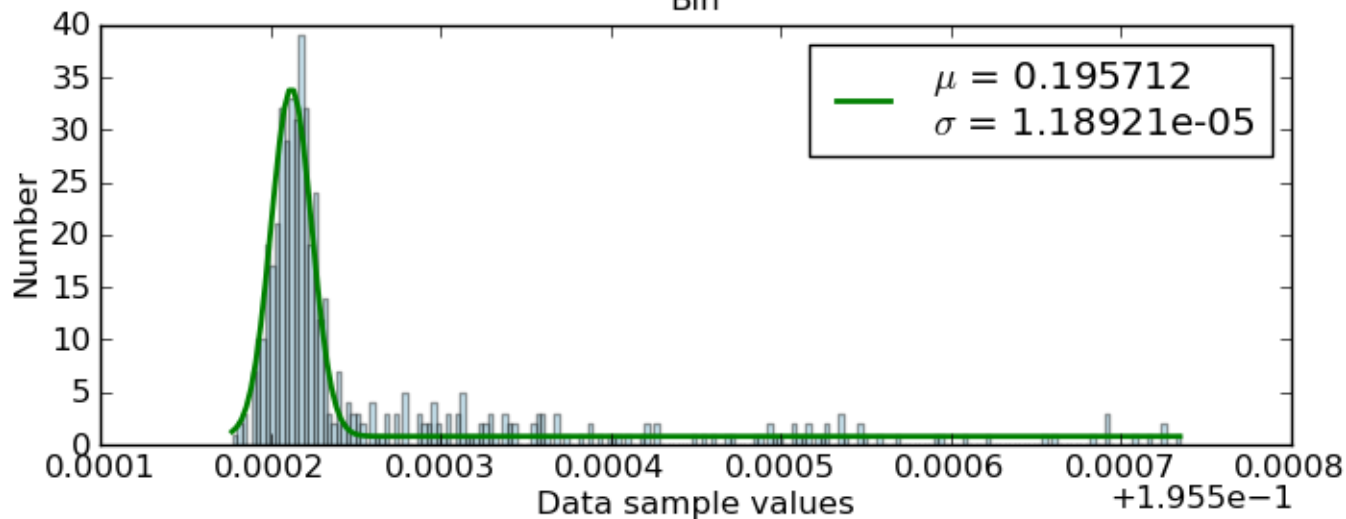


$$(S/N)_i = (X_i - \text{mean}) / \text{rms}$$

i — profile bin



$$\text{Flux}_i = (S/N)_i * \text{SEFD}$$



# Other factors affecting flux measurements

2

- Scattering → hard to get S/N, it is underestimated
- Refractive scintillations.  
Can change pulsar flux by a factor of  $\sim 1.5$ . Need long-term monitoring program  
Diffractive scintillations is not a factor → averaged out,  $\Delta\nu_d < 0.2$  MHz
- Beam jitter by the ionosphere.  
Can be up to  $\sim 2$ -3 arcmins, i.e. half the Full-Core HBA TA beam (at half maximum)
- Variation of Tsys with time due to rise/set of the Galactic plane (up to 30-40% when Galactic plane is in the FoV) and other strong background sources.  
Also with pointing direction due to noise coupling effects.

Despite these factors:

- We've got  $\sim 20\%$  agreement with EOR data for the new LOFAR pulsar J0815+4611
- Flux estimates from the MSSS images (Rene Breton) for several MSPs — on average there is an agreement within  $\sim 40\%$

# Pulsar flux calibration (software)

- `tsky.py` – Tsky (GL, GB, freq) or (RA, DEC, freq)
- `lofar_tinst.py` – T of the instrument (both HBA and LBA)  
    `--plot` – Tinst-vs-Freq diagnostic plot
- `lofar_gain.py` – Aeff (freq, EL) for a 48-tile station (HBA only)  
    `--plot` - diagnostic plots  
    `--model <arts | arisN >`. For hamaker\_carozzi beam model  
    one can use corresponding function(s) after  
    importing it as a module
- `snr.py` – calculate S/N using different methods (Q-Q probability plot, Off-pulse range, Polynomial to the baseline), so one can choose proper method and/or other parameters (fscrunching/bscrunching, off-pulse window) for flux calculation

*use -h option to get help*

# Pulsar flux calibration (software)

- **lofar\_psrflux.py** – to calculate flux density in mJy for a given PSRFITS file (ar-file). First tscrunching all observation (so, good only for not very long ones)
  - plot, --plot-saveonly – diagnostic plots
  - spectrum=#NCHAN – to produce calibrated spectrum for N output channels, and plot
  - spectrum-skip-first-channels=#INCHAN
  - spectrum-skip-last-channels=#INCHAN
  - model <arts | arisN | hamaker\_carozzi>
- **lofar\_fluxcal.py** – to calibrate the samples in mJy in the PSRFITS file (or writes out new file). Calibrates separately individual sub-integrations. Can also calibrate different Stokes separately.
  - model <arts | arisN | hamaker\_carozzi>
  - plot\* and --spectrum\* options are also there

*use -h option to get help*

Both programs can read .h5 file to get number of stations using *--meta* option (preferable Way). Unfortunately, info about the flagged tiles is not yet available for BF data. Currently, this info can be obtained via separate scripts (explained further) and passed to the *lofar\_fluxcal.py* or *lofar\_psrflux.py* via command-line option *--flagged*

# Pulsar flux calibration (software)

2

All scripts are available in the github:

<http://github.com/vkond/LOFAR-BF-pulsar-scripts>

in the **fluxcal** sub-directory



# Lets flux-calibrate some data... (1)

2

1. Lets calibrate the data from the previous Part I of this tutorial
  - Use \*.ar file after the step 12 or 14 of Part I
  - Lets assume the file to calibrate is called b0809.ar
2. Update your docker with the following commands:
  - only needed for provided docker/singularity containers, antenna position files were not kept in the tree, so we need to download them again
  - `cd /opt/lofarsoft`
  - `mkfir -p usg/data/lofar`
  - `cd usg/data/lofar`
  - `svn co http://usg.lofar.org/svn/code/trunk/data/lofar .`
  - `cd <t4-pulp>/tutorial`

# Lets flux-calibrate some data... (3)

2

3. Then we need to run *snr.py* script to determine the method to calculate S/N and other parameters specific for the method. There are 3 methods you can choose from QQ, Off, and Polynom. So far, the best method to use is «Off», where you determine the Off-pulse window to calculate the S/N of the profile. The «QQ» method is based on calculating noise rms on Q-Q plots, and in my experience is not very reliable. The «Polynom» method can be used for very broad profiles. In this case the polynom is fit to the profile, gets subtracted and noise mean/rms are calculated. There are pitfalls in this method. During the calibration the S/N within each channel/sub-integration should be high enough, otherwise noise will be fit, and rms will be underestimated (S/N will be overestimated). There is also «Psrstat» method, but you can not use it for calibration. It is present only in *snr.py* to cross-check against other methods.

- First to make *snr.py* work faster, we will Ftp scrunch our ar-file, as we only need the average profile for *snr.py*:
  - *pam -FTp -e prof b0809.ar*
- Then run *snr.py* command using created .prof file as an input:
  - *snr.py --snrmethod=Off --plot b0809.prof*

# Lets flux-calibrate some data... (4)

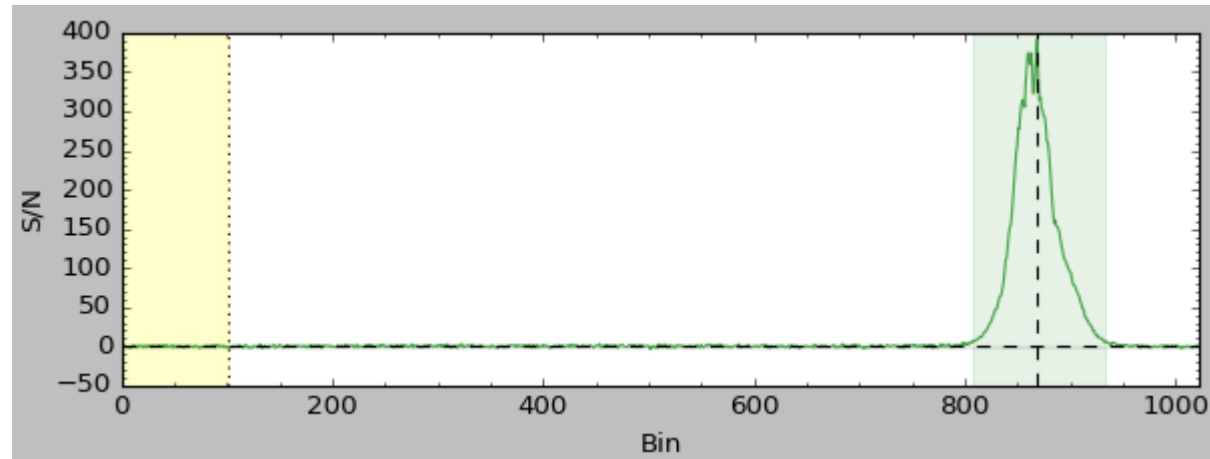
2

4. You get the output and top sub-plot look like this:

b0809.prof

	QQ	Off	Polynom (102)	Psrstat
Mean:	89810.8	89803.3	89803.3	89801.3
RMS:	20.5085	15.3718	238.005	15.7974
Peak S/N:	293.585	392.185	25.3296	381.744
Peak bin:	869	869	869	869
Peak phase:	0.848633	0.848633	0.848633	0.848633
Mean S/N:	12.146	16.6988	1.07851	16.3745
Eff width (bins):	42	44	44	44
Weff/P ratio:	0.0413713	0.0425788	0.0425788	0.042894
S/N:	1910.88	2589.63	167.25	2530.00
Chi^2/dof:	2229.27	3984.39	16.62	1811.21 (-c snr)
#bins with S/N>5.00:	121	128	68	137

5. Yellow shows the current Off-pulse window, the horizontal area is 1-sigma region, and vertical blueish region, are the samples above 3-sigma.
6. Now, you can play with *--off-left* and *--off-right* options, together with extra phase-scrunching (*-b*) and optional profile phase-rotation (*-r*) to select desirable Off-pulse window



# Lets flux-calibrate some data... (5)

2

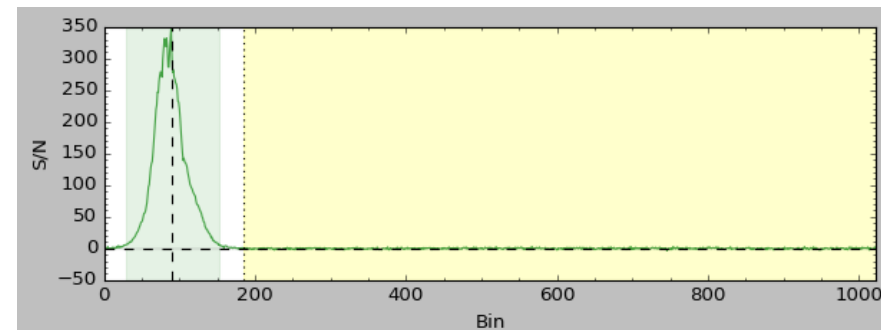
7. However, it is easier to start with `--auto-off` option which in most cases will do the best job for you. It will try to automatically rotate the profile and maximise the Off-pulse window for you. You may only want to bscrunch your profile if it is needed (`-b`). The command is then simply:

→ `snr.py --snrmethod=Off --auto-off --plot b0809.prof`

8. And the output and plot will look like this:

```
b0809.prof
-----
              |      QQ      |      Off      | Polynom (102) | Psrcstat
-----
Mean:          | 89810.8       | 89805.3       | 89805.3       | 89801.3
RMS:           | 20.5085      | 17.3178      | 246.049      | 15.7974
Peak S/N:      | 293.585      | 347.995      | 24.4931      | 381.744
Peak bin:      | 89           | 89           | 89           | 89
Peak phase:    | 0.0869141    | 0.0869141    | 0.0869141    | 0.0869141
Mean S/N:      | 12.146       | 14.7028      | 1.03483      | 16.3745
Eff width (bins): | 42          | 43           | 43           | 44
Weff/P ratio:  | 0.0413713    | 0.0422499    | 0.0422499    | 0.042894
-----
S/N:           | 1910.88      | 2288.95      | 161.10       | 2530.00
Chi^2/dof:     | 2229.27      | 3135.70      | 15.53        | 1811.21 (-c snr)
#bins with S/N>5.00: | 121         | 125          | 66           | 137
-----
AUTO-OFF: --off-left 185 --off-right 1024 -r 0.761719 -b 1
```

9. If you are not satisfied with the Off-pulse selection, you can adjust it by using the option `--auto-off-adjust` to adjust ON-pulse window



# Lets flux-calibrate some data... (5)

2

7. However, it is easier to start with `--auto-off` option which in most cases will do the best job for you. It will try to automatically rotate the profile and maximise the Off-pulse window for you. You may only want to bscrunch your profile if it is needed (`-b`). The command is then simply:

→ `snr.py --snrmethod=Off --auto-off --plot b0809.prof`

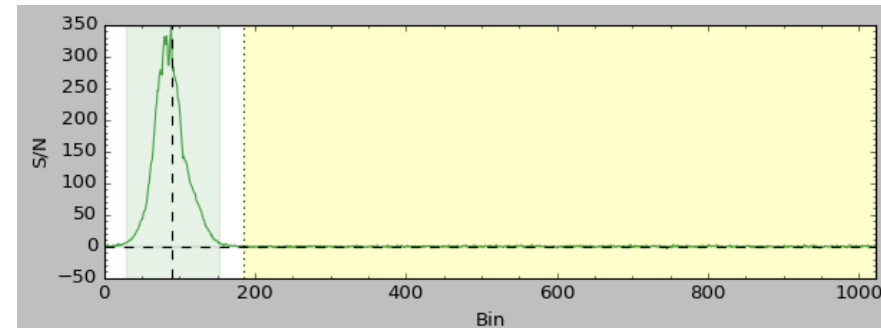
8. And the output and plot will look like this:

```
b0809.prof
-----
              |      QQ      |      Off      | Polynom (102) | Psrstat
-----
Mean:          | 89810.8       | 89805.3       | 89805.3       | 89801.3
RMS:           | 20.5085      | 17.3178      | 246.049      | 15.7974
Peak S/N:      | 293.585      | 347.995      | 24.4931      | 381.744
Peak bin:      | 89           | 89           | 89           | 89
Peak phase:    | 0.0869141    | 0.0869141    | 0.0869141    | 0.0869141
Mean S/N:      | 12.146       | 14.7028      | 1.03483      | 16.3745
Eff width (bins): | 42          | 43           | 43           | 44
Weff/P ratio:  | 0.0413713    | 0.0422499    | 0.0422499    | 0.0422499
-----
S/N:           | 1910.88      | 2288.95      | 161.10       | 2530.00
Chi^2/dof:     | 2229.27      | 3135.70      | 15.53        | 1811.21 (-c snr)
#bins with S/N>5.00: | 121        | 125          | 66           | 137
-----
```

```
AUTO-OFF: --off-left 185 --off-right 1024 -r 0.761719 -b 1
```

These are the parameters determined by the script. You must use them exactly as the input for the `lofar_fluxcal.py` or `lofar_psrflux.py`.

9. If you are not satisfied with the Off-pulse selection, you can adjust it by using the option `--auto-off-adjust` to adjust ON-pulse window



# Lets flux-calibrate some data... (bad tiles)

2

10. For calibration it is important to know what tiles/dipoles were flagged as bad, or at least to know the total fraction of flagged tiles in a given observation. Unfortunately, for BF data this information is not yet available in the HDF5 metadata. Instead, other workarounds must be used to get this info.
11. The Radio Observatory collects this information in the database, and thanks to Wilfred Frijswijk and Sander ter Veen, they provided us with the means to get this information.
12. There is *getState.py* script together with the ascii db about the flagged tiles that can be downloaded now from ASTRON's resource page, and will be part of the LOFAR-BF-pulsar-scripts Github repository as well. Make sure to update the db file «*hardwire\_states\_latest.txt*» file regularly especially for calibration of new observations.
13. To get the flagged tiles/dipoles info for a given timestamp, array config, run:
  - First you need to copy one of the .h5 files from the same directory where all .ar and .AR files are. It can be any one h5-file. This file is needed to provide info about stations used and the time of observation
    - `cp <t4-pulp>/pulp/CS_XXYY/L667494/pulp/cs/rawvoltages/SAP0/BEAM0/L667450_SAP000_B000_S0_P000_bf.h5 .`
  - Run *getState.py*:
    - `cp -r /usr/local/src/LOFAR-BF-pulsar-scripts/tiles/lofar_antenna_state .`
    - `cd lofar_antenna_state/`
    - `./getState.py -i ../L667450_SAP000_B000_S0_P000_bf.h5 -s all -f ../flagged.txt`
    - `cd ..`
14. Get the fraction of the flagged tiles to be used as the input for flux-calibration scripts:
  - `get_flagged_tiles.py -v -f flagged.txt -a HBA L667450_SAP000_B000_S0_P000_bf.h5`



# Lets flux-calibrate some data... (7)

2

15. The output from *get\_flagged\_tiles.py* looks like this:

```
Number of core sub-stations: 48  
Number of core stations: 24  
Number of total flagged tiles: 58  
Fraction of bad tiles: 0.0503472 [5.03472%]  
Worst (sub-)station(s) [#tiles=7, fraction=29.1667%]: CS401HBA1
```

# Lets flux-calibrate some data... (7)

2

15. The output from *get\_flagged\_tiles.py* looks like this:

```
Number of core sub-stations: 48  
Number of core stations: 24  
Number of total flagged tiles: 58  
Fraction of bad tiles: 0.0503472 [5.03472%]  
Worst (sub-)station(s) [#tiles=7, fraction=29.1667%]: CS401HBA1
```

This value must be used in the *--flagged* option for *lofar\_fluxcal.py* or *lofar\_psrflux.py*

16. Now, it is time to run flux calibration script:

- *lofar\_fluxcal.py -t 12 -f 16 --flagged 0.0503472 --meta L667450\_SAP000\_B000\_S0\_P000\_bf.h5 --off-left 185 --off-right 1024 -r 0.761719 -b 1 --snrmeth=Off --model hamaker\_carozzi --plot-saveonly --plot --spectrum 5 -v b0809.ar*
- For the full list of available options use run: *lofar\_fluxcal.py -h*
- *-f 16* – to fscrunch in frequency by a factor of 16, otherwise it will run much longer
- *-t 12* – to scrunch in time by a factor of 12 (1-min average)
- you can get more verbose output with *--vv* option

# Lets flux-calibrate some data... (8)

2

17. The output from *lofar\_fluxcal.py* can look like this:

```
#
# Source: J0814+7429
# RA: 08:14:59.500 DEC: +74:29:05.70 GL(deg): 139.998 GB(deg): 31.6177
# Start MJD: 58373.8576254798 Mid-obs MJD: 58373.859361385
# Tobs(s): 299.964 Cfreq(MHz): 149.512 BW(MHz): 78.125 OrigNchan: 400 ChanWidth(MHz): 0.195312
# Nsub: 5 (tscrunchd: 12) SubintDur(s): 59.9929 Nchan: 25 (fscrunchd: 16) ChanWidth(MHz): 3.125 Nbins:
# Npol: 4 Data state: Stokes
# Nstations: 24 Coherence factor: 0.85
# Bad dipoles/tiles(%): 5.03472
# RFI fraction(%): 28.5825
# S/N method: Off [ --off-left 185 --off-right 1024]
# Beam model: hamaker_carozzi
# File: b0809.ar

Calibrating...
 0 ( 5) Mid-point MJD: 58373.8579726801 AZ(deg): 1.1 EL(deg): 37.3464
 1 ( 5) Mid-point MJD: 58373.8586606414 AZ(deg): 1.18378 EL(deg): 37.3494
 2 ( 5) Mid-point MJD: 58373.8593336469 AZ(deg): 1.26572 EL(deg): 37.3526
 3 ( 5) Mid-point MJD: 58373.8600365638 AZ(deg): 1.35131 EL(deg): 37.356
 4 ( 5) Mid-point MJD: 58373.8607544364 AZ(deg): 1.43869 EL(deg): 37.3598

#
# Output spectrum (Nch=5):
#
# Freq SEFD S/N S/N Prof Chi^2/ Weff DC Speak Sensit. Smean Smean Err
# (MHz) (Jy) mean peak Sign. dof (bins) (%) (mJy) (mJy) (mJy) (mJy)
#-----
118.262 407.459 15.4677 362.889 2397.45 3305.06 43.6466 4.26236 49635.6 141.34 2115.65 4.41687
133.887 303.178 6.58059 161.063 1041.79 638.183 41.8377 4.08572 21248.4 133.636 868.151 4.17612
149.512 306.224 5.17115 122.363 804.952 402.991 43.2747 4.22604 13260.7 109.405 560.402 3.4189
165.137 333.075 3.90974 93.6699 612.386 236.639 42.741 4.17392 12138.5 129.162 506.652 4.03632
180.762 405.246 0.861603 23.5518 144.151 13.3495 37.4609 3.65829 6897.45 285.129 252.328 8.91027
#-----
#
# PSR SEFD S/N S/N Prof Chi^2/ Weff DC Speak Sensit. Smean Smean Err
# (Jy) mean peak Sign. dof (bins) (%) (mJy) (mJy) (mJy) (mJy)
#-----
J0814+7429 354.084 10.9703 258.736 1704.86 1741.51 43.4171 4.23995 20298.3 76.8859 860.636 2.40269
```

# Lets flux-calibrate some data... (8)

2

17. The output from *lofar\_fluxcal.py* can look like this:

```
#
# Source: J0814+7429
# RA: 08:14:59.500 DEC: +74:29:05.70 GL(deg): 139.998 GB(deg): 31.6177
# Start MJD: 58373.8576254798 Mid-obs MJD: 58373.859361385
# Tobs(s): 299.964 Cfreq(MHz): 149.512 BW(MHz): 78.125 OrigNchan: 400 ChanWidth(MHz): 0.195312
# Nsub: 5 (tscrunchd: 12) SubintDur(s): 59.9929 Nchan: 25 (fscrunchd: 16) ChanWidth(MHz): 3.125 Nbins:
# Npol: 4 Data state: Stokes
# Nstations: 24 Coherence factor: 0.85
# Bad dipoles/tiles(%): 5.03472
# RFI fraction(%): 28.5825
# S/N method: Off [ --off-left 185 --off-right 1024]
# Beam model: hamaker_carozzi
# File: b0809.ar
```

Calibrating...

```
0 ( 5) Mid-point MJD: 58373.8579726801 AZ(deg): 1.1 EL(deg): 37.3464
1 ( 5) Mid-point MJD: 58373.8586606414 AZ(deg): 1.18378 EL(deg): 37.3494
2 ( 5) Mid-point MJD: 58373.8593336469 AZ(deg): 1.26572 EL(deg): 37.3526
3 ( 5) Mid-point MJD: 58373.8600365638 AZ(deg): 1.35131 EL(deg): 37.356
4 ( 5) Mid-point MJD: 58373.8607544364 AZ(deg): 1.43869 EL(deg): 37.3598
```

```
#
# Output spectrum (Nch=5):
#
```

# Freq # (MHz)	SEFD (Jy)	S/N mean	S/N peak	Prof Sign.	Chi^2/ dof	Weff (bins)	DC (%)	Speak (mJy)	Sensit. (mJy)	Smean (mJy)	Smean Err (mJy)
118.262	407.459	15.4677	362.889	2397.45	3305.06	43.6466	4.26236	49635.6	141.34	2115.65	4.41687
133.887	303.178	6.58059	161.063	1041.79	638.183	41.8377	4.08572	21248.4	133.636	868.151	4.17612
149.512	306.224	5.17115	122.363	804.952	402.991	43.2747	4.22604	13260.7	109.405	560.402	3.4189
165.137	333.075	3.90974	93.6699	612.386	236.639	42.741	4.17392	12138.5	129.162	506.652	4.03632
180.762	405.246	0.861603	23.5518	144.151	13.3495	37.4609	3.65829	6897.45	285.129	252.328	8.91027

```
#-----
#
# PSR      SEFD      S/N      S/N      Prof      Chi^2/      Weff      DC      Speak      Sensit.      Smean      Smean Err
#          (Jy)      mean      peak      Sign.      dof      (bins)      (%)      (mJy)      (mJy)      (mJy)      (mJy)
#-----
J0814+7429 354.084 10.9703 258.736 1704.86 1741.51 43.4171 4.23995 20298.3 76.8859 860.636 2.40269
```

Note, this is only the nominal error, there are many factors that can affect flux measurements, and we estimate the conservative error on the flux to be within 40-50% (see Bilous et al. 2016)

# Lets flux-calibrate some data... (9)

2

18. The more detailed output is also being saved to *b0809.calib.flux.ascii*

19. The calibrated ar-file (in units of mJy) is also created: *b0809.calib.ar*

20. Also ascii files with flux measurements across the band are created if *--spectrum* option was used. In our case file *b0809.calib.spectrum.5.txt* was created, where «5» means that band was split in 5 frequency parts. The file reads as:

#	Freq	SEFD	S/N	S/N	Prof	Chi^2/	Weff	DC	Speak	Sensit.	Smean	Smean Err
#	(MHz)	(Jy)	mean	peak	Sign.	dof	(bins)	(%)	(mJy)	(mJy)	(mJy)	(mJy)
#												
118.262	407.459	15.4677	362.889	2397.45	3305.06	43.6466	4.26236	49635.6	141.34	2115.65	4.41687	
133.887	303.178	6.58059	161.063	1041.79	638.183	41.8377	4.08572	21248.4	133.636	868.151	4.17612	
149.512	306.224	5.17115	122.363	804.952	402.991	43.2747	4.22604	13260.7	109.405	560.402	3.4189	
165.137	333.075	3.90974	93.6699	612.386	236.639	42.741	4.17392	12138.5	129.162	506.652	4.03632	
180.762	405.246	0.861603	23.5518	144.151	13.3495	37.4609	3.65829	6897.45	285.129	252.328	8.91027	

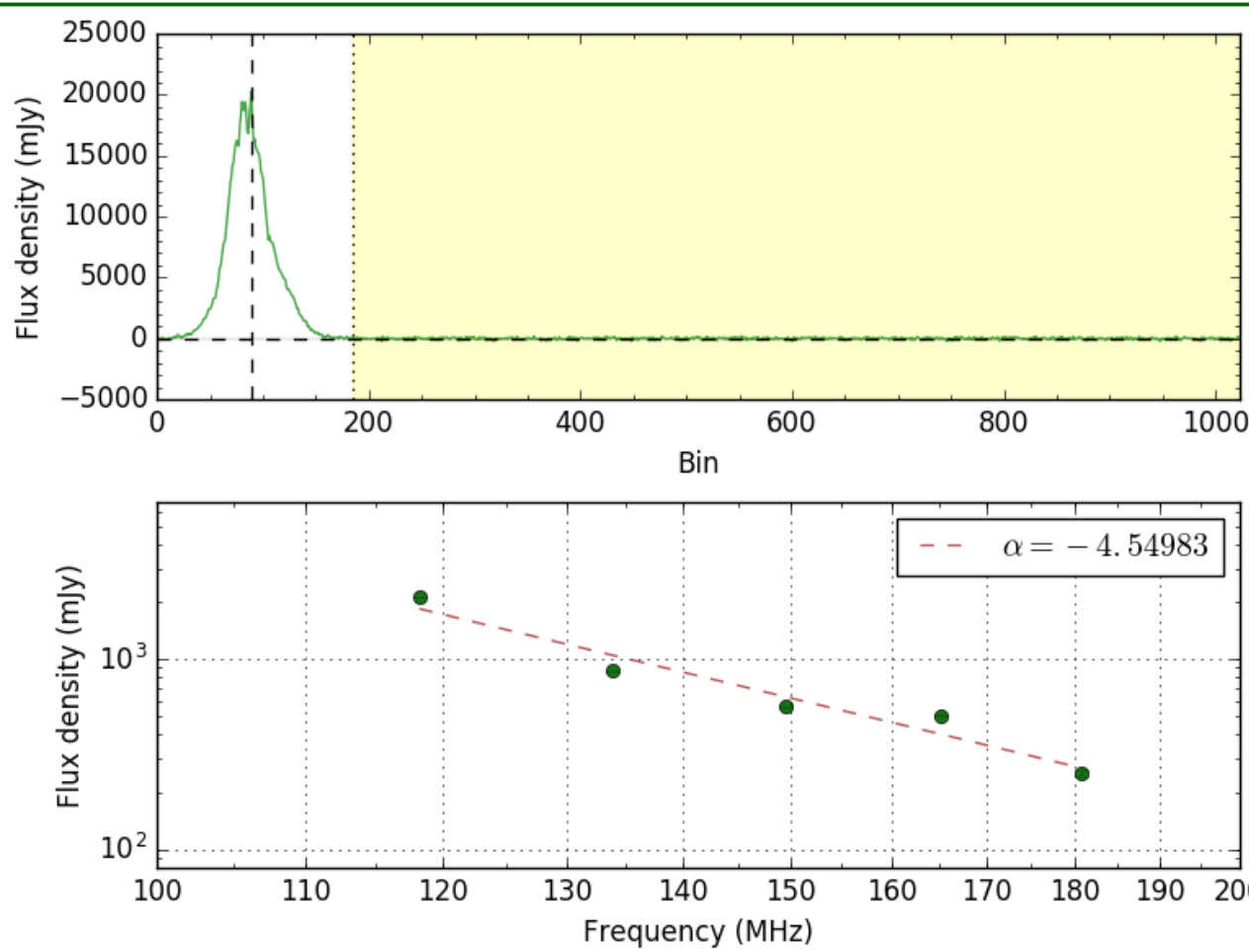
→ In the *--spectrum* option you can specify several splits, like «3,5» and then two files will be created where band is split in 3 and 5 parts. You can also use options *--spectrum-skip-first-channels* and/or *--spectrum-skip-last-channels* to skip number of channels in the original file (possibly scrunched if *-f* was used).

21. For calibration I suggest to use the original (unscrunched) file after the pipeline (could be RFI zapped). In this case the weights are not changed. If you give scrunched files that were scrunched after the RFI zapping, then the RFI fraction can no longer be correctly retrieved. In such case you can also consider to use *--max-weight* option.

# Lets flux-calibrate some data... (10)

2

22. There is also the diagnostic plot created (if options `--plot` and `--plot-saveonly` were used). In our case the image file is `b0809.calib.flux.png` and it looks like this:



The spectrum with fit is also shown if option `--spectrum` was used. Otherwise, file will only have the upper sub-plot

**NOTE**, do not trust the in-band spectral indices! They are very imprecise.



# Lets flux-calibrate some data... (11)

2

23. You can get these warning messages from *casacore* when running flux calibration:

```
2018-09-18 10:29:24 SEVERE MeasTable::dUTC(Double) (file /usr/local/src/casacore/src/measures/Measures/MeasTable.cc, line 4396) Leap second table TAI_UTC seems out-of-date.
2018-09-18 10:29:24 SEVERE MeasTable::dUTC(Double) (file /usr/local/src/casacore/src/measures/Measures/MeasTable.cc, line 4396)+ Until the table is updated (see the CASA documentation or your system admin
),
2018-09-18 10:29:24 SEVERE MeasTable::dUTC(Double) (file /usr/local/src/casacore/src/measures/Measures/MeasTable.cc, line 4396)+ times and coordinates derived from UTC could be wrong by 1s or more.#
```

24. Of course the corresponding files should be updated, but in practice the corresponding values for AZ, EL, effective area will be pretty much the same if the actual time was off by 1 s, so you can ignore it, but make sure to update your Casacore installation later.

25. To make use of hamaker\_carozzi model, you should have *mscorpol* package installed (written by Tobia Carozzi). You can find it here:

- <https://github.com/2baOrNot2ba/mscorpol>
- you also need python-casacore to be installed

or more recent DreamBeam package:  
<https://github.com/2baOrNot2ba/dreambeam>  
Note: *lofar-fluxcal.py* have not been tested with DreamBeam yet

26. In theory, same scripts can be used to calibrate LBA data, but in practice only HBA calibration was characterised and conservative errors were derived. At the moment we simply do not know how good this calibration is for LBA data.

27. The calibration was presented here for HBA Core data. For the Core the reference station is set to be CS002 by default. This can be changed with the *--station* option. In this case the coordinates for another station should also be given with *--latitude* and *--longitude* options. When only one station was used in a observation, this station will be the reference. The coordinates still need to be given with *--latitude* and *--longitude*. This could be the case for a FE observation or observation with an International station.

# Explore PulP processing for other observing setups

1. Manually process the raw data for other observing setups (Stokes I, IQUV data, 6-ring setup, Fly's Eye observation), following the workflow presented in T4, and compare with the results from the automated PulP

→ Tip: you can also check the log-files in the corresponding directories for PulP-processed data in <t4-pulp>/pulp/, such as *\*\_summary\*.log*, *\*\_sap000\_beam0000\*.log* to see what commands were run, and options used.

# LTA retrieve of «PulP'ed» data (1)

- use LTA web-interface (see D1 by Manu Orru)
- or scripts to download specifically pulsar BF data processed by PulP
  - you need to know the exact ObsID(s) to download the data
  - or, if you want all data from the project, then you need to know the project code

Why scripts vs. web-interface?

- Pros:**
- no need to surf through many pages to select files you need on the web
  - you can stage and download many files from different ObsIDs and projects in one go
  - you can specifically choose to download only summary data from all given ObsIDs excluding processed data in \*\_red directories
  - the download will start as soon as there is at least one tarball already staged without waiting for all files to be staged. This makes it faster
  - the downloaded data will be automatically extracted

- Cons:**
- You need to know the exact ObsIDs or Project code to download the data
  - Sophisticated search or filtering is not possible

What scripts?

- *lta-query.py*
- *lta-retrieve.py*

Where?

<https://github.com/vkond/LOFAR-BF-pulsar-scripts>  
in the LTA sub-directory

# LTA retrieve of «PulP'ed» data (2)

1. retrieval is done via *wget* command, so you must setup wget configuration file *~/.wgetrc*
  - Create file if it does not exist yet
  - Add new line with username info: `user=<your LTA username>`
  - Add new line with the password: `password=<your LTA password>`
  - **NB. This *~/.wgetrc* is unencrypted**, so at least close it from reading by others, with `chmod 600 ~/.wgetrc`
2. Also check that you have file *~/.awe/Environment.cfg* with correct LTA username and password (fields *database\_user* and *database\_password*)
  - If you don't have such file, also check that *lta-retrieve.py -h* command provides you with the options listing. Otherwise, you may need to install the script from the Github. Together with scripts there is also an example of this configuration file, where you need to change the username and the password.
  - Change the current value in *database\_user* after the «:» with your LTA username
  - Do the same for your password in the field *database\_password*
  - Protect your file from reading by others (this file is also not encrypted):  
`chmod 600 ~/.awe/Environment.cfg`
3. Obtain csv-file with all data stored in the LTA for the given project
  - `lta-query.py -p <PROJECT_CODE>`

# LTA retrieve of «PulP'ed» data (3)

4. For example, for the Pulsar HBA Census project LC1\_003:

→ *lta-query.py -p LC1\_003*

→ You then get file csv-file *lc1\_003.csv* in the working directory

5. The contents of this csv-file is as follows:

FILENAME,FILESIZE,CREATION\_DATE,URI,OBSERVATIONID

"LOFAR\_PULSAR\_ARCHIVE\_locus001\_L202462\_red\_23cd49f3.tar",24303851520,2014-06-03  
06:48:45,"srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lofar/ops/projects/lc1\_003/202462/  
LOFAR\_PULSAR\_ARCHIVE\_locus001\_L202462\_red\_23cd49f3.tar", "202462"

"LOFAR\_PULSAR\_ARCHIVE\_locus001\_L202467\_red\_f60201c2.tar",25033717760,2014-06-03  
05:30:54,"srm://srm.grid.sara.nl:8443/pnfs/grid.sara.nl/data/lofar/ops/projects/lc1\_003/202467/  
LOFAR\_PULSAR\_ARCHIVE\_locus001\_L202467\_red\_f60201c2.tar", "202467"

....

6. The last column ObsID is especially useful for the observations since ~Cycle 3, when pipeline ID and ObsID become different, so this can help to get the link between observation and its PulP data products

# LTA retrieve of «PulP'ed» data (4)

7. To download all tarballs for a given ObsId (or PipeID):

→ *lta-retrieve.py -u <your LTA username> --csvfile=lc1\_003.csv L202460*

8. You can see all available options with:

→ *lta-retrieve.py -h*

Usage: *lta-retrieve.py* <ObsID.txt1> <ObsID.txt2>...

## Options:

-h, --help            show this help message and exit  
 --sap=SAP#            retrieve data only for the given SAP  
 --tab=TAB#            retrieve data only for the given TAB  
 --part=PART#          retrieve data only for the given PART  
 --summary-only        retrieve only summary directories (CSplots, or CVplots, or redIS). This option has priority over options --sap, --tab, --part, and --skip-summary  
 --skip-summary        Do not retrieve summaries  
 --stage-only          Only stage the data without downloading  
 --skip-staging        Skip staging and start downloading right away  
 --obsids              input arguments are ObsIDs instead of ascii files. Based on given ObsIDs corresponding files will be looked at designated area on CEP2  
 -f FORMAT, --format=FORMAT  
                       column format of input ascii files. By default (websummary), it is the same as from web-summary pages. Other format is 'manual', it's csv format from manual LTA query (expert mode)

--csvfile=CSV-FILE    specify single csv-file (comma-separated-values) with srm-links for all given ObsIDs. With this option, it is assumed that you give the list of ObsIDs instead of ascii files, therefore this option automatically sets --obsids and --format='manual'. Only lines for given ObsIDs will be used from this csv-file  
 --query                as --csvfile but runs SQL query instead of using given csv file. One must specify project as well with --project option. If both --csvfile and --query are given, then --csvfile option has the priority  
 -p PROJECT, --project=PROJECT  
                       specify the project to query. Only to be used with --query option  
 -u USERNAME, --username=USERNAME  
                       specify the LTA username. By default, it's the same as your current login name  
 -l, --log              optional parameter to turn on wget output  
 -q, --quiet            turn off logging from the communication with the LTA database

# LTA retrieve of «PulP'ed» data (5)

9. Using *--format* and *--obsids* options is obsolete, as nowadays you should just use *--csvfile* option.
10. You should always provide *-u* option
11. Using *--query* with *-p* is the same as with *lta-query.py*. However, if you plan on running multiple *lta-retrieve.py* commands, it is more efficient to run *lta-query.py* first, and then re-use of the csv-file.
12. You can only stage the data with *--stage-only*, or skip the staging and proceed with the download (when you know the data were already staged) with *--skip-staging*.
13. You can download (or stage) only the summaries with *--summary-only*, or skip the summaries with *--skip-summary*.
14. You can also specify to download the data (not summaries) for a given SAP, TAB, or frequency part, with options *—sap*, *--tab*, *--part*. Note, however, that these options make only sense to use from ~Cycle 3, when data were ingested via central system. Before that data were ingested to the LTA manually where filenames do not have info about SAP, TAB, and PART.

# LTA retrieve of «PulP'ed» data (6)

15. Now, try to query the project and retrieve the data from your favorite ObsID....
16. Note, staging can take a while (~few days), and depending on the size of the tarballs, your network, the download can take some time as well. I have staged the data from LC1\_003 before, so, you can try to download some of the observations from this project.
17. Beware, the disk size is not unlimited, and you are working with several other groups on the same node. So, do not try to download many observations. If you changed your mind, remove the data you have downloaded previously.
18. First, run *lta-query.py* to get the csv-file with all ingested data for this project
19. Looking in csv-file pick one ObsID/PipeID to download
20. Tip to download all the data for this project. **(Be mindful of the data volume!)**  
The *lta-retrieve.py* requires the list of space-separated ObsIDs to download. To get this list, you can try this bash command:

```
cat lc1_003.csv | grep -v FILENAME | cut -d , -f 5 | cut -d \" -f 2 | sort -n -k 1 | uniq | awk '{printf "L%s\n", $1}' - | paste -s -d ''
```