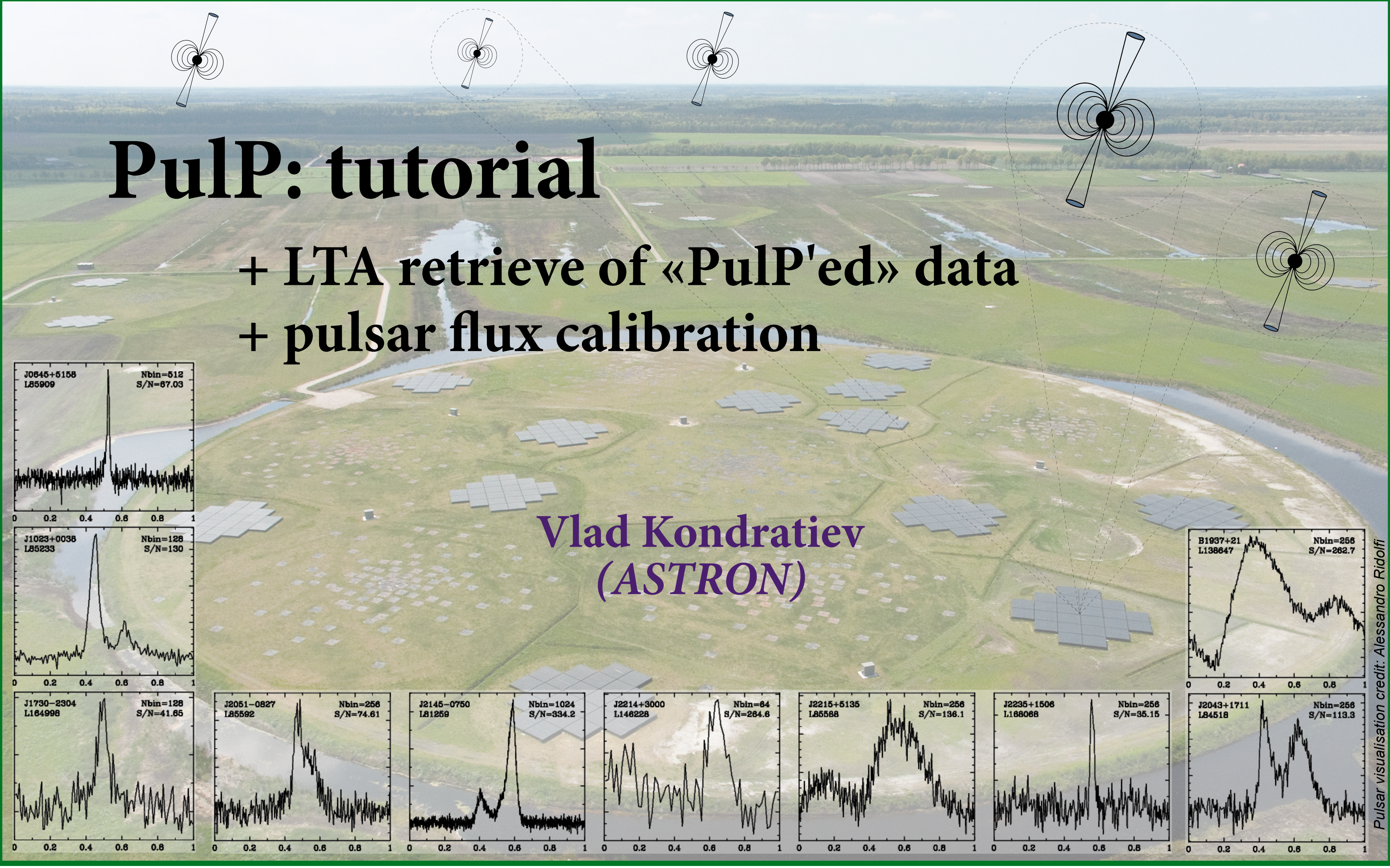
PulP: tutorial

+ LTA retrieve of «PulP'ed» data
+ pulsar flux calibration

Vlad Kondratiev
(ASTRON)


J0645+5158 L85909 Nbin=512 S/N=67.03
J1023+0038 L85233 Nbin=128 S/N=130
J1730−2304 L164998 Nbin=128 S/N=41.65
J2051−0827 L85592 Nbin=256 S/N=74.61
J2145−0750 L81259 Nbin=1024 S/N=334.2
J2214+3000 L146228 Nbin=64 S/N=264.6
J2215+5135 L85588 Nbin=256 S/N=136.1
J2235+1506 L168068 Nbin=256 S/N=35.15
J2043+1711 L84518 Nbin=256 S/N=113.3
B1937+21 L138647 Nbin=256 S/N=262.7
0 0.2 0.4 0.6 0.8 1
Pulsar visualisation credit: Alessandro Ridolfi

5th LOFAR Data Processing School — September 20, 2018

Curriculum:

- First things first:
  we need to set up the working environment on CEP3

- Following the PulP steps:
  → we will manually run PulP commands to understand them better
  → 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

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

- Pulsar flux calibration

Setting working environment on CEP3 cluster

1. Login to CEP3 head node:
   → ssh -XY lhd001 (or lhd002)

2. Open your CEP3 reservation:
   → srun -A lofar_school2018 --reservation=lofar_school2018_114 -N 1 -w lof015 -u bash -i

3. Open new terminal and login again to CEP3 head node

4. In the same terminal login now to the node lof015:
   → ssh -XY lof015

5. module load docker

6. docker images

7. docker run --rm -it -e DISPLAY -v $PWD/.Xauthority:/home/lofarsys/.Xauthority -w $HOME -v $HOME:$HOME -v /home/kondratiev:/home/kondratiev -v /data:/data nexus.cep4.online.lofar:18080/lofar-pulp:latest /bin/bash

8. ...

Explore PulP processing (1)

1. During the software rollout last week we have carried out short 5-min test observations of PSR B0809+74 with different observing setups and ran the PulP. In this tutorial you will try to execute different PulP commands manually, explore the output data, and compare with the results from the automated pipeline. There is no time during the tutorial to do this for all observations, so we will do it just for one observation with single TAB in CV mode. You may try to do it for others later, or pick another during this tutorial and follow similar commands.

2. All raw data are in /data/scratch/LOFARSCHOOL2018_T9_PULP/data/raw

3. The PulP-processed data are in /data/scratch/LOFARSCHOOL2018_T9_PULP/data/pulp

4. In this example we will be processing CV observation L667450 in …/raw/CS_XXYY/L667450

5. Create the working directory:
   → cd /data/scratch/LOFARSCHOOL2018_T9_PULP/home/
   → mkdir -p <your username> [if you have not had it yet]
   → cd <your username>
   → mkdir pulp_XXYY [you can name it whatever you want]
   - NOTE: if this command fails with «Permission denied», try run this command with «sudo», i.e.
     - sudo mkdir pulp_XXYY
     - and change the permissions to write into this directory for everybody:
       - chmod 777 pulp_XXYY OR: chmod go+rw pulp_XXYY

   → cd pulp_XXYY

Explore PulP processing (2)

6. First lets dedisperse/fold the data for a single (1st) frequency part
   - → copy parfile from «pulp» directory to the current directory:
     - cp /data/scratch/LOFARSCHOOL2018_T9_PULP/data/pulp/CS_XXYY/L667494/pulp/cs/0809+74.par .
   - → dspsr -b 1024 -A -L 5 -E 0809+74.par -O B0809+74_L667450_SAP0_BEAM0_PART0 -U minX1 -t 1 /data/scratch/LOFARSCHOOL2018_T9_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]
   - → 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

7. 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 /data/scratch/LOFARSCHOOL2018_T9_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.

Explore PulP processing (3)

8. 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?

9. 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?

10. 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

11. Dedispersion:
    - → pam -D -e dd.ar b0809.pazi.ar
    - → psredit b0809.pazi.dd.ar
      - What is changed?

Explore PulP processing (4)

12. Create different diagnostic plots using *pav*
    - → Make use of the PulP lecture or read pav manual: *pav -h*
    - → Compare diagnostic plots with plots from autmated PulP, look for *status.png* file

13. 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!

14. You are free to try out processing other observations with different observing setups. Make use of PulP lecture to see what commands to run. **Hint!** You can also check the log-files in the corresponding directories for PulP-processed data in /data/scratch/LOFARSCHOOL2018 _T9_PULP/data/pulp, such as **_summary*.log*, **_sap000_beam0000*.log*.

15. Optional: do single-pulse analysis for these 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?

LTA retrieve of «PulP'ed» data (1)

- use LTA web-interface (see L5 by Tom Franzen)
- 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 vlad --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. Create your working directory in */data/scratch/LOFARSCHOOL2018_T9_PULP/home/*, e.g. *.../home/<username>*

19. First, run *lta-query.py* to get the csv-file with all ingested data for this project

20. Looking in csv-file pick one ObsID/PipeID to download

21. Tip to download all the data for this project. NB. - do not do it on the CEP3 node during the school, otherwise we will run out of disk space! 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 ' '
```

Pulsar flux calibration (intro)

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

C = SEFD

$$\Delta S_{\rm sys} = \frac{T_{\rm sys}}{G\sqrt{n_{\rm p} t_{\rm obs} \Delta f}} = C\sigma_{\rm p},$$

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
- Gain(f) ≠ const
- Tsys = Tsky + Tinst
- Tsky(f) ~ f $^{-2.55}$
- Tinst(f) ≠ const
- Tsrc(f) ≠ const (ignore for now)

  +
- Broken tiles (~5%)
- Coherence scaling S/N ~ 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)

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

C = SEFD

$$\Delta S_{\rm sys} = \frac{T_{\rm sys}}{G\sqrt{n_{\rm p} t_{\rm obs} \Delta f}} = C\sigma_{\rm p},$$

$$\mathrm{SEFD} = \frac{2\beta k[T_{\rm inst}(f) + T_{\rm sky}(f, \mathrm{GL}, \mathrm{GB})]}{N_{\rm s}^{\gamma} A_{\rm eff}(f, \mathrm{EL})[1-\xi]\sqrt{n_{\rm p}[1-\zeta(f)](\frac{\mathrm{T_{obs}}}{\mathrm{nbins}})\Delta f}}$$

β — digitization factor = 1
GL, GB — Galactic longitude and latitude
γ — coherence factor ≈ 0.85
N_s — number of stations used
n_p — number of polarizations (2)
A_{eff} — effective area of a 48-tile station

ξ — average fraction of bad/flagged dipoles/tiles
ζ — RFI fraction
nbins — number of bins in the profile
T_{obs} — observation length (s)
Δf — frequency channel width (Hz)

Pulsar flux calibration (intro)

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

C = SEFD

$$\Delta S_{\rm sys} = \frac{T_{\rm sys}}{G\sqrt{n_{\rm p} t_{\rm obs} \Delta f}} = C\sigma_{\rm p},$$

New preliminary coherence factor for Cobalt:
γ = 0.816

$$\mathrm{SEFD} = \frac{2\beta k[T_{\rm inst}(f) + T_{\rm sky}(f, \mathrm{GL}, \mathrm{GB})]}{N_{\rm s}^{\gamma} A_{\rm eff}(f, \mathrm{EL})[1-\xi]\sqrt{n_{\rm p}[1-\zeta(f)](\frac{\mathrm{T_{obs}}}{\mathrm{nbins}})\Delta f}}$$

β — digitization factor = 1
GL, GB — Galactic longitude and latitude
γ — coherence factor ≈ 0.85
N_s — number of stations used
n_p — number of polarizations (2)
A_{eff} — effective area of a 48-tile station

ξ — average fraction of bad/flagged dipoles/tiles
ζ — RFI fraction
nbins — number of bins in the profile
T_{obs} — observation length (s)
Δf — frequency channel width (Hz)

Beam models

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 Aeff is averaged over all azimuths, as the stations are randomly rotated.

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

3) **“hamaker_carozzi”**, maximim theoretical value of A_{eff} (A_{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.

Aeff is scaled by B(PSR)/B(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(PSR) is normalized by reference value of the CasA observation B(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)

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

C = SEFD

$$\Delta S_{\rm sys} = \frac{T_{\rm sys}}{G\sqrt{n_{\rm p} t_{\rm obs} \Delta f}} = C\sigma_{\rm p},$$

$$\Downarrow$$

$$\mathrm{SEFD} = \frac{2\beta k[T_{\rm inst}(f) + T_{\rm sky}(f, \mathrm{GL}, \mathrm{GB})]}{N_{\rm s}^{\gamma} A_{\rm eff}(f, \mathrm{EL})[1-\xi]\sqrt{n_{\rm p}[1-\zeta(f)](\frac{\mathrm{T_{obs}}}{\mathrm{nbins}})\Delta f}}$$

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

Other literature/presentations:

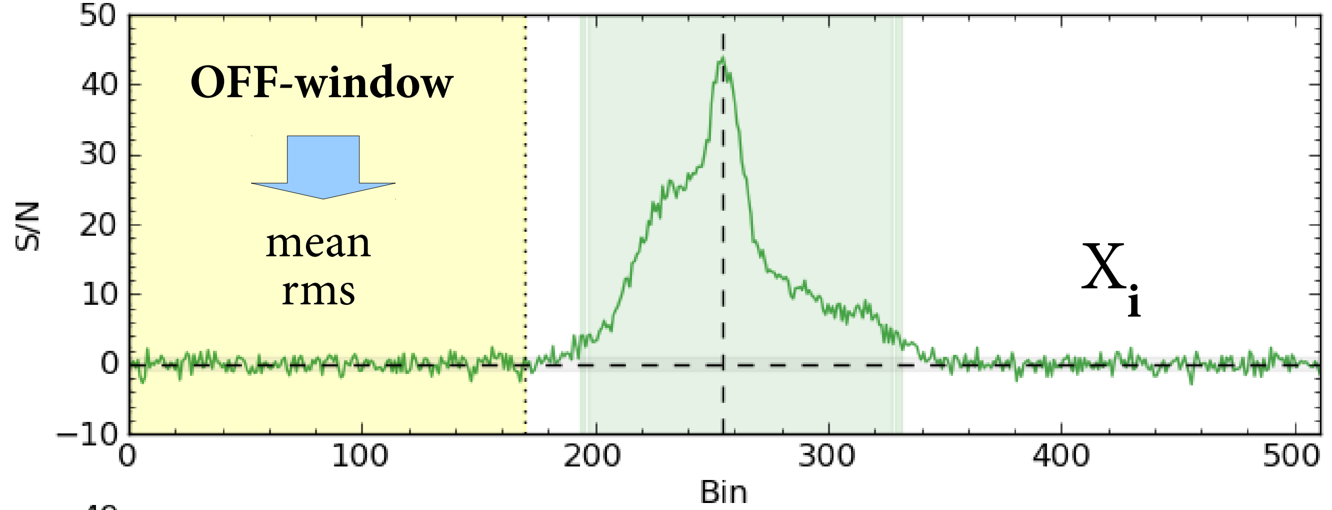
- More info about the LOFAR pulsar calibrations in the following papers:

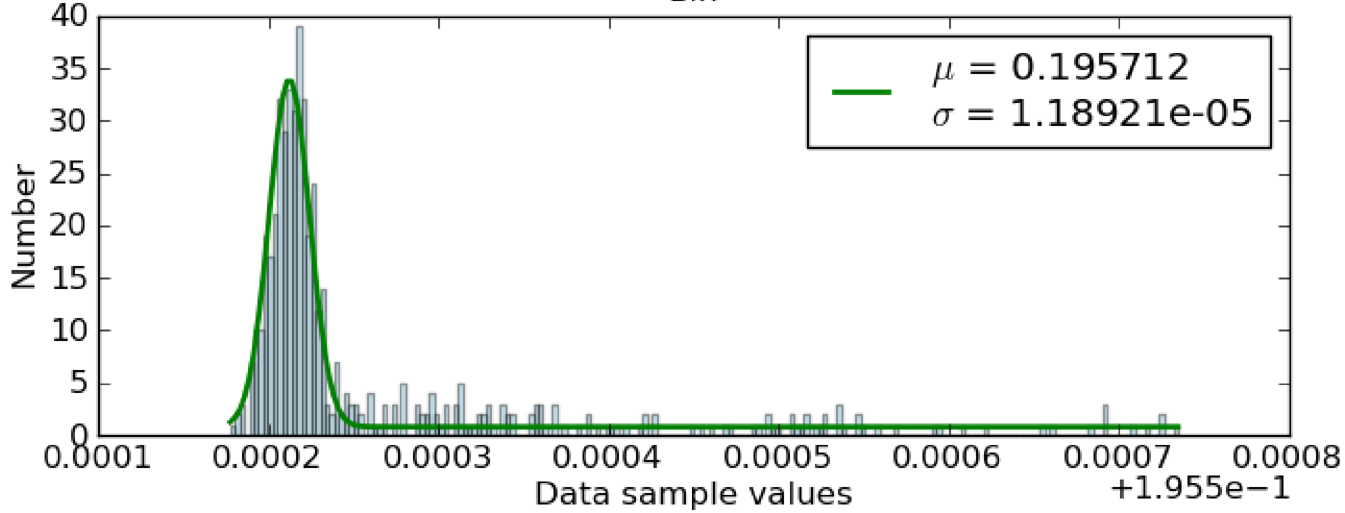
  → «A LOFAR Census of MSPs», Kondratiev et al. 2016, A&A, 585, 128
  → «A LOFAR Census of non-recycled pulsars:...», Bilous et al. 2016, A&A, 591, 134

- 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)





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

i — profile bin

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

Other factors affecting flux measurements

- Scattering → hard to get S/N, it is underestimated
- Refractive scintillations.
  Can change pulsar flux by a factor of ~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 ~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 ~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 ~40%

use -h option to get help

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 <u>output</u> 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

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)

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)

1. I will use as an example some HBA Census data from LC1_003, namely L202460
2. Create the working directory:
   → cd /data/scratch/LOFARSCHOOL2018_T9_PULP/home/
   → mkdir -p <your username> [if you have not had it yet]
   → cd <your username>
   → mkdir L202460_fluxcal

3. The data for L202460 retrieved from the LTA are here:
   → /data/scratch/LOFARSCHOOL2018_T9_PULP/data/LC1_003/L202460
   → There, in L202460_Csplots/stokes/SAP0/BEAM0, there are several PSRFITS files, namely:
   - *_S[0-3].ar – these are the original non-dedispersed, non-scrunched, non-RFI-zapped files for all Stokes parameters (S0 — total intensity I, S1 — Q, S2 — U, S3 — V).
   - *_S[0-3].fscr.AR — dedispersed, fscrunched, non-RFI-zapped
   - *_S[0-3].paz.fscr.AR — dedispersed, fscrunched, RFI-zapped (*clean.py* or *paz*)
   - *_S[0-3].paz.fscr.pdmp.AR — dedispersed, fscrunched, Tscrunched, RFI-zapped, DM/period is optimized by the *pdmp*. Because it is Tscrunched, for long 5+ min observations, for lofar_fluxcal.py it is better to use non-*pdmp* archives.

4. For flux calibration we are only interested in Stokes I files, i.e. S0.
5. Copy the file *_S0.paz.fscr.AR to your working directory L202460_fluxcal (or make a soft link)
   → cd /data/scratch/LOFARSCHOOL2018_T9_PULP/home/<your username>/L202460_fluxcal
   → cp /data/scratch/LOFARSCHOOL2018_T9_PULP/data/LC1_003/L202460/L202460_Csplots/stokes/SAP0/BEAM0/B0940+16_L202460_SAP0_BEAM0_S0.paz.fscr.AR .
   → (or use '*ln -s*' instead of *cp* command)
   → mv B0940+16_L202460_SAP0_BEAM0_S0.paz.fscr.AR b0940.ar (just to make the filename simpler)

Lets flux-calibrate some data… (2)

6. Before we proceed further we need to run a couple of commands which are only specific for our CEP3 docker installation, or these LC1_003 data, namely:

→ */home/kondratiev/bin/fix_hamakerjones.sh* – this script fixes the path for the ElementMainResponse program in mscorpol package. Was overlooked in the current docker that we use.

→ *psredit -c ext:obsfreq=158.5922241 -m b0940.ar* – this command is only for our LC1_003 data, which had the wrong value of the center frequency.

→ **NOTE**, it is *always* a good idea to use psredit or vap commands to check the header information of your PSRFITS files before flux calibration to make sure the information there is correct.

→ Another example of extra command that is needed (**but not in this case!**) is when archive file was created by *dspsr* using filterbank *.fil* as the input. Because the normal frequency order in SIGPROC's filterbank file is reversed, in the output .ar file your frequency order will be also reversed with negative value for the bandwidth (bw). Flux calibration scripts do not understand it (although they will throw the warning), and frequency order must be reversed then. You can do it with:

- *pam --reverse_freqs -m <input ar-file>*

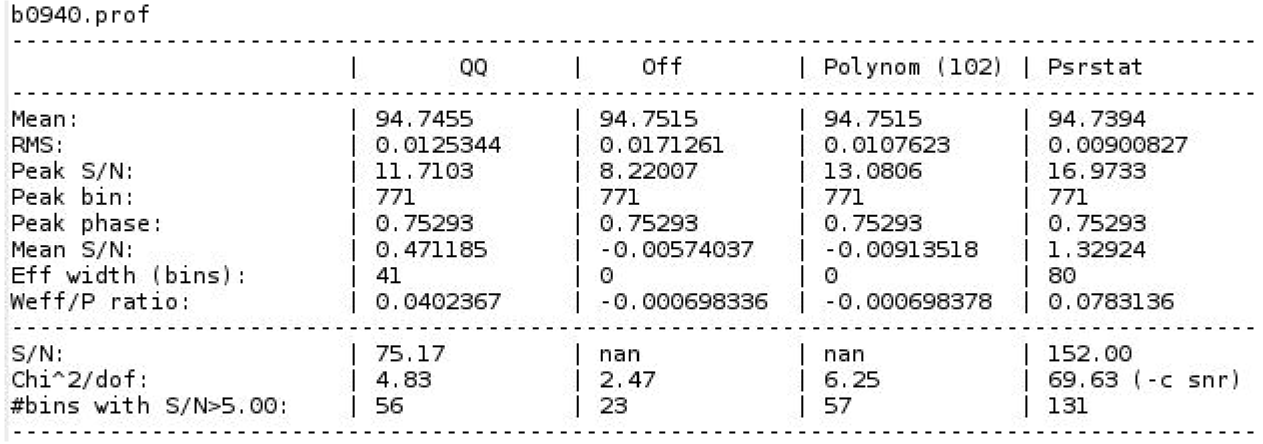
Lets flux-calibrate some data… (3)

7. Then we need to run *snr.py* script to determine the method to calcalute 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 b0940.ar*
- → Then run snr.py command using created .prof file as an input:
  - *snr.py --snrmethod=Off --plot b0940.prof*

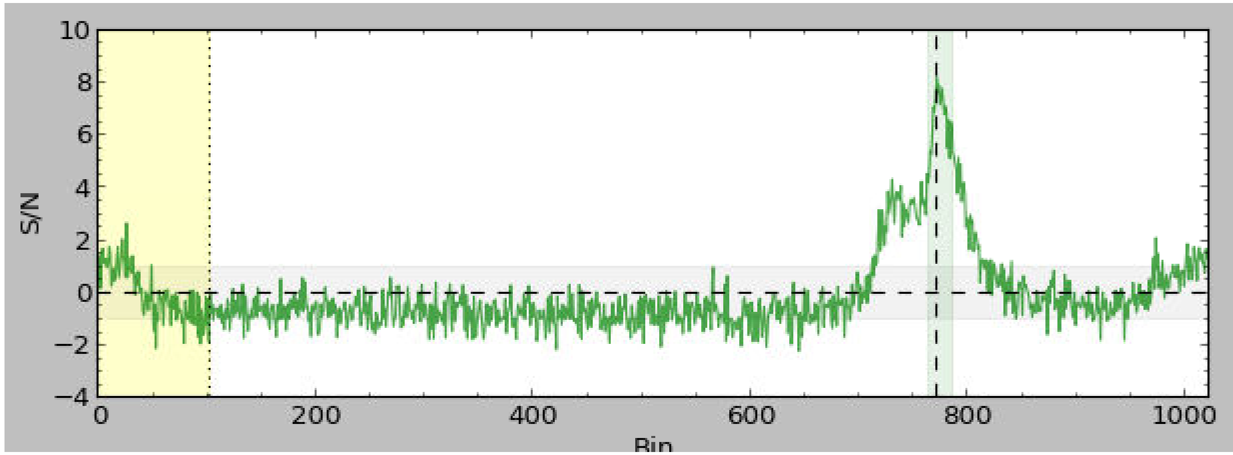
Lets flux-calibrate some data… (4)

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



```
b0940.prof
-------------------------------------------------------------------------------
                     |       QQ      |     Off       | Polynom (102)  | Psrstat
-------------------------------------------------------------------------------
Mean:                | 94.7455       | 94.7515       | 94.7515        | 94.7394
RMS:                 | 0.0125344     | 0.0171261     | 0.0107623      | 0.00900827
Peak S/N:            | 11.7103       | 8.22007       | 13.0806        | 16.9733
Peak bin:            | 771           | 771           | 771            | 771
Peak phase:          | 0.75293       | 0.75293       | 0.75293        | 0.75293
Mean S/N:            | 0.471185      | -0.00574037   | -0.00913518    | 1.32924
Eff width (bins):    | 41            | 0             | 0              | 80
Weff/P ratio:        | 0.0402367     | -0.000698336  | -0.000698378   | 0.0783136
-------------------------------------------------------------------------------
S/N:                 | 75.17         | nan           | nan            | 152.00
Chi^2/dof:           | 4.83          | 2.47          | 6.25           | 69.63 (-c snr)
#bins with S/N>5.00: | 56            | 23            | 57             | 131
-------------------------------------------------------------------------------
```

9. Yellow shows the current Off-pulse window, the horizontal area is 1-sigma region, and vertical blueish region, are the samples above 3-sigma.



10. 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

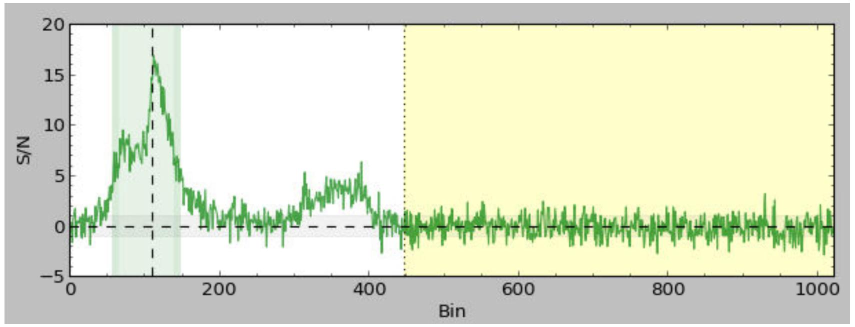
Lets flux-calibrate some data… (5)

11. 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 b0940.prof*

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

```
b0940.prof
---------------------------------------------------------------------------------------
                        |       QQ        |    Off          | Polynom (102)  | Psrstat
---------------------------------------------------------------------------------------
Mean:                   | 94.7455         | 94.739          | 94.739         | 94.7394
RMS:                    | 0.0125344       | 0.00902981      | 0.0112689      | 0.00900827
Peak S/N:               | 11.7103         | 16.976          | 13.6029        | 16.9733
Peak bin:               | 111             | 111             | 111            | 111
Peak phase:             | 0.108398        | 0.108398        | 0.108398       | 0.108398
Mean S/N:               | 0.471185        | 1.37477         | 1.1016         | 1.32924
Eff width (bins):       | 41              | 82              | 82             | 80
Weff/P ratio:           | 0.0402366       | 0.0809831       | 0.0809832      | 0.0783136
---------------------------------------------------------------------------------------
S/N:                    | 75.17           | 154.59          | 123.87         | 152.00
Chi^2/dof:              | 4.83            | 10.77           | 6.92           | 69.63 (-c snr)
#bins with S/N>5.00:    | 56              | 87              | 74             | 131
---------------------------------------------------------------------------------------
AUTO-OFF: --off-left 447 --off-right 1024 -r 0.644531 -b 1
```

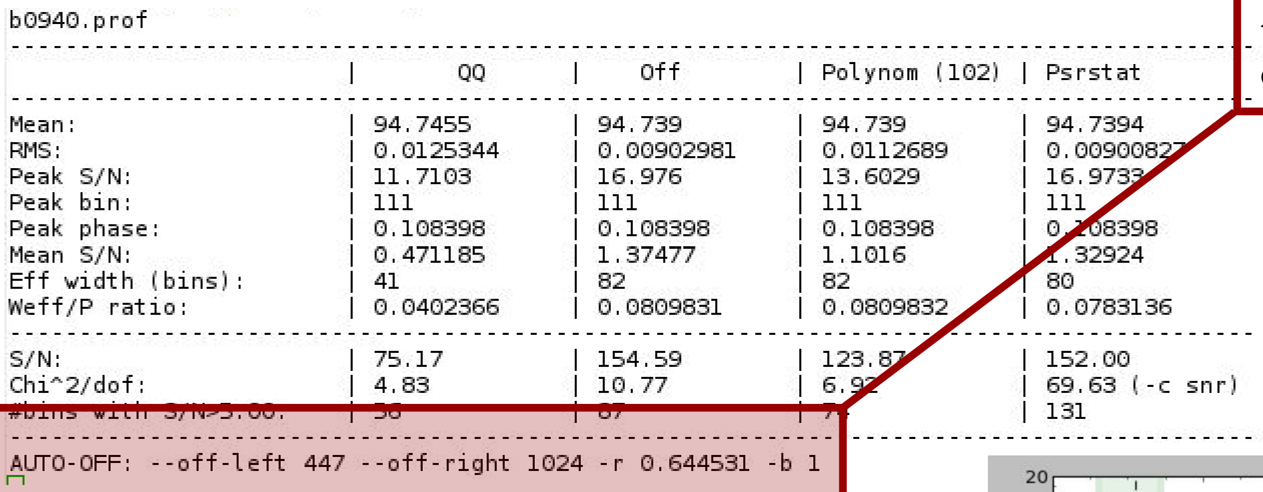
13. If you are not satisfied with the Off-pulse selection, you can adjust it by using the option --*auto-off-adjust* to adjust **ON**-pulse window



Lets flux-calibrate some data… (5)

11. 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 b0940.prof*

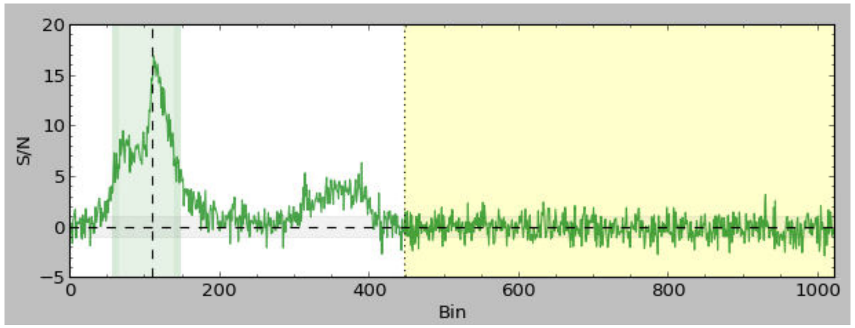
12. And the output and plot will look like this:



```
b0940.prof
------------------------------------------------------------------------------------------------
                        |      QQ        |    Off         | Polynom (102)  | Psrstat
------------------------------------------------------------------------------------------------
Mean:                   | 94.7455        | 94.739         | 94.739         | 94.7394
RMS:                    | 0.0125344      | 0.00902981     | 0.0112689      | 0.00900827
Peak S/N:               | 11.7103        | 16.976         | 13.6029        | 16.9733
Peak bin:               | 111            | 111            | 111            | 111
Peak phase:             | 0.108398       | 0.108398       | 0.108398       | 0.108398
Mean S/N:               | 0.471185       | 1.37477        | 1.1016         | 1.32924
Eff width (bins):       | 41             | 82             | 82             | 80
Weff/P ratio:           | 0.0402366      | 0.0809831      | 0.0809832      | 0.0783136
------------------------------------------------------------------------------------------------
S/N:                    | 75.17          | 154.59         | 123.87         | 152.00
Chi^2/dof:              | 4.83           | 10.77          | 6.92           | 69.63 (-c snr)
#bins with S/N>3.00:    | 58             | 87             | 74             | 131
------------------------------------------------------------------------------------------------
AUTO-OFF: --off-left 447 --off-right 1024 -r 0.644531 -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*.

13. 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)

14. For calibration it is important to know what tiles/dipoles were flagged as bad, or ar 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.

15. 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.

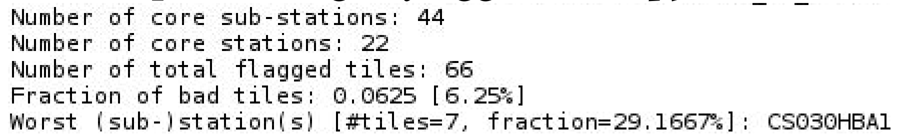
16. There is *getState.py* script together with the ascii db about the flagged tiles that can be downloaded now from ASTRON's resource page, and will be part of the LOFAR-BF-pulsar-scripts Github repository as well. Make sure to update the db file «*hardwire_states_latest.txt*» file regularly especially for calibration of new observations.

17. To get the flagged tiles/dipoles info for a given timestamp, array config, run:
   → First you also 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 /data/scratch/LOFARSCHOOL2018_T9_PULP/data/LC1_003/L202460/L202460_Csplots/stokes/SAP0/BEAM0/L202460_SAP000_B000_S0_P000_bf.h5
   → */home/kondratiev/bin/lofar_antenna_state/getState.py -i <h5-file> -s all -f flagged.txt*

18. 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 <h5-file>*

Lets flux-calibrate some data… (7)

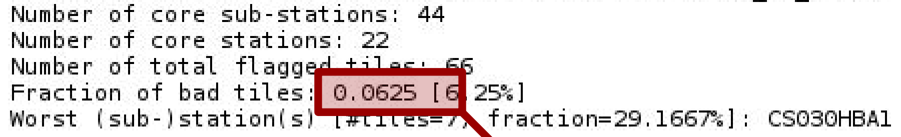
19. The output from *get_flagged_tiles.py* looks like this:



```
Number of core sub-stations: 44
Number of core stations: 22
Number of total flagged tiles: 66
Fraction of bad tiles: 0.0625 [6.25%]
Worst (sub-)station(s) [#tiles=7, fraction=29.1667%]: CS030HBA1
```

Lets flux-calibrate some data… (7)

19. The output from *get_flagged_tiles.py* looks like this:



```
Number of core sub-stations: 44
Number of core stations: 22
Number of total flagged tiles: 66
Fraction of bad tiles: 0.0625 [6.25%]
Worst (sub-)station(s) [#tiles=7, fraction=29.1667%]: CS030HBA1
```

This value must be used in the *--flagged* option for *lofar_fluxcal.py* or *lofar_psrflux.py*

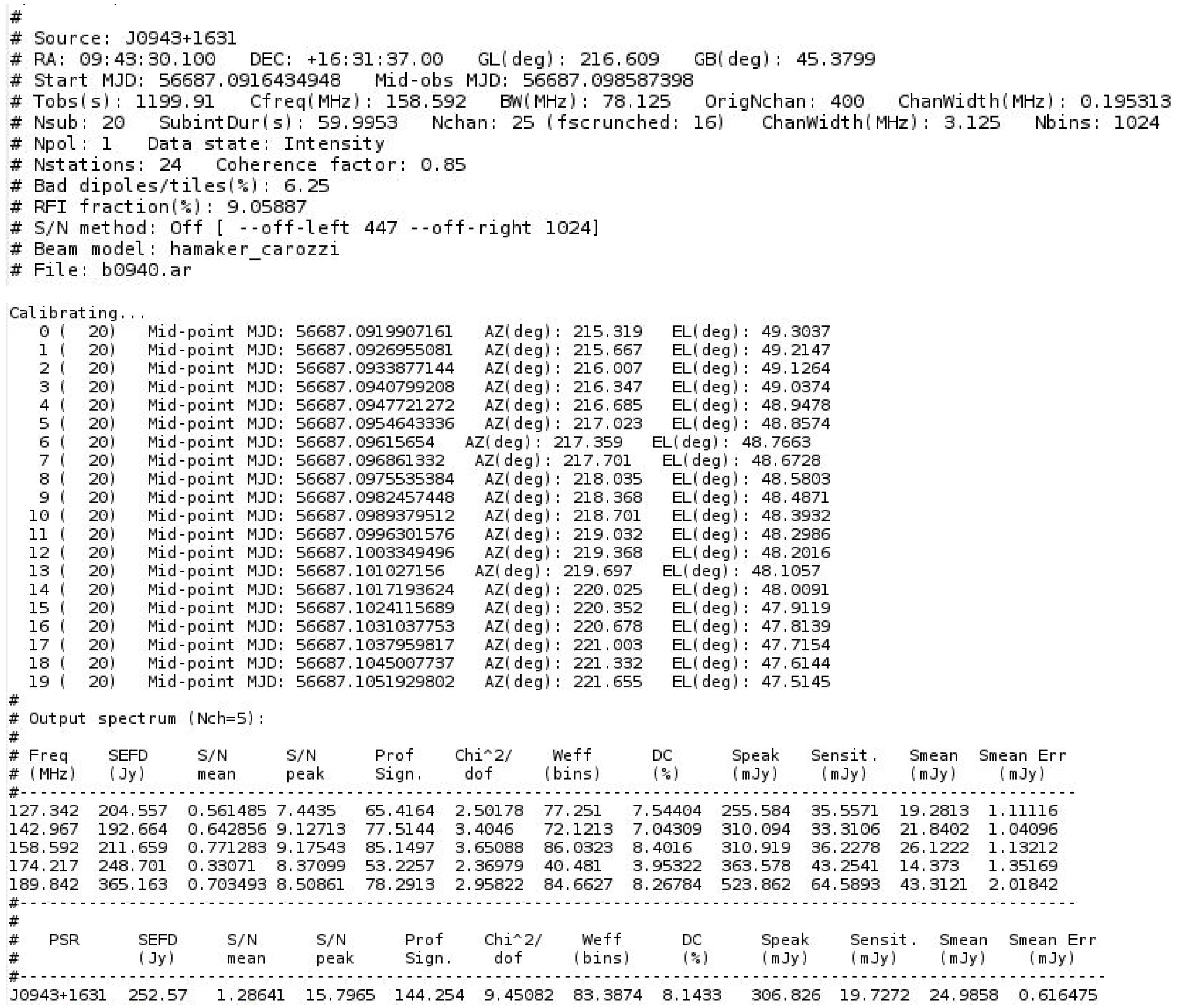
20. Now, it is time to run flux calibration script:

→ *lofar_fluxcal.py -f 16 --flagged 0.0625 --meta L202460_SAP000_B000_S0_P000_bf.h5 --off-left 447 --off-right 1024 -r 0.644531 -b 1 --snrmethod=Off --model hamaker_carozzi --plot-saveonly --plot --spectrum 5 -v b0940.ar*

→ For the full list of available options use run: *lofar_fluxcal.py -h*
→ *-f 16* – to fscrunch by a factor of 16, otherwise it will run much longer
→ you can get more verbose output with *--vv* option

Lets flux-calibrate some data… (8)

21. The output from *lofar_fluxcal.py* can look like this:

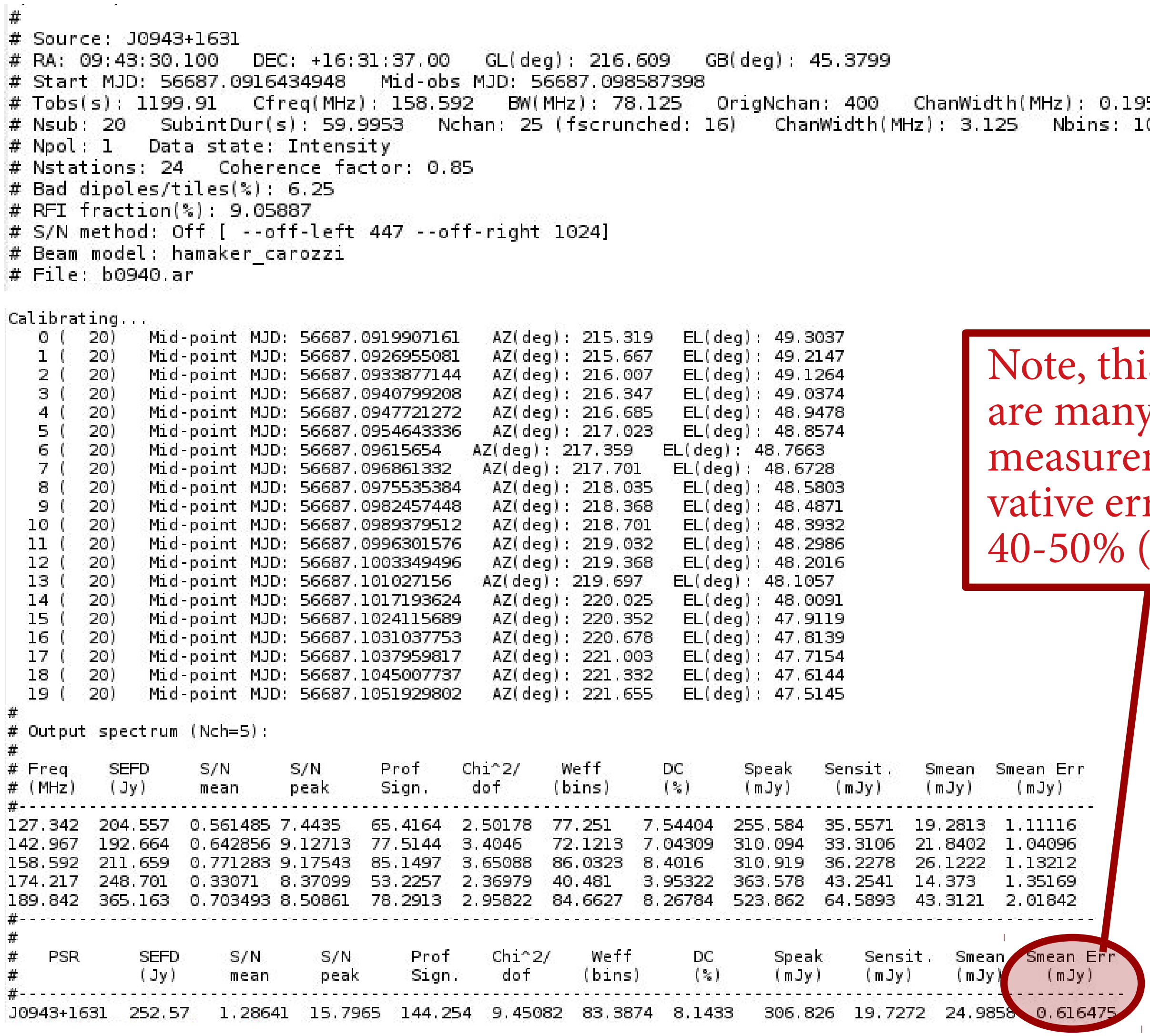


```
#
# Source: J0943+1631
# RA: 09:43:30.100   DEC: +16:31:37.00   GL(deg): 216.609   GB(deg): 45.3799
# Start MJD: 56687.0916434948   Mid-obs MJD: 56687.098587398
# Tobs(s): 1199.91   Cfreq(MHz): 158.592   BW(MHz): 78.125   OrigNchan: 400   ChanWidth(MHz): 0.195313
# Nsub: 20   SubintDur(s): 59.9953   Nchan: 25 (fscrunched: 16)   ChanWidth(MHz): 3.125   Nbins: 1024
# Npol: 1   Data state: Intensity
# Nstations: 24   Coherence factor: 0.85
# Bad dipoles/tiles(%): 6.25
# RFI fraction(%): 9.05887
# S/N method: Off [ --off-left 447 --off-right 1024]
# Beam model: hamaker_carozzi
# File: b0940.ar

Calibrating...
   0 (  20)   Mid-point MJD: 56687.0919907161   AZ(deg): 215.319   EL(deg): 49.3037
   1 (  20)   Mid-point MJD: 56687.0926955081   AZ(deg): 215.667   EL(deg): 49.2147
   2 (  20)   Mid-point MJD: 56687.0933877144   AZ(deg): 216.007   EL(deg): 49.1264
   3 (  20)   Mid-point MJD: 56687.0940799208   AZ(deg): 216.347   EL(deg): 49.0374
   4 (  20)   Mid-point MJD: 56687.0947721272   AZ(deg): 216.685   EL(deg): 48.9478
   5 (  20)   Mid-point MJD: 56687.0954643336   AZ(deg): 217.023   EL(deg): 48.8574
   6 (  20)   Mid-point MJD: 56687.09615654   AZ(deg): 217.359   EL(deg): 48.7663
   7 (  20)   Mid-point MJD: 56687.096861332   AZ(deg): 217.701   EL(deg): 48.6728
   8 (  20)   Mid-point MJD: 56687.0975535384   AZ(deg): 218.035   EL(deg): 48.5803
   9 (  20)   Mid-point MJD: 56687.0982457448   AZ(deg): 218.368   EL(deg): 48.4871
  10 (  20)   Mid-point MJD: 56687.0989379512   AZ(deg): 218.701   EL(deg): 48.3932
  11 (  20)   Mid-point MJD: 56687.0996301576   AZ(deg): 219.032   EL(deg): 48.2986
  12 (  20)   Mid-point MJD: 56687.1003349496   AZ(deg): 219.368   EL(deg): 48.2016
  13 (  20)   Mid-point MJD: 56687.101027156   AZ(deg): 219.697   EL(deg): 48.1057
  14 (  20)   Mid-point MJD: 56687.1017193624   AZ(deg): 220.025   EL(deg): 48.0091
  15 (  20)   Mid-point MJD: 56687.1024115689   AZ(deg): 220.352   EL(deg): 47.9119
  16 (  20)   Mid-point MJD: 56687.1031037753   AZ(deg): 220.678   EL(deg): 47.8139
  17 (  20)   Mid-point MJD: 56687.1037959817   AZ(deg): 221.003   EL(deg): 47.7154
  18 (  20)   Mid-point MJD: 56687.1045007737   AZ(deg): 221.332   EL(deg): 47.6144
  19 (  20)   Mid-point MJD: 56687.1051929802   AZ(deg): 221.655   EL(deg): 47.5145
#
# 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)
#----------------------------------------------------------------------------------------------------------
127.342  204.557  0.561485 7.4435   65.4164  2.50178  77.251   7.54404  255.584  35.5571  19.2813  1.11116
142.967  192.664  0.642856 9.12713  77.5144  3.4046   72.1213  7.04309  310.094  33.3106  21.8402  1.04096
158.592  211.659  0.771283 9.17543  85.1497  3.65088  86.0323  8.4016   310.919  36.2278  26.1222  1.13212
174.217  248.701  0.33071  8.37099  53.2257  2.36979  40.481   3.95322  363.578  43.2541  14.373   1.35169
189.842  365.163  0.703493 8.50861  78.2913  2.95822  84.6627  8.26784  523.862  64.5893  43.3121  2.01842
#----------------------------------------------------------------------------------------------------------
#
#   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)
#-------------------------------------------------------------------------------------------------------------
J0943+1631   252.57   1.28641  15.7965  144.254  9.45082  83.3874  8.1433   306.826  19.7272  24.9858  0.616475
```

Lets flux-calibrate some data… (8)

21. The output from *lofar_fluxcal.py* can look like this:



```
#
# Source: J0943+1631
# RA: 09:43:30.100   DEC: +16:31:37.00   GL(deg): 216.609   GB(deg): 45.3799
# Start MJD: 56687.0916434948   Mid-obs MJD: 56687.098587398
# Tobs(s): 1199.91   Cfreq(MHz): 158.592   BW(MHz): 78.125   OrigNchan: 400   ChanWidth(MHz): 0.195313
# Nsub: 20   SubintDur(s): 59.9953   Nchan: 25 (fscrunched: 16)   ChanWidth(MHz): 3.125   Nbins: 1024
# Npol: 1   Data state: Intensity
# Nstations: 24   Coherence factor: 0.85
# Bad dipoles/tiles(%): 6.25
# RFI fraction(%): 9.05887
# S/N method: Off [ --off-left 447 --off-right 1024]
# Beam model: hamaker_carozzi
# File: b0940.ar

Calibrating...
    0 (   20)   Mid-point MJD: 56687.0919907161   AZ(deg): 215.319   EL(deg): 49.3037
    1 (   20)   Mid-point MJD: 56687.0926955081   AZ(deg): 215.667   EL(deg): 49.2147
    2 (   20)   Mid-point MJD: 56687.0933877144   AZ(deg): 216.007   EL(deg): 49.1264
    3 (   20)   Mid-point MJD: 56687.0940799208   AZ(deg): 216.347   EL(deg): 49.0374
    4 (   20)   Mid-point MJD: 56687.0947721272   AZ(deg): 216.685   EL(deg): 48.9478
    5 (   20)   Mid-point MJD: 56687.0954643336   AZ(deg): 217.023   EL(deg): 48.8574
    6 (   20)   Mid-point MJD: 56687.09615654   AZ(deg): 217.359   EL(deg): 48.7663
    7 (   20)   Mid-point MJD: 56687.096861332   AZ(deg): 217.701   EL(deg): 48.6728
    8 (   20)   Mid-point MJD: 56687.0975535384   AZ(deg): 218.035   EL(deg): 48.5803
    9 (   20)   Mid-point MJD: 56687.0982457448   AZ(deg): 218.368   EL(deg): 48.4871
   10 (   20)   Mid-point MJD: 56687.0989379512   AZ(deg): 218.701   EL(deg): 48.3932
   11 (   20)   Mid-point MJD: 56687.0996301576   AZ(deg): 219.032   EL(deg): 48.2986
   12 (   20)   Mid-point MJD: 56687.1003349496   AZ(deg): 219.368   EL(deg): 48.2016
   13 (   20)   Mid-point MJD: 56687.101027156   AZ(deg): 219.697   EL(deg): 48.1057
   14 (   20)   Mid-point MJD: 56687.1017193624   AZ(deg): 220.025   EL(deg): 48.0091
   15 (   20)   Mid-point MJD: 56687.1024115689   AZ(deg): 220.352   EL(deg): 47.9119
   16 (   20)   Mid-point MJD: 56687.1031037753   AZ(deg): 220.678   EL(deg): 47.8139
   17 (   20)   Mid-point MJD: 56687.1037959817   AZ(deg): 221.003   EL(deg): 47.7154
   18 (   20)   Mid-point MJD: 56687.1045007737   AZ(deg): 221.332   EL(deg): 47.6144
   19 (   20)   Mid-point MJD: 56687.1051929802   AZ(deg): 221.655   EL(deg): 47.5145
#
# 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)
#------------------------------------------------------------------------------------------------------------------------
127.342  204.557  0.561485 7.4435    65.4164   2.50178   77.251    7.54404   255.584   35.5571   19.2813   1.11116
142.967  192.664  0.642856 9.12713   77.5144   3.4046    72.1213   7.04309   310.094   33.3106   21.8402   1.04096
158.592  211.659  0.771283 9.17543   85.1497   3.65088   86.0323   8.4016    310.919   36.2278   26.1222   1.13212
174.217  248.701  0.33071  8.37099   53.2257   2.36979   40.481    3.95322   363.578   43.2541   14.373    1.35169
189.842  365.163  0.703493 8.50861   78.2913   2.95822   84.6627   8.26784   523.862   64.5893   43.3121   2.01842
#------------------------------------------------------------------------------------------------------------------------
#
#   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)
#------------------------------------------------------------------------------------------------------------------------
J0943+1631   252.57    1.28641   15.7965   144.254   9.45082   83.3874   8.1433    306.826   19.7272   24.9858   0.616475
```

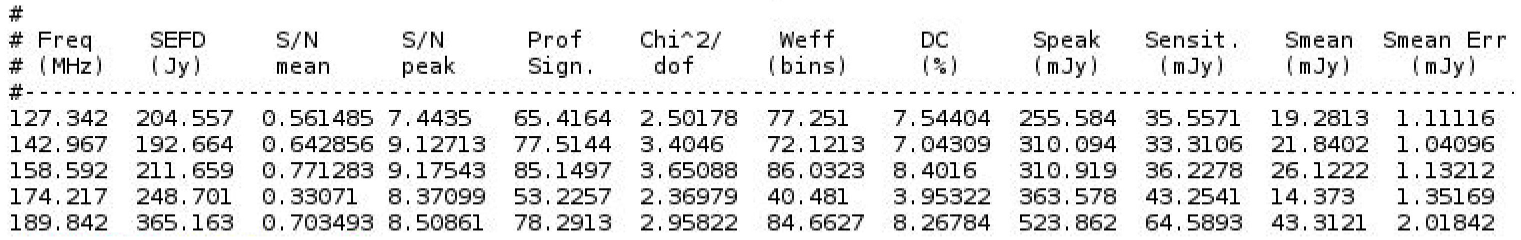
Note, this is only the nominal error, there are many factors that can affect flux measurements, and we estimate the conservative error on the flux to be within 40-50% (see Bilous et al. 2016)

Lets flux-calibrate some data… (9)

22. The more detailed output is also being saved to *b0940.calib.flux.ascii*

23. The calibrated ar-file (in units of mJy) is also created: *b0940.calib.ar*

24. Also ascii files with flux measurements across the band are created if *--spectrum* option was used. In our case file *b0940.calib.spectrum.5.txt* was created, where «5» means that band was split in 5 frequency parts. The file reads as:



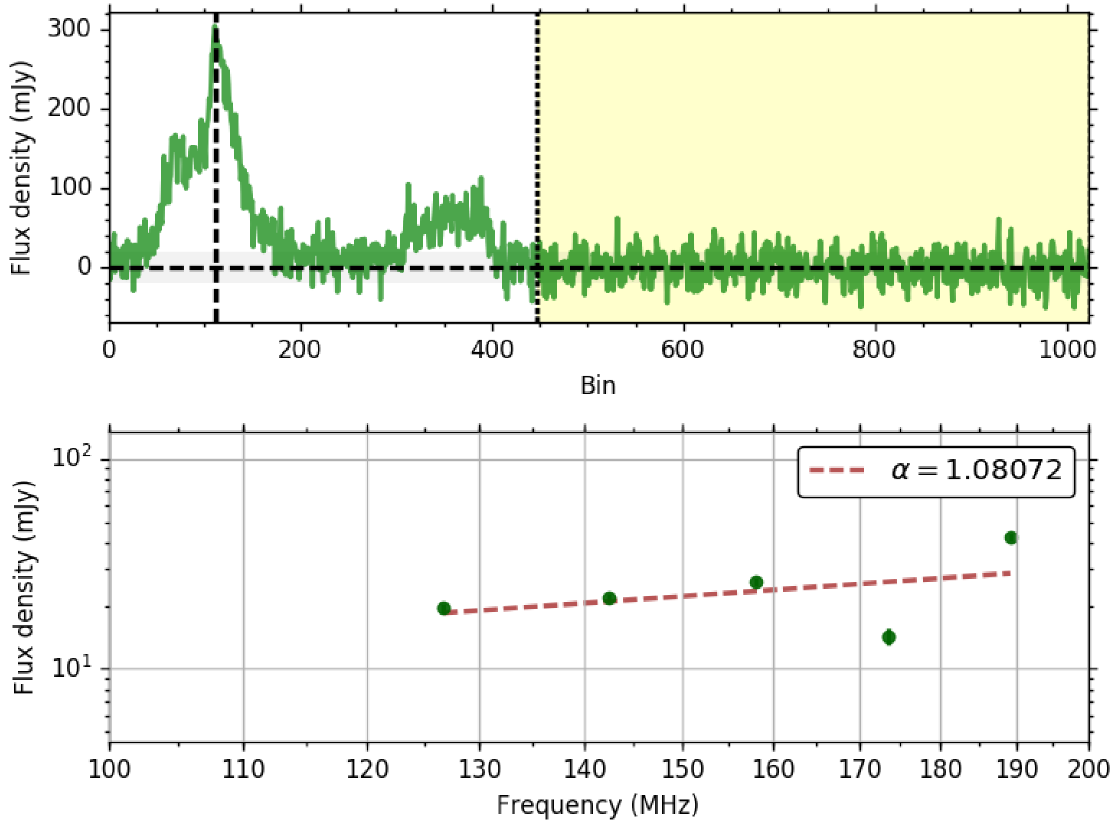
```
#
# Freq     SEFD     S/N      S/N      Prof     Chi^2/   Weff     DC       Speak    Sensit.  Smean    Smean Err
# (MHz)    (Jy)     mean     peak     Sign.    dof      (bins)   (%)      (mJy)    (mJy)    (mJy)    (mJy)
#----------------------------------------------------------------------------------------------------------------
127.342  204.557  0.561485 7.4435   65.4164  2.50178  77.251   7.54404  255.584  35.5571  19.2813  1.11116
142.967  192.664  0.642856 9.12713  77.5144  3.4046   72.1213  7.04309  310.094  33.3106  21.8402  1.04096
158.592  211.659  0.771283 9.17543  85.1497  3.65088  86.0323  8.4016   310.919  36.2278  26.1222  1.13212
174.217  248.701  0.33071  8.37099  53.2257  2.36979  40.481   3.95322  363.578  43.2541  14.373   1.35169
189.842  365.163  0.703493 8.50861  78.2913  2.95822  84.6627  8.26784  523.862  64.5893  43.3121  2.01842
```

→ 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).

25. 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)

26. There is also the diagnostic plot created (if options *--plot* and *--plot-saveonly* were used). In our case the image file is *b0940.calib.flux.png* and it looks like this:

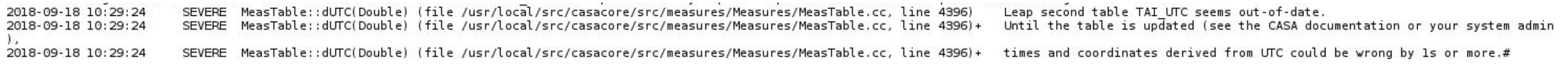


The spectrum with fit is also shown if option *--spectrum* was used. Otherwise, file will only have the upper sub-plot

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

Lets flux-calibrate some data… (11)

27. 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.#
```

28. 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 Cacasore installation later.

29. 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

30. 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.

31. 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 the case for a FE observation or observation with an International station.

The End