Tutorial on using pycryoPAF to calibrate and image Parkes cryoPAF shared-risk spectral-line observations

Lister Staveley-Smith, ICRAR

This version: 6 April 2025

Table of Contents

Introduction *2*
Requirements *2*
Data formats and data volume *2*
Command-line use *3*
Help *3*
Data inspection *4*
SDHDF content 4
Plotting spectra 5
SHDF concatenation *6*
Pre-processing *6*
Positions and frequencies 6
Intensity calibration *7*
Splitting on-source and off-source data 7
Flux density (continuum sources) 7
Brightness temperature (HI) 7
Validation 8
Spectral baseline removal *8*
Point-and-shoot 8
Multiple point-and-shoot 9
Continuous scans (active or drift) 9
Imaging *10*
Visualisation *11*

Introduction

This document describes the use of various routines to calibrate and image spectral-line data from the cryoPAF instrument on the Parkes/Murriyang telescope using the pycryoPAF suite of python routines. The routines were developed for the purposes of rapid scientific commissioning and are not intended to be of National Facility standard. Bugs may not be fixed and the routines may not be interoperable with future upgrades of the SDHDF data format or with all modes of spectral-line observing (e.g. binned pulsar mode). The output format is SDHDF for spectral data and FITS for image data. The code is written using widely available python modules. It is available on a shared-risk basis and may be modified or shared according to CC BY-NC. You are welcome to contact the author with queries and bug reports.

Requirements

The code has been written and tested using python 3.11/3.12 and may not be compatible with earlier versions, and definitely not with python 2. Each routine has its own requirements, but the following is a summary of the most common modules required. If you use conda, they may be installed using the command "`conda install <module>`":

```
numpy                                scipy                                os
h5py                                 pkg_resources                        getopt
astropy                              json                                 glob
matplotlib                           sys
pylab                                re
```

Data formats and data volume

cryoPAF data is written in the very efficient SDHDF format, as described by Toomey et al. (2024) A&C, 47, 100804. This format stores spectral-line data in multidimensional arrays (typically time, polarisation, frequency, bin) in the following hierarchy: beam number, then sub-band (zoom) number. Metadata (positions, frequencies etc) are easily extracted.

Some pycryoPAF routines can modify data in-situ, which is very efficient. However most will want to write another file. The output files are also written in SDHDF format, but these have extra metadata and are 'chunked' for even more efficient access and expandable so that extra data may be appended.

With typical observing parameters (72 beams, 2 zoom bands, 2 sec integration time), the data volumes are quite large (2TB/day), so please be careful to choose values for bandwidth, frequency resolution, integration time etc which are suitable for your experiment.

Imaging data is output in FITS format using standard WCS coordinate systems. The gridding process may alter the flux scale and beam size. Please check these with care.

Calibration data is input/output using "`json`" format text files.

Command-line use

pycryoPAF routines require command-line input, so are easily scriptable. Most use the “getopt” package, but simpler routines just require an argument. For example:

`paf-list.py *.hdf`	This will summarise the contents of hdf files in the current directory
`paf-scan.py -s paf_241118_100007.hdf`	This uses “getopt” to separate the option (`“-s”`) from the parameters (filename).

If the python routines are not executable, make them so (`chmod +x paf-list.py`) or use full-length command such as:

```
python paf-list.py *.hdf
```

Help

For those routines using getopt, help is available with the “-?” query. For example, the command:

```
paf-scan.py -?
```

results in the following output:

```
OPTION   DESCRIPTION

?        help
a=       start sub-band
b=       end sub-band
A=       start beam
B=       end beam
d=       lower plot limit
e=       number of smoothing channels
F=       channel range for polynomial fit (comma-separated)
g        apply this calibration file
h=       apply this calibration to other beams (first entry)
H=       Stokes I
i        use medians for all time averages
median   no plot
n        unblocked plot
N        Output data file
o=       polynomial baseline order (0 and -ve values applied pre-time-average)
p=       source (signal) spectrum
sig=     upper plot limit
u=       start channel number (within each sub-band)
v=       end channel number (within each sub-band)
w=       large font (for publication)
zoom     number of SVD singular values
```

Options need to be prefaced with a “-” sign. Some options (e.g. `“-i”`) are switches and don’t require an argument. Other options (e.g. `“-s <filename>”`) require an argument.

Data inspection

SDHDF content

A useful tool to explore the data structure of HDF5 files is silx, which was developed for examining data from synchrotron facilities. It can be installed easily (`conda install silx -c conda-forge`), and allows viewing of the data hierarchy, data structure and can even make plots. Launch it from the unix command line using the command ‘`silx view`’.

pycryoPAF provides a quick method to list the basic contents of many HDF5 files (namely filename, source name, #beams, #subbands, RA, DEC, cal status, integration time). This is useful for bookkeeping purposes:

`paf-list.py *.hdf`	This will summarise the contents of hdf files in the current directory.

Output:

```
HDR_DEFN_VERSION (first file): 4.0
  HDF file             Source              B I   RA          Dec      cal   sec
paf_250316_231312_B71 NGC6744              1 2 19:09:45.3 -63:51:21 OFF 583.4
paf_250316_232400_B71 NGC6744OFF           1 2 19:03:45.3 -63:51:21 OFF 593.6
test.hdf              NGC6744              1 2 19:09:45.3 -63:51:21 OFF 583.4
```

If only a single HDF5 file is selected, `paf-list` will do a deep dive, summarise the file contents on a cycle-by-cycle basis, and plot all sky positions:

`paf-list.py paf_250316_231312_B71.hdf`	This will summarise the contents of a single hdf file and plot all positions in the scan .

Output:

```
Only 1 file specified, so executing deep dive for paf_250316_231312_B71.hdf band_SB0
HDR_DEFN_VERSION (first file): 4.0
  HDF file             Source              B I   RA          Dec      cal   sec
2025-03-16-23:13:17.1 NGC6744              1 2 19:09:45.3 -63:51:21 OFF 10.2 (0)
2025-03-16-23:13:27.3 NGC6744              1 2 19:09:45.3 -63:51:21 OFF 10.2 (1)
2025-03-16-23:13:37.5 NGC6744              1 2 19:09:45.3 -63:51:21 OFF 10.2 (2)
2025-03-16-23:13:47.8 NGC6744              1 2 19:09:45.3 -63:51:21 OFF 10.2 (3)
2025-03-16-23:13:58.0 NGC6744              1 2 19:09:45.3 -63:51:21 OFF 10.2 (4)
2025-03-16-23:14:08.3 NGC6744              1 2 19:09:45.3 -63:51:21 OFF 10.2 (5)
2025-03-16-23:14:18.5 NGC6744              1 2 19:09:45.3 -63:51:21 OFF 10.2 (6)
2025-03-16-23:14:28.7 NGC6744              1 2 19:09:45.3 -63:51:21 OFF 10.2 (7)
2025-03-16-23:14:39.0 NGC6744              1 2 19:09:45.3 -63:51:21 OFF 10.2 (8)
2025-03-16-23:14:49.2 NGC6744              1 2 19:09:45.3 -63:51:21 OFF 10.2 (9)
2025-03-16-23:14:59.4 NGC6744              1 2 19:09:45.3 -63:51:21 OFF 10.2 (10)
etc.
```

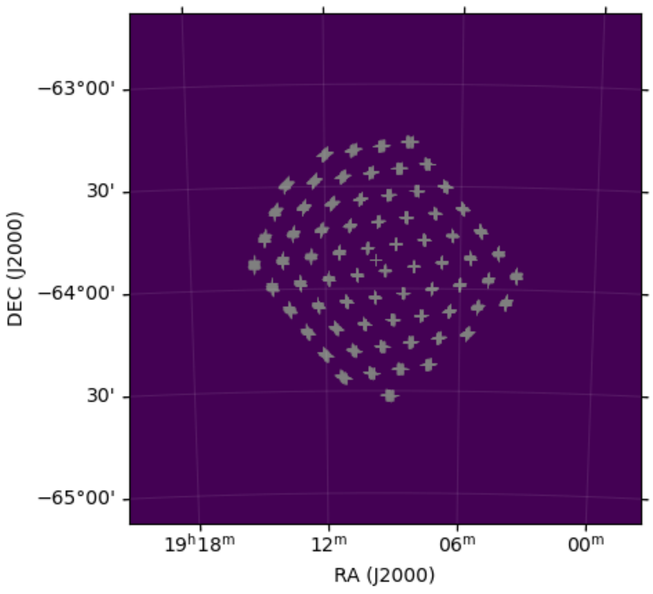


Figure 1: paf-list will project the sky positions onto an RA-DEC grid if only a single HDF file is examined.

Plotting spectra

Although cryoPAF data sets are very large, it is important to visually inspect subsets to ensure it is what you expect and is up-to-standard. Both `paf.py` and `paf-scan.py` can optionally plot time-averaged spectra, frequency-averages and frequency-time waterfalls for individual beam and subbands.

`paf-scan.py -s paf_250316_231312_B71.hdf -a 1 -z`	This will plot a time-averaged spectrum, power v. time, and a waterfall plot for each XX, YY polarisation for subband 1.

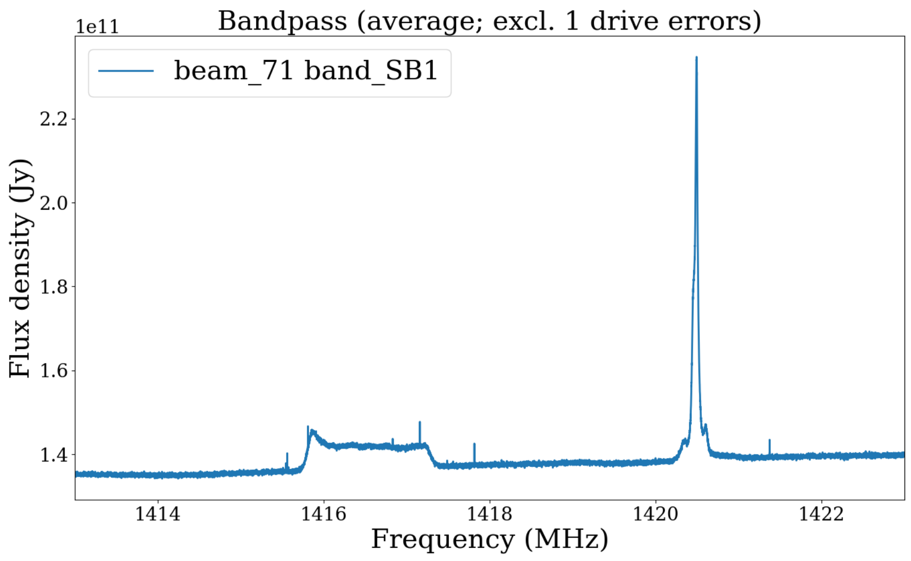


Figure 2: paf-scan can plot time-averaged spectra for inspection purposes (uncalibrated data).

SHDF concatenation

Data are normally delivered from the telescope in the form of one beam per HDF file[1]. Whilst this can be useful for rapid transfer using rsync etc, it quickly becomes inconvenient to have thousands of files to process, even though many pycryoPAF routines are capable of iterating over multiple HDF files.

It is recommended to use the hierarchical nature of HDF to concatenate all beams into a single file. If space is short, the original beam files (which are available in the ATOA archive anyway) can be deleted.

`paf-join.py paf_250316_231312_B*.hdf`	This will concatenate all beam data into a single file with the automatically-generated name '`paf_250316_231312.hdf5`'.

It is suggested that the extension 'hdf5' is used for processed data.

Pre-processing

Positions and frequencies

Early cryoPAF data only contained positions for the central beam (usually B71) or, in the case of scan/calibration data, completely incorrect RA/Dec data. This can be fixed by using the valid Az/El and focal angle data together with the beam geometry file. Re-calculated coordinates are updated in-situ – i.e. no new output file is generated. Position re-calculation *should* no longer be needed, but see 'help' if this is not the case. Note that there will be arcmin errors in this calculation due to master equatorial offsets etc.

Frequencies contained in HDF files from the telescope are 'topocentric', so they will need to be converted to a more suitable reference frame. Again, frequencies are modified in-situ for each beam and subband. No new file is created. SDHDF format uses a single frequency array per subband, so the reference frame conversion is calculated at the mid-UT and mid-position for each beam in each file. The reference frame label is stored in the SDHDF file, so `paf-fix` will not re-apply the Doppler/frame correction if already applied.

`paf-fix.py -s paf_250316_231312.hdf5 -f BARY`	This will convert the frequency scale from topocentric to barycentric (the available frames are BARY and LSRK).

It is recommended that `paf-fix` be executed in dry-run mode (`paf-fix.py -d`) before committing to recalculating any positions/frequencies. Note that astropy may have to download updated Earth orientation data.

`paf-fix` can also modify source names in the case of incorrect metadata. For historical reasons, a parset file (e.g. closepack72.parset) needs to be in the directory, even when not re-calculating positions.

[1] Note that cryoPAF beam and subband numbering are zero-based.

Intensity calibration

Currently, the cryoPAF noise source is not enabled, so calibration is only maintained when the gain is constant. Frequent calibration is currently recommended.

Splitting on-source and off-source data

A typical calibration will involve observing a calibration source by stepping through all the beams without switching off the data-taking. So, each beam in the HDF file will contain a few on-source integration cycles, a lot of off-source cycles, and a lot of cycles with 'drive errors'.[2]

`paf-calsplit.py` will examine the calibration data and will create two output file – an 'on-source' file, and an 'off-source' file. Recognised calibrators are 1934-638, Hydra A, 0407-658, 0823-500, S8 and S9. The latter two are HI brightness temperature standards.

`paf-calsplit.py -s paf_250223_233303.hdf5 (-a 1)`	This will split a file into on-source and off-source data. Beam and subband selection is possible.

For obvious reasons, the off-source file will be much larger than the on-source file.

Flux density (continuum sources)

Calibration parameters are written to a 'json' calibration file containing the scaling parameters for each beam, subband and polarisation. There is no facility to include frequency dependence within a subband, although multiple calibration files can be calculated using different frequency ranges.

`paf.py -s paf_250223_233303_ON.hdf5 -r paf_250223_233303_OFF.hdf5 -a 1 -v 3000 -w 12000 -A 0 -B 72 -g -n`	This writes a calibration file (`paf_250223_233303_ON_CAL.json`) for subband SB01 by examing on-source and off-source data from channels 3000 to 12000 for all 72 beams.

It is important to select a source and RFI-free calibration region to obtain good calibration parameters (although there is some robustness in the algorithm in both time and frequency domains). Calibration parameters should normally be obtained and applied for each polarisation (XX, YY) separately (npol=2), but if the power levels are reasonably equal it is ok to use the "-i" option to just calculate a Stokes I calibration (npol=1). The "`-n`" option suppresses plots, which is useful for munching through lots of data.

Brightness temperature (HI)

Similar calibration parameters (but to brightness temperature in Kelvin instead of flux density in Jy) can be generated by observing Galactic HI standards.

`paf.py -s paf_250225_011220_ON.hdf5 -p 1 -F '1000,14000,17000,18000' -e 10 -A 0 -B 72 -g -n`	This writes a brightness calibration file using data containing observations of the S9 Galactic standard region. A first-order polynomial is fit using data in the channel ranges (1000, 14000) and (17000, 18000) for

[2] In this context, a drive error is when the telescope is not in-lock, so that the associated coordinates are incorrect.

	all 72 beams. A 10-channel smooth is applied.

In this case, it is recommended NOT to use an off-source reference region, since this will subtract out some of the Galactic HI signal. cryoPAF spectral baselines are flat enough for this purpose.

Validation

A simple way to validate calibration data is to generate system temperature plots using `paf-tsys`.

`paf-tsys.py *.json`	This will generate a number of system temperature plots from all calibration files in the current directory. If the path contains both flux density and brightness temperature calibration data (at similar frequencies), efficiency curves are also plotted.

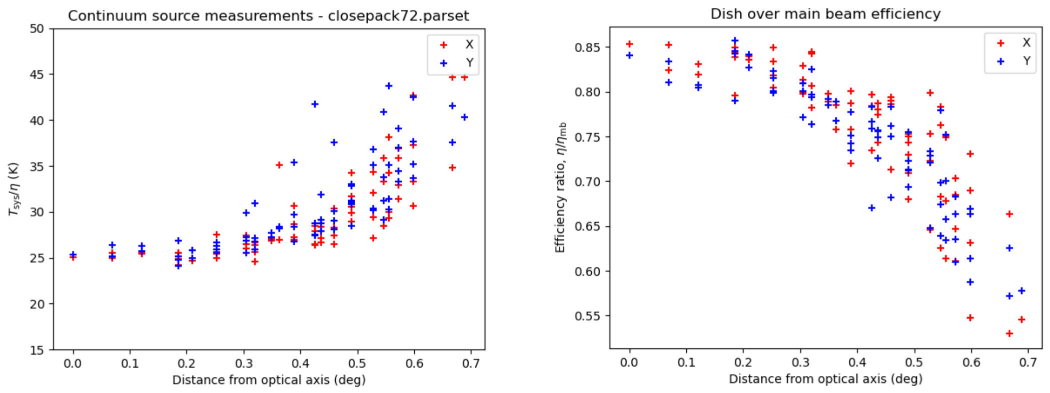


Figure 3: example plots of (left) the ratio of system temperature to dish efficiency as a function of off-axis radius; and (right) the ratio of dish efficiency to main beam efficiency as a function of off-axis radius for 72 beams and two polarisations at 1.4 GHz.

Spectral baseline removal

Point-and-shoot

A point-and shoot observation is a very convenient spectral-line observing mode with the cryoPAF because the focal plane is usually oversampled at most frequencies. If the files are fairly short, it is often possible to average the data using `paf.py` for each beam before the parallactic angle changes significantly (see Figure 1). Parallactification will extend the length of time over which off-axis beams can be averaged, but beware that spectral baselines are sensitive to the rotation angle of the receiver (focus position angle). An off-source 'reference' region of a similar integration time, and not containing emission in any of the beams, is normally applied to this type of observation. A single reference region can be used for a number of nearby (in time) point-and-shoot observations if the receiver rotation angle is similar.

`paf.py -s paf_250224_020145.hdf5 -r paf_250224_020935.hdf5 -h paf_250223_233303_ON_cal.json -a 0 -b 2 -A 0 -B 72 -n -o paf_250224_020145_N6744_qcal2IF.hdf5`	This calibrates and averages (in time) data for subbands SB01 and SB02 for all 72 beams and creates an output file. An off-source 'reference'

`-n`	spectrum is applied. Plots are suppressed.

Now that calibration has been applied, it is also useful to average XX and YY to Stokes I and apply any spectral baselines.

`paf.py`
`    -s paf_250224_020145_N6744_1_qcal2IF.hdf5`
`    -A 0 -B 72 -a 1 -F '1000,6500,10000,14000'`
`    -p 1 -i -n`
`    -o paf_250224_020145_N6744_19bcalI.hdf5`
	This averages XX and YY to Stokes I and fits a first-order spectral baseline data in the channel ranges (1000, 14000) and (17000, 18000) for all 72 beams. Plots are suppressed.

Be aware the `paf.py` always applies time averaging, even if the telescope position is changing.

Multiple point-and-shoot

A fast way of observing is to combine multiple pointings into a single HDF file. In this case, no time averaging is possible. However, similar to `paf-calsplit`, the routine `paf-split.py` will split such data into multiple position groupings. Each of these can then be time-averaged as above, which saves a lot of disk space and speeds up gridding. For example:

`paf-split.py -s paf_241118_100007.hdf5`
`             -a 1 -X 71`
	This averages XX and YY to Stokes I and fits a first-order spectral baseline data in the channel ranges (1000, 14000) and (17000, 18000) for all 72 beams. The optical axis beam (‘`-X`’), which determines when to create a new group, is 71. Plots are suppressed.

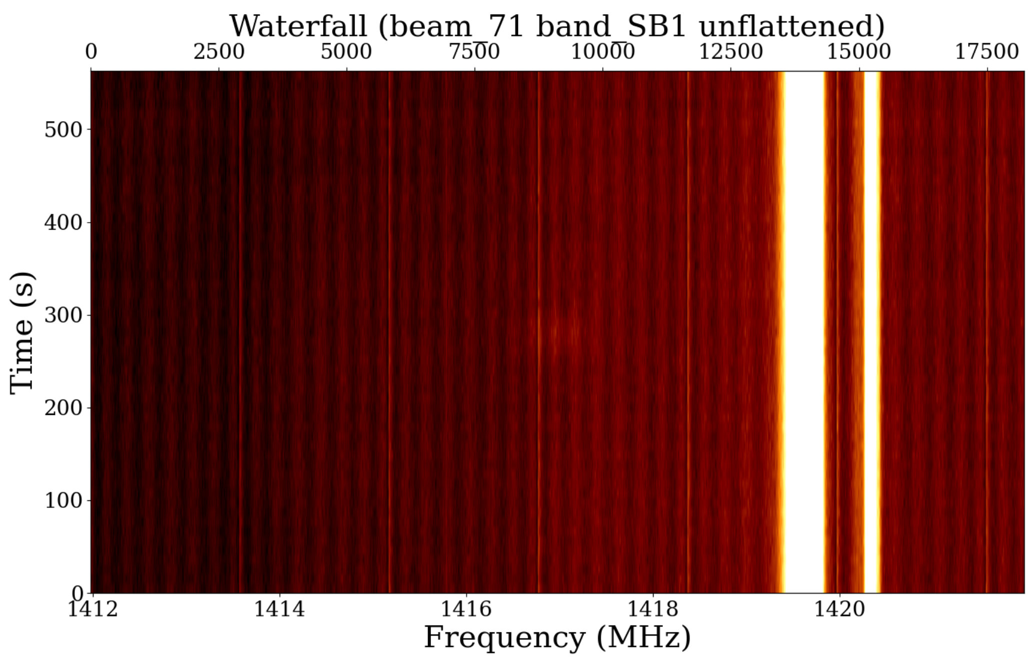
`paf.py -s @paf_241118_100007_G_in.txt`
`       -r paf_241118_050326_OFF.hdf5`
`       -o @paf_241118_100007_S9G_out.txt`
`       -h paf_241118_061543_ON_cal.json`
`       -p 1 -F '10000,15000,20000,25000'`
`       -n -a 1 -A 0 -B 72`
	This takes the `paf-split` files listed in the named text file, applies calibration and time-averaging. A single off-source pointing is used, and a spectral baseline is subtracted. The output is written to the filenames in the named text file.

Continuous scans (active or drift)

Active or passive (drift) scans are a convenient way of mapping large areas of sky. ‘`paf-scan`’ can reduce such data using a variety of algorithms, most of which are best suited to the detection of discrete sources such as external galaxies (i.e. compact in frequency and unresolved on the sky).

`paf-scan.py`
`     -s paf_250224_064528.hdf5 -A 71 -a 1`
`     -h paf_250223_233303_ON_cal.json -z`
	This applies a mean spectral and temporal flattening filter to beam 71, subband 1 and provides various diagnostic plots (mean spectrum, mean power v. time, waterfall, and flattening waterfall).

The above example uses a simple 2D mean filter (i.e. dividing by the mean spectrum and the frequency-average power). Median and SVD filters are also available.



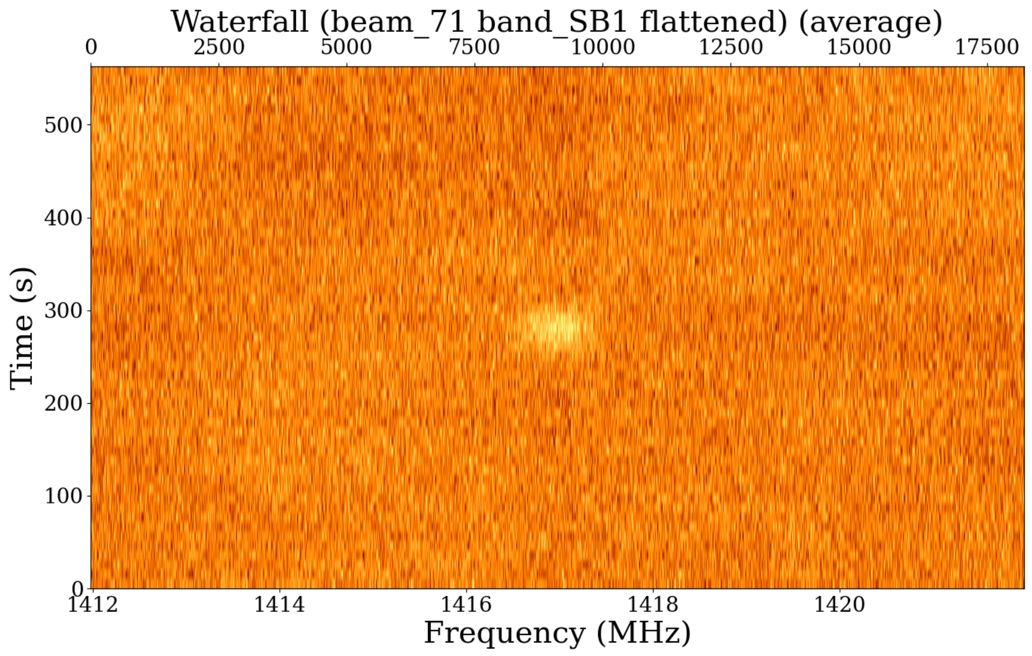


Figure 4: a comparison of (top) unflattened and (bottom) flattened data using the above command (YY is shown). A fake 2D Gaussian of 1% amplitude (relative to Tsys) was added to the raw data.

No time-averaging is applied, so the output files will be as large as the input files, unless frequency averaging, or subband selection is also enabled (which is recommended, otherwise gridding into cubes will be slow). `paf-scan` should not be used in cases where the telescope is not scanning or the emission is spatially extended, as there may be signal loss. Similarly, SVD with n>1 singular value removed may cause near-complete signal loss.

Imaging

Data from point-and-shoot observing or scanning can be gridded into a sky cube using '`paf-grid`'. The routine is functional but fairly basic, so you should consider whether other single-dish software suits your needs better (e.g.gridzilla). Although vectorized, it is not multithreaded nor parallelised, so may be slow and may also be limited by memory availability. After all, it is python.

`paf-grid.py -s 'paf_241118_100007_G*cal.hdf5' -o LMC.fits -c LMC.counts.fits -a 1 -A 0 -B 72 -G LMC_grid.txt -S -I`	This grids all wildcarded data files in the SB01 zoom window for all 72 beams. A weight image is also produced. The gridding parameters are described in the text file 'LMC_grid.txt'. '`-S`' and '`-I`' are convolution and interpolation switches, respectively.

The above example is the command used to grid 49 pointings towards the LMC. Since the basic gridding kernel is a top-hat, a Gaussian convolution is recommended ('`-S`' switch). If there are a few internal pixels with no data, it is often safe to interpolate ('`-S`' switch) using SciPy's '`CloughTocher2Dinterpolator`'.

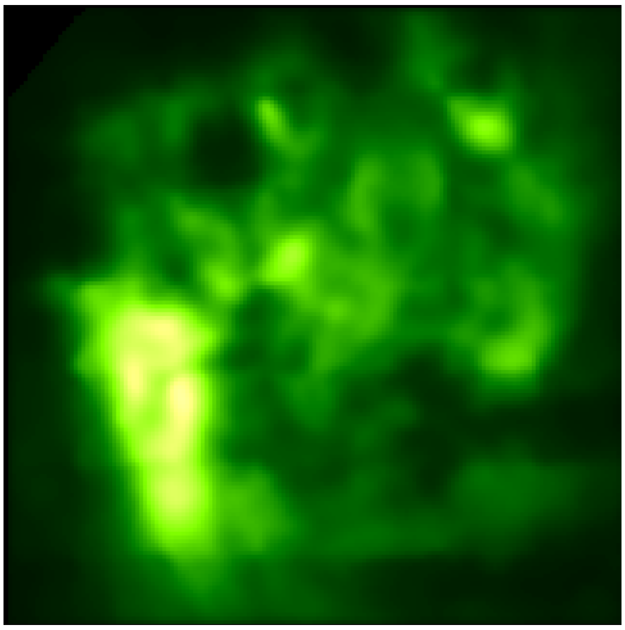


Figure 5: a moment map of the LMC cube produced by the above paf-grid example.

The gridding text file (LMC_grid.txt in the above example), contains the required grid parameters, which are self-explanatory:

```
# LMC all gridding parameters
coord = [80.0, -69.0, 1419.0]  # deg, deg, MHz
dim = [160, 160, 40] # LMC
dcoord = [-0.05, 0.05, 0.005]      # deg, deg, MHz
# Other (more constant parameters)
fwhm = 3.0  # FWHM convolving kernel in pixels (smooth=True)
rad  = 0.15 # radius of gridding kernel (deg)
```

RA/Dec SIN projection is currently enabled, but the grid is set up using astropy WCS, so the full range of coordinate systems and projections is potentially available with ease.

Visualisation

Data cubes from `paf-grid` are in standard FITS format, so may be conveniently viewed in kvis, CARTA, DS9, iDaVIE, etc.