Running the SPAM pipeline

This section (created with significant help from Pratik Dabhade) describes how to run the pipeline on continuum observations at 150, 235, 325 or 610 MHz, with 16 or 32 MHz of bandwidth, obtained with the GMRT software correlator backend (GSB).

Basic pipeline run

Let's assume you have requested and downloaded a single night observation from the GMRT archive in LTA format. You will end up with 3 files:

- <project name>_<observe date>.lta - The visibility data in LTA format (e.g., 29_043_24aug2015.lta)
- <project name>_<observe date>.lta##<observation number>.obslog - The operator log file in ascii format; contains useful information about how the observations went.
- <project name>_<observe date>.lta##<project_name>.FLAGS[.<index>] - The online flag information in ascii format; contains status information of individual antennas, used for flagging

We will assume that the observation contains at least one of the primary calibrators 3C48, 3C147 or 3C286.

First we convert LTA to UVFITS format:

```
lta_file_name = "./<project name>_<observe date>.lta"
uvfits_file_name = "./fits/<project name>_<observe date>.lta.UVFITS"
convert_lta_to_uvfits( lta_file_name, uvfits_file_name )
```

Then we derive calibration and flagging information from the primary calibrator(s), pick the best one, transfer those informations to all other source visibilities present in the observation, and export each of these pre-calibrated visibility data sets to UVFITS:

```
pre_calibrate_targets( uvfits_file_name, flags_file_name = lta_file_name +
"##*.FLAGS*" )
```

The fits subdirectory in our project directory now contains pre-calibrated visibility data sets per source. The file name convention is as follows: <source name>_<observation reference date>_<polarization(s)>_<sideband>.UVFITS

The main pipeline will take the pre-calibrated visibilities of any source you specify, and produce a series of images with increasingly more flagging and calibrations.

```
target_uvfits_file_name = "./fits/<source name>_<observation reference
date>_<polarization(s)>_<sideband>.UVFITS"
process_target( target_uvfits_file_name )
```

This may typically take 6-8 times the duration of the actual observation to complete. The good news is that you can walk away and check the results later. All processing output is captured in a log file:

./datfil/spam_<source name>_<start date>_start_time>.log The final image is: ./fits/<source name>.SP2B.PBCOR.FITS

There's a million options available to make this process work for different / more complicated data sets. Some of the most common situations are discussed below.

Changing image weights

The user can change some of the imaging parameters (AIPS IMAGR style), including the image weights. By default, the main pipeline uses no UV range cuts, and sets the Briggs robust parameter to -1 to compensate for the broad PSF wings due to the centrally condensed UV coverage. This can be changed as follows:

```
imagr_params = { 'robust' : 0., 'uvrang' : [ 0.5, 10. ] } # AIPS IMAGR parameters are passed as a Python dictionary
process_target( target_uvfits_file_name, imagr_params = imagr_params )
```

Main pipeline intermediate files

The main pipeline takes the data through various repetitions of calibration, flagging, and imaging. If so desired, one can keep all the processed fits and uvfits of the target you are processing. This way, for instance, you can monitor the progress in image quality in steps. This is activated by disabling the minimize_storage option:

```
process_target( target_uvfits_file_name, minimize_storage = False )
```

Dual-frequency observations

In case of dual-frequency observations, the 235 MHz visibilities are located in the LL polarization and 610 MHz visibilities in the RR polarization. To process one or both, they can be split in the following way:

```
convert_lta_to_uvfits( lta_file_name, uvfits_file_name_235, stokes_list = [ "LL" ]  )
convert_lta_to_uvfits( lta_file_name, uvfits_file_name_610, stokes_list = [ "RR" ]  )
```

The two output UVFITS files can then be processed further as usual. Please note that you may have to manually limit the frequency channel range for the 235 MHz observations, as in most cases the correlated bandwidth is 32 MHz while the 235 MHz receiver bandwidth is limited to 16 MHz. Please see below.

Limited bandwidth observations

In some cases, especially for 150 and 235 MHz observations, a bandpass filter is activated during observations to suppress RFI near the observing band. This means that parts of the correlated bandwidth (often 16.7 or 33.3 MHz) are not usable. A preferred frequency range can be manually selected by running

```
pre_calibrate_targets( uvfits_file_name )
```

and inspecting the resulting bandpass plots (postscript format) in the ./prtfil subdirectory via the shell command

```
gv ./prtfil/*_BANDPASS.PS
```

Select a channel range over which the bandpass phases are well-behaved (linear) for most antennas, then re-run

```
channel_range = [ 10, 225 ] # example channel range to keep
pre_calibrate_targets( uvfits_file_name, channel_range = channel_range )
```

Using different calibration models

By default, a point source model based on the NVSS, WENSS, VLSSr, SUMSS, and MGPS-2 catalogs is used to bootstrap the phase calibration at the start of the main pipeline (*process_target()*), and is also used to correct the astrometry during later stages of the pipeline run after self-calibration and during peeling. This may not always give the desired result. It is possible to use other reference models. The simplest is to switch to a point source model based on TGSS.

```
process_target( target_uvfits_file_name, use_tgss = True )
```

In case of observations of the same field in multiple GMRT bands, it is often useful to start with processing the lowest frequency data (which means the widest field-of-view, but also lowest resolution), and use the resulting (primary beam corrected) image as a calibration model for the next lowest, etc. Easiest is to run PyBDSM/F on the *.SP2B.PBCOR.FITS image and save the extracted gaussian list in ASCII format.

```
catalog_name = "<project_dir>/fits/<field_name>.SP2B.PBCOR.pybdsm.gaul"
catalog = read_pybdsm_ascii_catalog( catalog_name )
source_list = create_source_list_from_catalog( catalog )
resolution = 15. # representative resolution of model image in arcsec
process_target( target_uvfits_file_name, model_source_list = source_list,
model_resolution = resolution )
```