ALBIUS: Oxford final report

Compiled by: Hans-Rainer Klöckner (June 2012)

Overview

The Oxford contribution to ALBiUS has been significantly delayed due to the availability of matching effort, which was related to the funding situation at the beginning of this project in the UK. Therefore the Oxford ALBIUS contribution started in October 2009. The person in charge of the scientific and technological aspects in this project was Steve Rawlings. Unfortunately Steve passed away early 2012 and it is not clear how to relate some of the aspects in this project to matching effort and how to associate some of the people's work to ALBIUS deliverables.

Since November 2011 nobody has been employed and Klöckner is the only person who worked a longer period in the ALBIUS program to oversee some parts of the project. It should be noted that the work (also beyond November 2011) that could be counted as ALBIUS deliverable may have not been support by this program, and possible financial contribution may need to be organised by the financing department of the institute.

It is therefore also not clear how to account the man-month of the individual people involved:

- Hans-Rainer Klöckner (Task 6.3.2) has been appointed for 100% (October to December 2009) and 50% (from January 2010 to October 31st 2011) [Oxford, MPIfR]

- Stef Salvini, Fred Dulwich, Benjamin Mort [OeRC]

- Tom Mauch [Oxford, Hertfordshire, SKA South Africa]

Oxford has been involved in the following work packages in the ALBIUS program: calibration algorithm, quality control, and data excision. This report tries to summarise the work, on a best-effort basis, of the individual people mentioned in this report.

2. Calibration Algorithms (Stef Salvini, Fred Dulwich, Benjamin Mort, Hans-Rainer Klöckner)

Overview

Here the initial work of testing a novel calibration algorithm for radio synthesis telescopes is described. This algorithm, developed outside of ALBIUS, aims to minimise the chi-square between observed and model visibilities and has been shown scale to LOFAR stations (~100 elements), LOFAR core (~ 300 elements), and SKA Phase 1 stations (~1,000 elements). The kernel of the algorithm is a new minimisation that replaces the Levenberg-Marquardt algorithm, or others, that have been traditionally used. Additionally, when few bright sources are used for calibration, the minimisation can be carried out over the space of the largest eigenvalues/eigenvectors, for increased performance and stability.

Each iteration of the algorithm requires O (n^2) operations. Two versions of the algorithm exist: the first version, employed within ALBIUS, treats polarisations independently from each other (no cross-talk). The second version can optimise against cross-talk between polarisations, the complex gains thus being represented by 2 x 2 matrices.

The algorithm, originally developed in MATLAB, was cast in C and incorporated in a module accessible from ParselTongue (PT). This was successfully used for the calibration of GMRT (see next section). The algorithm was also employed to calibrate preliminary results from the LOFAR UK LBA (see last section).

Although no data flagging is performed as part of the algorithm, the calibration procedure can be easily inserted into a tool-chain that incorporates flagging. Experiments have shown that the algorithm is resilient to partial cross-correlation (loss of antennas and/or baselines), and baseline restrictions.

Outside the scope of ALBIUS, the algorithm has been extended to allow for the calibration of fully-polarised data, including polarisation cross-talk, and this version was implemented experimentally within MeqTrees (S. Salvini, O. Smirnov, 2012).

The algorithm – a pilot study with GMRT data (Hans-Rainer Klöckner)

Here a description is given of a pilot study to test the novel calibration algorithm with "real data." Using GMRT data of the calibrator source 3C48 observed at 608 MHz, and the software tools generated in the GMRT pipeline (the pipeline is not an ALBIUS deliverable, for more information see section quality control) it was possible to build a test environment for the algorithm within ParselTongue (PT). The algorithm was incorporated in a python module accessible from PT. The output of the module itself passes the gains (real, imaginary part) per antenna, and the required input of the module is the complex-correlation matrix (CCM) of the data and the model sky. Here a description is given on how the necessary input can be generated in PT, and the results of the calibrated data are shown.

The calibration module makes use of CCM's for a single integration, and the main task of this work was to determine these input matrices. This data format differs from the random-groups UV fits files used by AIPS, where the visibilities are sorted by time. In order to determine a CCM, the dataset needs to be structured by baseline instead. The function ("datatobase") has been developed in PT and is distributed in the COMP_LIB.py library (see section "data excision – UVFITS to MS flagging"). In addition to the observed visibilities, a model sky visibility dataset needs to be generated. The following describes the steps needed to generate a model visibility set (continuum only e.g. one channel):

copy the original data with the task UVCOP

edit the real and imaginary part of the visibilities to zero

<u>PT example code:</u>

```
visdata = AIPSDataVis(tdata[0],tdata[1],tdata[2],tdata[3])
stokes  = ['RR','LL']
for visibility in visdata:
     for st in range(len(stokes)):
               visibility.visibility[IF,:,st,0] = [0]
               visibility.visibility[IF,:,st,1] = [0]
               visibility.visibility[IF,:,st,2] = [1]
               visibility.update()
```

add sky model into the "zero" data with the task UVSUB, adding source by source iteratively

In addition, the final model CCM requires the autocorrelation values (the diagonal of the matrix), which, assuming a uniform telescope array, is the sum of the flux densities of all sources. This value is placed in each entry of the diagonal; errors in this estimate will result in errors in absolute amplitude.

Based on these functions, it is possible to pass the CCM and the model CCM per time step (in this case, 16 second integration time) to the calibration module, and receive the antenna-based corrections for the real and the imaginary part of the visibilities. These corrections can be placed into an AIPS SN table and the calibration can be applied via the standard tools in AIPS.

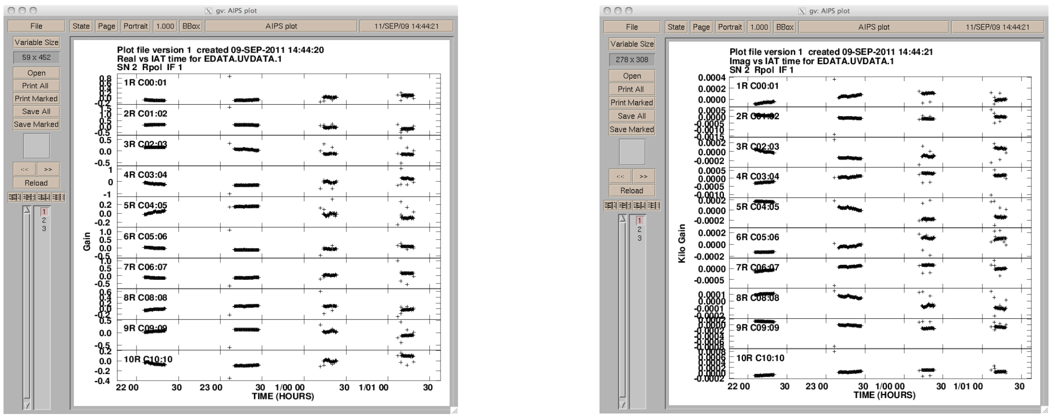
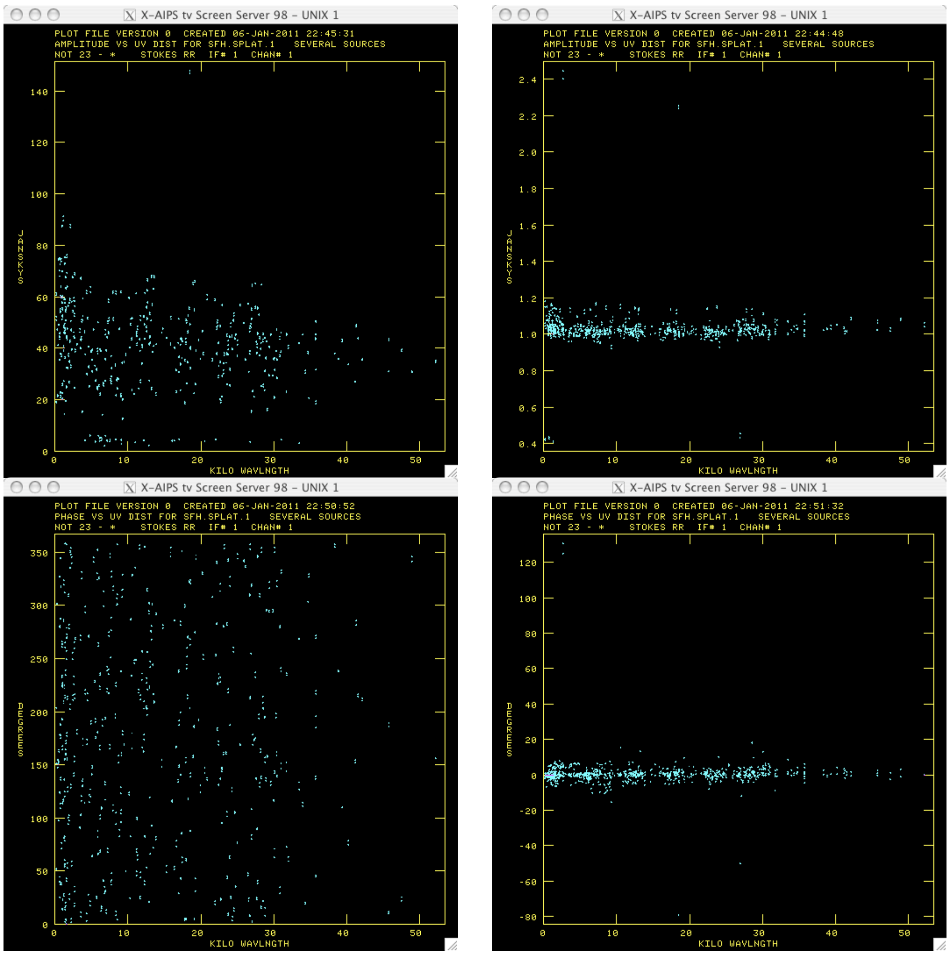


Figure 1: Calibration corrections versus time for antenna 1 to 10. Correction of the real (left) and the imaginary (right) values of the visibilities.

The correction of the visibilities per time and antenna are shown in Figure 1 and show a smooth variation with time. The outliers in the correction values could be used to flag bad UV data at these times: although this has not yet been investigated, it would be worthwhile to try.



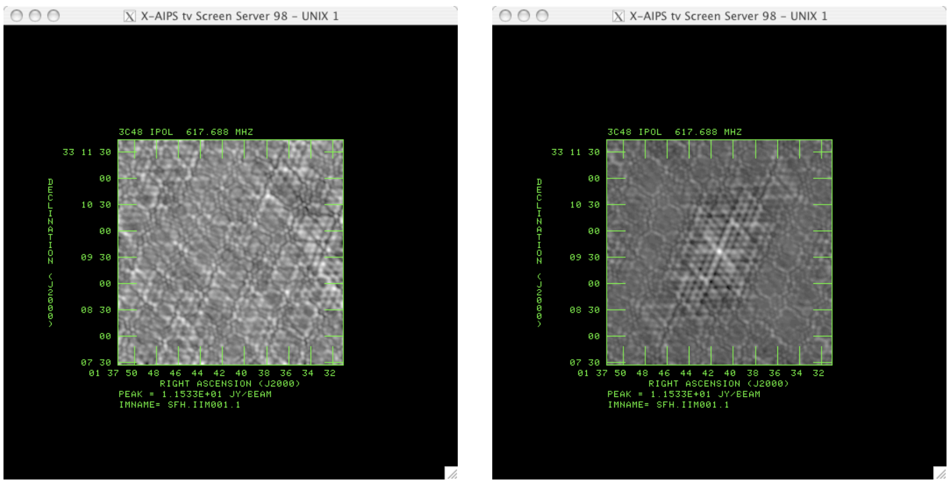


Figure 2: Un-calibrated data (left) versus calibrated data (right). Top to bottom is shown amplitude versus UV distance, the phase versus UV distance, and the dirty image. Note that the absolute amplitude of the source is set to 11 Jy.

Figure 2 shows the effect of the calibration estimates applied to “real” data. This test showed that it is possible to calibrate “real” datasets with the new algorithm. For an unresolved point source in the phase centre, the calibrated amplitudes and phases are constant, as expected. This pilot study provided a first glimpse on the capabilities of the novel calibration algorithm and triggered further improvement of the algorithm itself in many ways. For example, the first version of the GMRT test environment could only handle one polarisation and un-flagged datasets.

Test using single-station LOFAR data

The algorithm has been tested extensively by successfully calibrating complex antenna gains during a 12-hour all-sky observation, which was performed by cross-correlating antennas in the low-band array (LBA) of the LOFAR station at Chilbolton, UK. The observations were made by Griffin Foster (University of Oxford) in May 2010, and all 96 dual-polarisation dipoles in the LBA were included in the cross-correlation. Data were taken sequentially across 512 frequency channels, covering a total of 100 MHz of bandwidth, with all of the 195 kHz-wide channels consisting of separate one-second integrations on the sky. The complete channel range was scanned repeatedly, approximately every 520 seconds, throughout the observation period.

The results shown here concern only channel 300, at 58.4 MHz, although data from other channels were also calibrated as part of the test.

The calibration was performed using sky models of varying complexity. The simplest model used only two point sources, corresponding to the positions and fluxes of Cassiopeia A and Cygnus A. More complex models used the brightest 500 or 5000 pixels from the Global Sky Model data (de Oliveira-Costa et al., 2008), which included emission from the Galactic centre and the Galactic plane as well (see Figure 3). All-sky images were produced using the OSKAR-2 GPU imager and plotted using MATLAB.

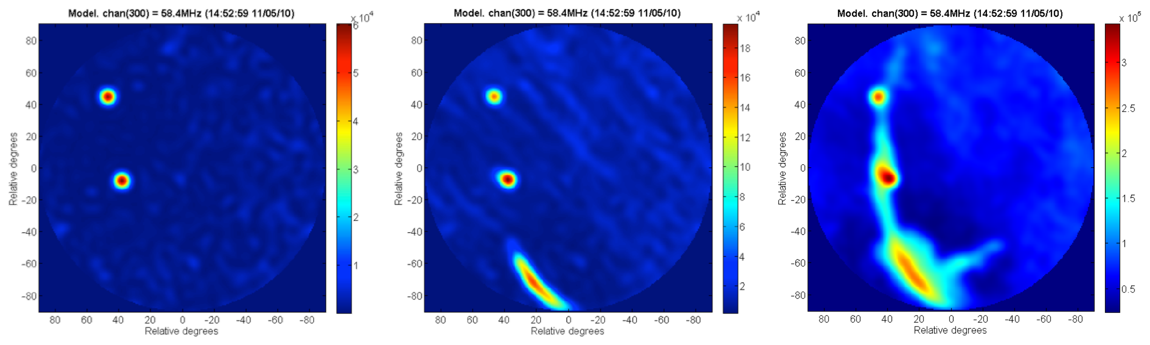


Figure 3: Images of the sky model with (left to right) 2, 500, and 5000 sources.

During the tests, it was found that the calibration performed using the simplest sky model (containing only Cas A and Cyg A) was not successful, because the model did not represent the data sufficiently well with the antenna model omitted. However, the attempts that were performed using models that included the Galactic centre were very successful at calibrating the data and producing a sensible all-sky image, as shown in Figure 4. These results utilised a sky model with the 500 brightest pixels, as there was little or no benefit from using more. The calibration process, which included generating model visibilities, took approximately 0.1 seconds in MATLAB using an Intel Core 2 E6750 running at 2.67 GHz.

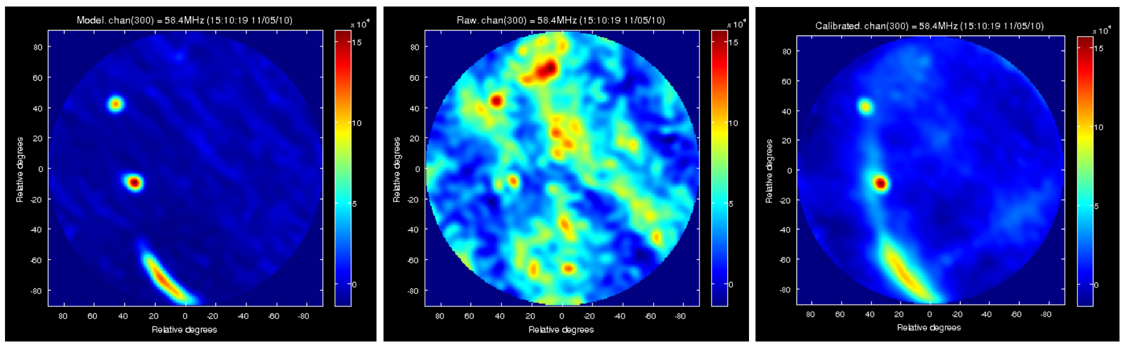


Figure 4: Calibration results using a sky model with 500 sources at 58.4MHz.

The gain solutions plotted over the 12 hour period are shown in Figure 5. These show that most of the amplitude gains were stable in time, with the exception of antenna 42: flagging this antenna may have improved the results further.

The cross-correlated visibility data from each integration were calibrated independently in this way, and the calibrated results from the whole observation are shown in an animation which is available separately.

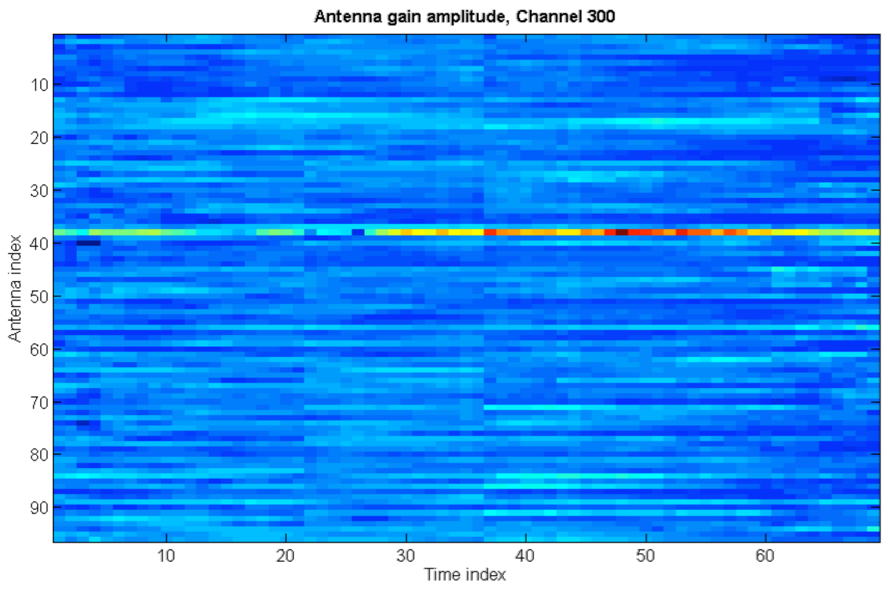


Figure 5: Antenna gain solutions as a function of time. The apparent discontinuities seen at time index 22 and 36 correspond to a larger than average time increment at those indices.

3. Quality Control (Hans-Rainer Klöckner, Tom Mauch)

Generally all work within the “Quality Control” work package was addressed by using GMRT datasets that were calibrated within an automated calibration pipeline. The GMRT pipeline is not part of the ALBIUS contribution, but it will be used as a “work horse” to investigate the usefulness, within a calibration workflow, of the data diagnostics. The pipeline itself is based on AIPS, ParselTongue (PT, developed in ALBUS and also ALBIUS) and the python software packages like scipy or numpy, which are all publicly available. The individual diagnostic plots are based on the python ploting library matplotlib (http://matplotlib.sourceforge.net/). Modules, which have been developed to access raw visibilities, can be used within python. This approach assures interoperability of the developed modules, and opens up the opportunity to develop modules that can be used by other calibration software packages (e.g. CASA).

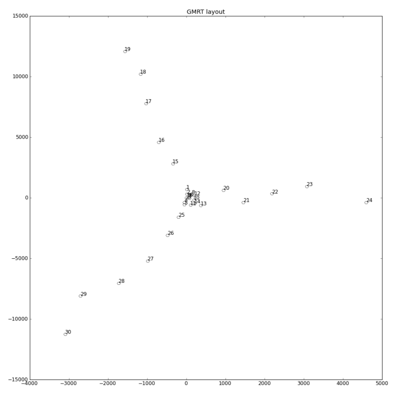
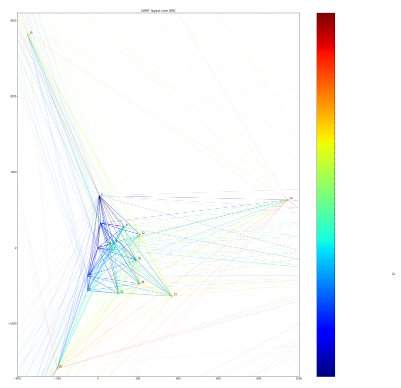
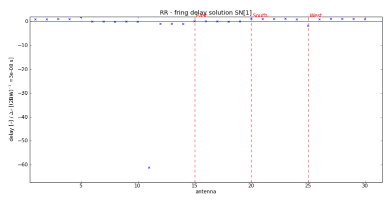
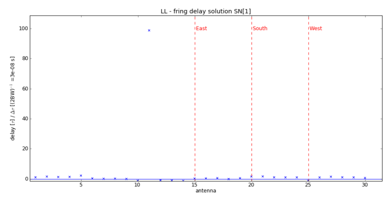
3.1.2 Data inspection (Hans-Rainer Klöckner, Tom Mauch)

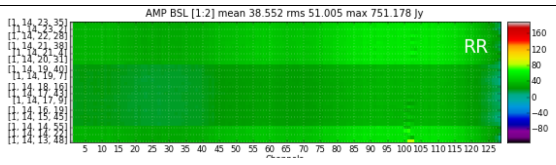
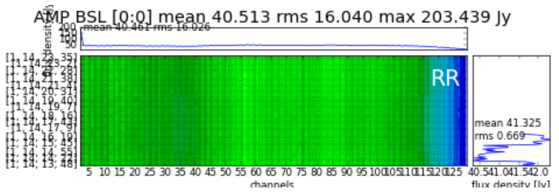
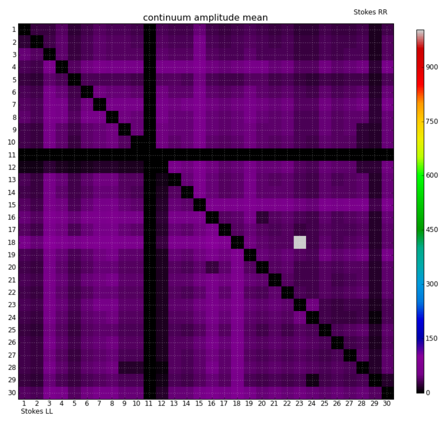
Data inspection is one of the most important steps in synthesis imaging processing, since only a few faulty visibilities have great influence in the image quality. Therefore, quality assessment of the visibilities at all stages in the calibration is crucial. Furthermore, the standard calibration is antenna based, but errors in the visibilities occur mostly on individual baselines (e.g. radio interference, and not matching sub-bands). In order to investigate the quality of the data it is essential to evaluate the data on individual baselines, and also the antenna-related calibration solutions.

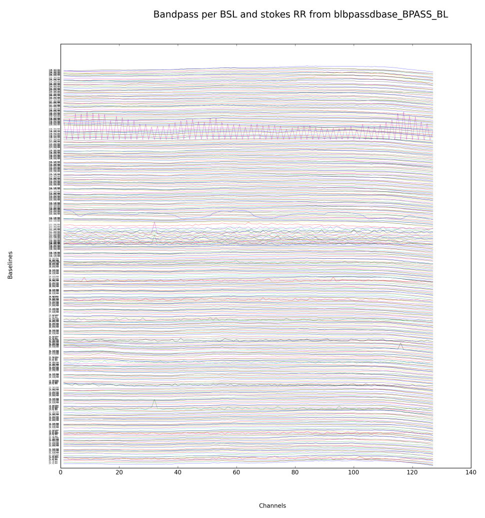
Several plots have been developed and tested of their practicability. In order to prepare the visibilities to be plotted, data handling modules have been developed and are available (PLOT_TOOLS.py). None of the new plot types are available in the classical AIPS software package. It should also be noted that some of the available plots in classical AIPS are exported as PostScript files, therefore plots of the UV visibilities will grow substantially in size. Those UV visibility plots that have been proven useful within the calibration pipeline have been translated into a python-based module.

The deliverables of this task are the description of the plots developed and the python modules to produce these plots (including a test dataset).

Metadata and visibility inspection plots (Hans-Rainer Klöckner)

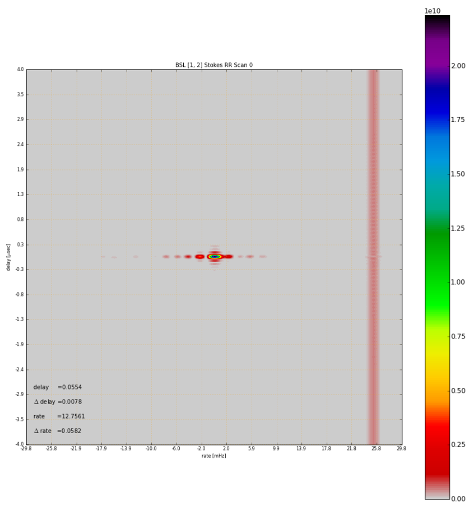
metadata inspection plots	individual procedures
	antenna distribution in X-Y coordinates function: antennapos (requires AN table)
	antenna distribution in X-Y coordinates, baselines color-coded by average amplitude function: colspidergram (requires AN table)
 	delay solutions versus telescope function: delayplot (requires SN table, which is generated by the function detdelay)

visibilities quality control	description and individual procedures
	dynamic spectrum of single baseline, time versus channels (amplitude color coded) function: plotspec by default the function will generate amplitude, phase, and weight plots
	dynamic spectrum of single baseline, time versus channels (amplitude color coded) and the average spectra: amplitude versus channel and amplitude versus time function: makeavgspecplotavg by default the function will generate amplitude, phase, and weight plots
	cross correlated “mean” amplitudes (color coded) per baseline and polarisation matrix. The upper triangle shows the estimates for polarisation “RR”, whereas the lower triangle shows polarisation “LL” function: prtbslprop optional estimates are: max, mean, median, min, max, number of visibilities, or standard derivation (std) of the integrated visibilities



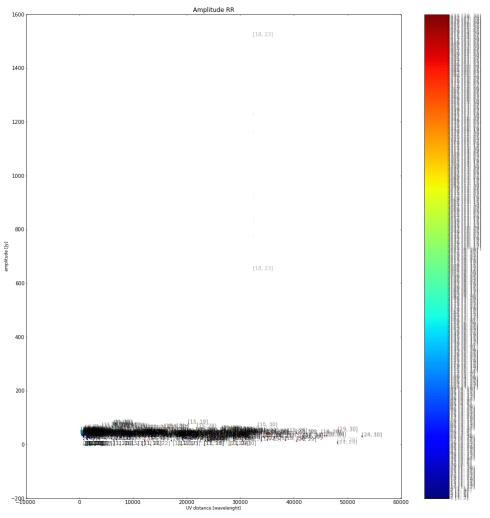
averaged spectrum per color coded baseline per single polarisation

function: blbpplot (requires input from blbpassdbase)



delay versus rate plot for single baseline

function: delayrateplt (requires input from delayrate)

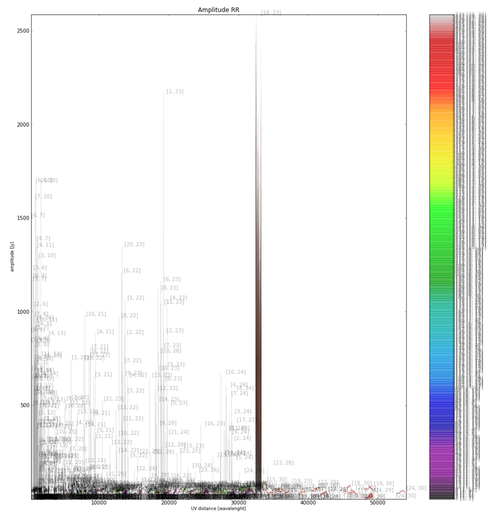


amplitude versus UV-distance plot for each visibility, color coded and labelled per baselines

this function exist in AIPS in task UVPLT without color coding and labelling, but it turned out that the color coding is the key to trace bad visibilities. Further it turned out that plotting each visibility is un-practical, like the AIPS version, and a new algorithm using polygons is implemented in the function colprtuvpolygon

function: colprtuv

optional: plot the phase, or imaginary versus real visibility data

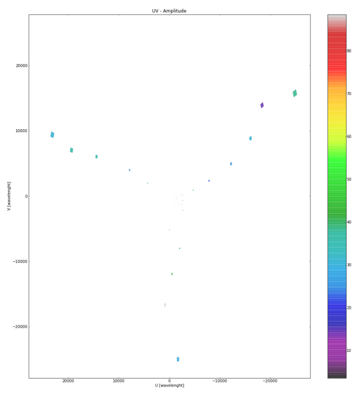


amplitude versus UV-distance plot using polygons to outline the visibilities, color coded and labelled per baselines

this function exist in AIPS in task UVPLT without color coding and labelling, but it turned out that the color coding is the key to trace bad visibilities

function: colprtuvpolygon

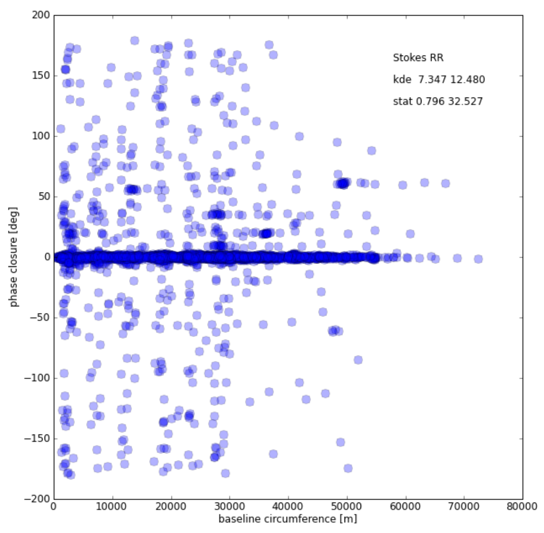
optional: plot the phase



UV – plane with color coded "mean" amplitude

function: prtuv

this function has been proofed to be not useful to spot bad data and further developments had been stopped

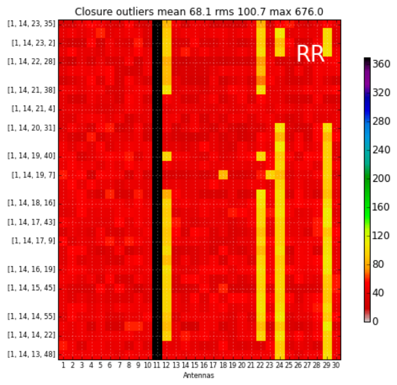


closure phase relation versus UV-distance - circumference of closure baselines

this function has been proofed to be not useful to spot bad data and further developments had been stopped

function: closurept

optional: amplitude closure



time versus telescope with color coded closure outliers

this function generates a histogram on the telescopes involved that produces 3 sigma outliers in the closure relation (e.g. 3 sigma around 0 for phase closure, see related plot closurept)

this function has been proofed to be not useful to spot bad data and further developments has been stopped

function: closureplt

optional: amplitude closure

The software suite and the example plots are publically available via the ALBIUS WIKI. The tar archive file PLOTLIB.tgz includes the following files:

AIPSLite.py	Module to assist in running AIPS/ParselTongue on machines without an AIPS installation. Developed by Stephen Bourke, JIVE
AIPSLiteTask.py	Module to assist in running AIPS/ParselTongue on machines without an AIPS installation. Developed by Stephen Bourke, JIVE
HRKinitAIPS.py	Module to facilitate the AIPS environment.
PLOTLIB.py	Module includes all function needed for the main script.
PLOT_TOOLS.py	Module to produce meta information on the dataset (e.g. statistics)
PLOT_TESTER.py	Main script to produce individual plots.
Example_PLOTS	Directory including the example output.

Image quality plots (Tom Mauch)

Image quality assessment of synthesis images is difficult to address, because various effects in the calibration of raw visibilities and the deconvolution in the image processing can produce significant artefacts. The first steps in the quality assessment are to determine the background RMS image. Based on this image the global and local dynamic estimates can provide a first indication on the image quality. Further image processing, such as building a source catalogue, would provide a second stage on quality assessment (e.g. comparing the flux density and the positions in the sky of the individual sources with their known flux density and position in the NVSS, WENSS, or SUMSS catalogue).

An important aspect of source finding in radio images is the determination of the background RMS noise in the image so that real source detections can be determined above this noise. Images often have a non-uniform background, which is the result of many different factors including the correction for the primary beam in individual pointings and artefacts from residual sidelobes which can appear around bright sources due to residual calibration errors during the imaging and cleaning process. Figure 1 below shows an example of noise peaks around a 400 mJy source in a 325 MHz image from the GMRT. In order to deal with the varying background in radio images, it is useful to construct a localised background map that shows the variation in background RMS at each pixel in the image, however this is not as straightforward as merely computing a local pixel RMS in a small box at each image pixel as the noise peaks around bright sources can be non-Gaussian.

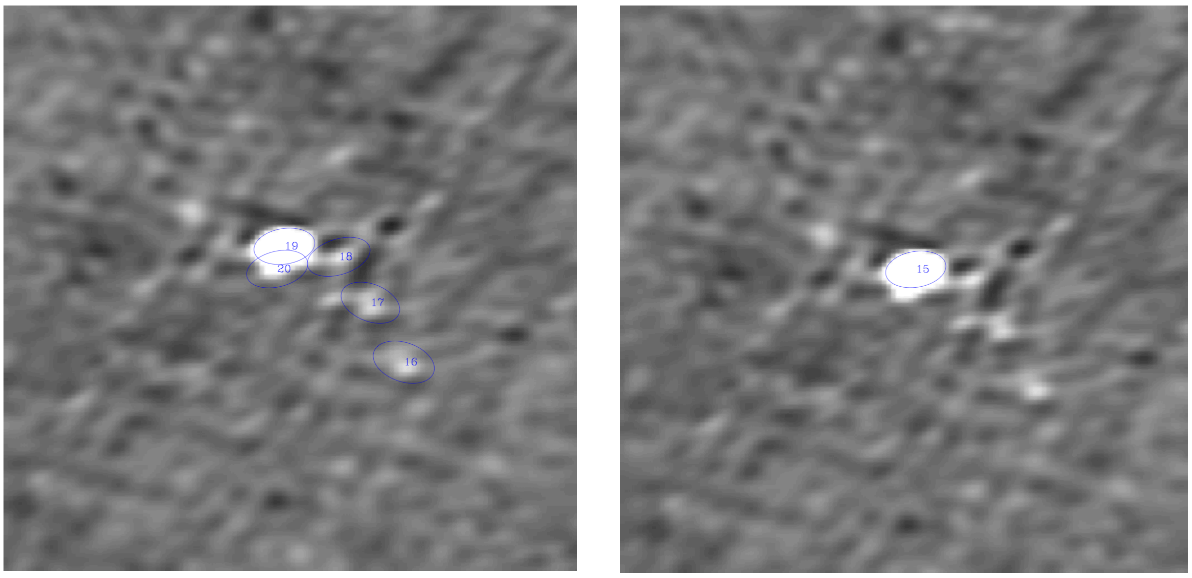


Figure 1: The left mage shows a 400 mJy source in a 325 MHz GMRT image with source detections of nearby artefacts using a constant background. The right image shows the same source with a local increase in the background added, here the nearby artefacts are not detected as sources.

We have developed an automated method to construct such background maps that has been used as part of a GMRT calibration and imaging pipeline. The method described is written in ParselTongue and makes use of numerous AIPS tasks that are described in more detail in the AIPS manuals (and beyond the scope of this report).

The procedure first determines a global background Gaussian RMS noise (σ_{global}) in the input image. This input image initially has any primary beam correction removed to ensure that the input image is uniform (ie. the noise doesn't increase towards the edges).

The AIPS task IMEAN is then used to fit the noise part of the pixel histogram in the central 50% of the (non-primary-beam corrected) image. In order to mimimise any contribution from source pixels to the calculation of this global image RMS, IMEAN is run iteratively using the fitted mean and RMS measured from the previous iteration until the measured noise mean changes by less than 1% to ensure that only noise pixels from the Gaussian background contribute to σ_{global}.

The input image is then divided by the measured global background and bright point sources brighter than a given threshold (S_{bright}) are found using SAD in AIPS. The threshold (S_{bright}) is selected such that all sources brighter than S_{bright} will potentially have noticable artifacts around them above the background noise and is related to the localised dynamic range of the observation (S_{bright}>DR x σ_{global}; S_{bright} =100σ_{global} was found to work well for 325 MHz GMRT images).

Noise peaks close to sources brighter than S_{bright} will be measured as real detections if not properly accounted for in the background and will lead to an increase in the measured local source density close to these bright sources (see Figure 1). Therefore, to determine the radius ($R_{artefacts}$) around each bright source that has increased noise and artefacts, the source density of 3σ_{global} sources as a function of radius from the bright source position is determined. The radius at which the local source density is equal to the background source density of all 3σ_{global} sources in the image is then taken as the radius of increased noise around bright sources ($R_{artefacts}$).

The spurious artefacts around each bright source can be removed by artificially increasing the noise in the background map, however care must be taken to not increase the background noise by too large an amount as genuine sources close to background sources should be retained.

To model the increased noise around bright sources a *local* dynamic range in the image is determined by measuring the ratio of the flux density of each 100σ_{global} bright source to the brightest 3σ_{global} source within the radius $R_{artefacts}$. The median value of the local dynamic range for all S>S_{bright} sources in the image is used as the *local* dynamic range, this median *local* dynamic range determination prevents moderately bright genuine sources close to each bright source from being rejected which would happen if *all* sources within the computed radius close to bright sources were to be rejected.

Starting from the input image a background image is created in the following way:

- A local RMS (σ_{local}) is determined at each pixel of the input image using the AIPS task RMSD. This calculates the RMS of pixels in a box of 5 times the major axis width of the restoring beam and is computed at each pixel in the input image. RMSD iterates its RMS determination 30 times and the computed histogram is clipped at 3σ on each iteration to remove the contribution of source data to the local RMS determination.

- The local RMS map then has a Gaussian added at the position of each $S>S_{bright}$ source with width determined from the radius of the local increased source density $R_{artefacts}$ and peak determined from the median local dynamic range described above.

Figure 2 shows a typical GMRT image and the background map constructed using this algorithm.

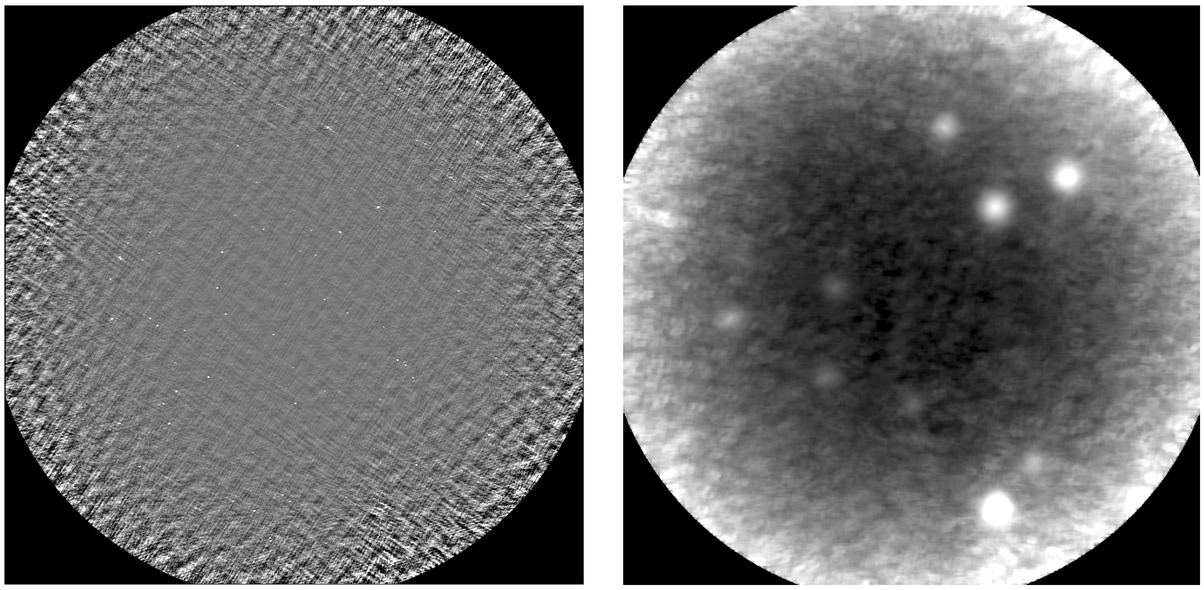


Figure 2: A typical 325 MHz GMRT image with bright sources and a background map constructed from it. Note the bright peaks in the background map which are at the location of bright sources in the original GMRT image.

Another important aspect of image quality control and source detection is to check that the flux densities and positions in the final image are accurate. Any systematic deviation of the positions or fluxes from those expected can indicate errors made during calibration, which need to be investigated further. A part of our automated backgrounding and cataloguing pipeline compares the measured positions and flux densities of sources in the image with known values from well established sky surveys (eg. NVSS & FIRST) as a quick quality check of images produced from for example an imaging pipeline. Figure 3 shows an example of this comparison (from the image shown in Figure 2).

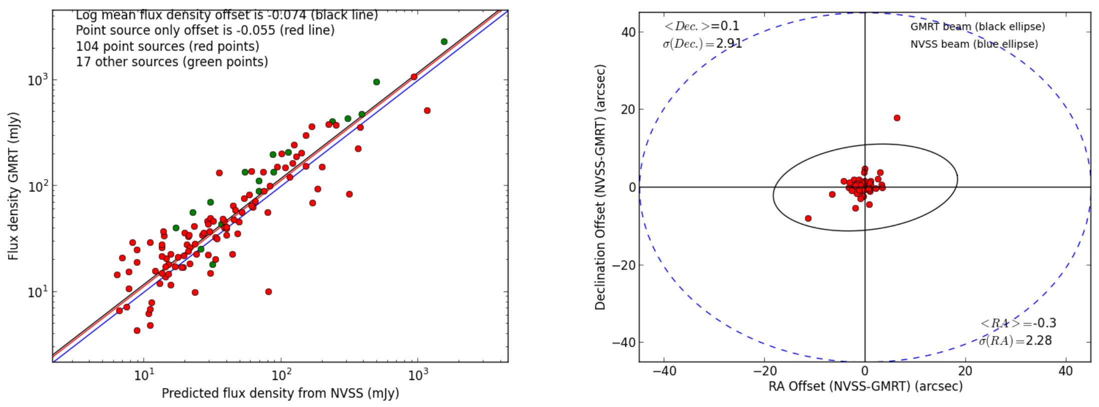


Figure 3: A comparison of the measured flux densities (left) and positions (right) from the GMRT image shown in Figure 2, with the NVSS positions and flux densities. 1.4 GHz NVSS flux densities have been scaled to the 325 MHz flux densities using a spectral index of -0.7 (S=ν^{α}).

3.1.3 Data Excision – UVFITS to MS flagging (Hans-Rainer Klöckner)

In order to understand the requirements needed to pass/mirror AIPS flag tables (FG) into MS compatible flag tables in CASA, the limitations of flagging UV-datasets in AIPS has been investigated. Within AIPS (and therefore ParselTongue, PT), some tasks are not capable of working with flag tables (FG) that exceed a predefined limit of flags at a given time. This limitation is a variable parameter in the AIPS software and can be changed by hand. However, there is always a risk of reaching this limit, especially if one observes in spectral line mode, and therefore a solution to this problem is needed that can be used on any AIPS installation. A working solution is described in the following in order to bypass the flagging limitation within AIPS. As a first step, the individual FG tables need to be split into tables that are smaller with respect to the "FG limits" using the AIPS task TBOUT. This task copies the FG into a new one, and the maximum number of flags can be defined by setting the parameter bcount and ecount. These "sub" FG tables can be applied to the UV data by using the AIPS task UVCOP iteratively. In general, the flagging by the task UVCOP is applied by changing the weights of the individual visibilities. The individual python and PT routines are available on request. This procedure is very IO heavy, and so might be impractical for large datasets. Similar bottlenecks can occur if the weights of the visibilities are changed via ParselTongue.

Depending on the parameter setting in the AIPS task UVCOP, the procedure described above could result in destructive flagging of the UV dataset, and therefore may make a one-to-one mirroring system impossible (e.g. missing out specific time ranges or entire baselines on the final dataset). Such destructive flagging may also be the case if one extracts the calibrated target source from the multi-source UV dataset using the AIPS tasks SPLIT or SPLAT.

Discussions with Stephen Bourke (JIVE), who was investigating a mechanism to implement a data mirroring system (AIPS/CASA) into ParselTongue, made clear that a full account of the applied flagging of the dataset can only be determined by comparing the visibilities one by one of the original and the "flagged" dataset. A software suite, within the PT environment, has been developed following this approach. The output of the software is a CASA compatible ASCII flag file that can be loaded by the CASA task FLAGCMD. The software suit and example files will be publically available via the ALBIUS WIKI (FGTRANS.tgz). A full description on how to work with the software is explained in the main script "UVFITSFG_2_FLAGCMD_TXT.py". Optionally the software can also produce an ASCII flag file that can be used by the AIPS task UVFLG (change the parameters: doflagcmdout = False and douvflagout = True).

Example of the flagging commands: first line of the flag file "3C147_FLAGCMD.FG.FGCMD" (CASA compatible) of the example directory "ExampleOUTPUT":

```
timerange='2007/3/2/14:13:40.0~2007/3/2/14:13:57.0'  antenna='1&2'  spw='0:96~96'  correlation='LL'
```

Same flag command (AIPS compatible; not presented in "ExampleOUTPUT"):

```
TIMERANG=1,14,13,40.0,1,14,13,57.0  ANTENNAS=1  BASELINE=2  BCHAN=96  ECHAN=96  BIF=1  EIF=1  STOKES='LL'  /
```

The following commands are needed to apply the flagging to the original UV dataset in CASA:

load UV dataset into CASA as MS set

task IMPORTUVFITS
fitsfile = ‘3C147.UV.FITS’
vis=’FGTEST.ms’
go

apply flags to the MS set

task FLAGCMD
vis=’FGTEST.ms’
flagmode=’file’
flagfile=’ 3C147_FLAGCMD.FG.FGCMD’
go

The software suite include following files:

AIPSLite.py	Module to assist in running AIPS/ParselTongue on machines without an AIPS installation. Developed by Stephen Bourke, JIVE
AIPSLiteTask.py	Module to assist in running AIPS/ParselTongue on machines without an AIPS installation. Developed by Stephen Bourke, JIVE
HRKinitAIPS.py	Module to facilitate the AIPS environment.
COMP_LIB.py	Module includes all function needed for the main script.
UVFITSFG_2_FLAGCMD_TXT.py	Main python script to compare visibilities.
README	Example input to the main script to produce files stored in the ExampleOUTPUT directory.
ExampleOUTPUT	Directory including the example output.

The following UV datasets are included in the FGTRANS.tgz tar archive: 3C147.UV.FITS, 3C147.FG.UV.FITS.

This software is already in use by colleagues in South Africa.

3.1.3 Data Excision – RFI mitigation

Data excision is the most time consuming part in the calibration of datasets obtained by radio telescopes, and there is a great desire to find an automated procedure to flag radio frequency interference (RFI). Unfortunately RFI affects the data in many ways and there is no uniform solution to this problem. Several software packages that flag radio data are publically available (by far the most successful package is the AOFlagger), but most of them do not offer great flexibility in order to eliminate the RFI in full. Here a software suite has been developed to bridge this gap, allowing the application of various RFI mitigation techniques and to evaluate which technique is most appropriate for the dataset in question. The individual RFI mitigation tools are written as individual sections in python and PT, to allow the interchange of modules between software packages like AIPS/CASA. Furthermore, this approach makes it possible to include new modules into the function to test novel RFI mitigation algorithms.

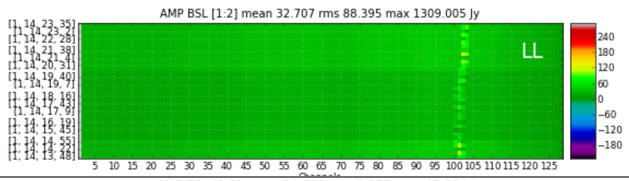
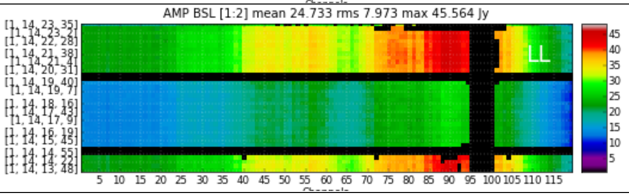
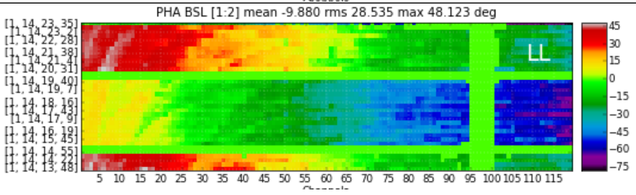
Here is a small description on the various steps in data handling and RFI mitigation techniques of the main flagger function “fgspec.”

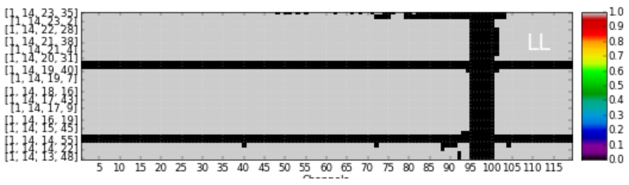
Work on raw visibility data
- flag the spectra on RMS without bandpass correction to eliminate strong outliers
- gravitational centre flagging in the real/imaginary plane

Determine a baseline-based band pass (either by smoothing the data or curve fitting)

- kernel convolution (Robets, Scharr, and Sobel operator) and dynamic spectrum RMS determination
- gradient filtering (Robets, Scharr, and Sobel operator)
- Gaussian convolution and dynamic spectrum RMS determination

The following figures show the results of a test run of the RFI mitigation software.

	raw dynamic spectrum of the amplitude of baseline 1-2 in stokes LL
	dynamic spectrum in amplitude after flagging applied
	dynamic spectrum in phase after flagging applied



dynamic spectrum in visibility weights after flagging applied. Note that the flagging in AIPS will alter the weights to >= 0.

The software suite and example files will be publically available via the ALBIUS WIKI (HRKFG.tgz). The tar archive includes the following files:

AIPSLite.py	Module to assist in running AIPS/ParselTongue on machines without an AIPS installation. Developed by Stephen Bourke, JIVE
AIPSLiteTask.py	Module to assist in running AIPS/ParselTongue on machines without an AIPS installation. Developed by Stephen Bourke, JIVE
HRKinitAIPS.py	Module to facilitate the AIPS environment.
FLAGGER_LIB.py	Module includes all function needed for the main script.
FLAGGER_PARA.py	file defines individual input parameter of SPEC_FG_TESTER.py
SPEC_FG_TESTER.py	main script to run the RFI mitigation
Example	example directory

The following UV datasets are also included in the tar archive: 3C147.UV.FITS, 3C147.FG.UV.FITS.

The RFI mitigation software has been tested on GMRT data and first tests have done using eMERLIN datasets. In collaboration with colleagues in Manchester, London, and Onsala, this software has provided input to the RFI mitigation efforts of several eMERLIN legacy projects.