Basic CABB Continuum Data Reduction with MIRIAD

Jamie Stevens (Jamie.Stevens@csiro.au)

September 17, 2012

1 Introduction

This tutorial will describe the regular method of calibrating, flagging and imaging for ATCA CABB data. Most of what we do here will be applicable to pre-CABB data as well.

The data we use here is from the ATCA experiment CX208. This experiment's aim was to measure the flux density of a source that appeared to have a very steep spectral index at low frequencies (below 1 GHz). The ATCA 16cm system (which covers the 1.1 – 3.1 GHz range) was used to get a measurement of the spectral index in this frequency range.

The experiment was performed on 28 April 2011. The array was in the 6A configuration at the time, because it was thought that confusion may have been a possibile explanation for the apparent spectral steepness.

Each section in this tutorial will deal with a stage of the processing. The outline of the process is as follows:

- Load the data.
- Perform initial calibration.
- Flag RFI.
- Calibrate.
- Apply calibration and prepare for imaging.
- Image and deconvolve.
- Measure.

1.1 Using MIRIAD

To start MIRIAD from the command line, you should normally only have to type '`miriad`' and press Enter. You should then get a terminal prompt: `miriad%`

To switch to a specific task and bring up the list of inputs it accepts, type '`inp task`' at the MIRIAD prompt. If you're within a task and want to see the current values for all the accepted inputs, you can simply type '`inp`'. To switch to a task and recall the settings that were used the last time the task was run, type '`tget task`'. After you have given the command to switch to a task, the command prompt will be replaced with the name of the task. For example, after typing '`inp uvplt`', the command prompt will become `uvplt%`.

To change a setting at the command line you simply type the name of the input followed by an `=` and then the value you want it to take. After you have typed the = sign, you can use TAB-completion to fill in the field if you want to use a filename in the current directory as the value. For example, to tell **uvplt** that you want to use Stokes I, you would give the command:

```
uvplt% stokes=i
```

You can change the value of an input without having to retype the entire set command by using the **er** command. For example, if `options=nofqav` was the current setting, and you wanted to add `nobase` to this list, you could type '`er options`' and this would change the current line to `options=nofqav` and place the cursor at the end of the line. You could then edit this line to produce `options=nofqav,nobase` and press Enter to set the value.

If you want to get some help about the current task, you can type '`help`' at the command prompt. This brings up the manual page for the current task, and you can use the UNIX **more** keys to navigate the documentation. You could also type '`help task`' to get the documentation for any specific task. Several keywords (like region and select) have their own documentation that can be viewed by typing their name instead of `task`, eg. '`help select`'. From outside the MIRIAD shell, you can view the documentation by typing '`mirhelp task`', or you can visit the webpage:

`http://www.atnf.csiro.au/computing/software/miriad/taskindex.html`

Once you have set the inputs as you desire for the current task, you can run the task by typing '`go`'.

In this tutorial, any MIRIAD tasks that we describe will be shown as they would be if you typed '`inp task`' after setting the appropriate inputs. For example, should we wish to make a plot of time vs amplitude in Stokes I for the ATCA flux calibrator 1934-638:

```
miriad% inp uvplt
  Task:   uvplt
  vis      = 1934-638.2100/
  line     =
  select   =
  stokes   = i
  axis     = time,amp
  xrange   =
  yrange   =
  average  =
  hann     =
  inc      =
  options  = nofqav
  subtitle =
  device   = /xs
  nxy      =
  size     =
  log      =
  comment  =
```

With this setup within the MIRIAD shell, the plot would be produced when you type '`go`'. You could also execute this command from the shell with the command:

```
   uvplt vis=1934-638.2100 stokes=i axis=time,amp options=nofqav device=/xs
```

2 Loading the data

The entire experiment is contained within a single RPFITS file `2011-04-28_1858.CX208`. We use the MIRIAD task **atlod** to load this data. We set the task up as shown:

```
miriad% inp atlod
  Task:   atlod
  in       = 2011-04-28_1858.CX208
  out      = cx208_2011-04-28.uv
  ifsel    = 1
  restfreq =
  options  = birdie,rfiflag,noauto,xycorr
  nfiles   =
  nscans   =
  nopcorr  =
  edge     =
```

Here, we choose to load the data from the RPFITS file into the MIRIAD dataset `cx208_2011-04-28.uv`. We only load the data from the first IF since when observing with the 16cm band, the central continuum frequency is set to be 2.1 GHz in each IF, and each IF is therefore just a copy of the other (recall that the usable 16cm frequency range is 1.1 – 3.1 GHz).

The `options` specified here make **atlod** flag a set of 14 channels that are always contaminated with self-generated RFI, along with some other regions of RFI that are known within this frequency band. We also tell **atlod** to discard the autocorrelation data, as we will not be using it, and to perform a phase correction between the X and Y products based on the online measurement of the XY phase. For low-frequency data, this set of options is recommended.

After the data is loaded, we need to flag the edge channels that are within the bandpass rolloff. The CABB filters have an area of about 32 MHz on each side that are less sensitive than the central part. To flag them easily, we use the **uvflag** command:

```
miriad% inp uvflag
  Task:   uvflag
  vis      = cx208_2011-04-28.uv/
  select   =
  line     =
  edge     = 40
  flagval  = flag
  options  =
  log      =
```

Using Miriad it is more convenient to deal with single-source datasets, whereas the set that comes from **atlod** contains all the sources that were observed. We use the **uvsplit** command to split the dataset into its component sources:

```
miriad% inp uvsplit
  Task:   uvsplit
  vis      = cx208_2011-04-28.uv
  select   =
  options  =
  maxwidth =
```

After executing this task, you will find the following directories in the working directory: `1934-638.2100`, `2051-377.2100`, `2058-425.2100`. Each of these directories is a single source, centred at a frequency of 2100 MHz; 1934-638 is the bandpass and flux calibrator, 2058-425 is the phase calibrator, and 2051-377 is the name we used for the field we want to image.

3 Initial calibration

We begin by using **uvspec** to get an idea of what the data looks like before calibration:

```
miriad% inp uvspec
  Task:   uvspec
  vis      = 1934-638.2100
  select   =
  line     =
  stokes   = xx,yy
  interval =
  hann     =
  offset   =
  options  =
  axis     = chan,amp
  yrange   =
  device   = /xs
  nxy      =
  log      =
```

The output should look something like Figure 1.

The initial calibration involves making a solution for the bandpass shape. The amplitude shown in Figure 1 as a function of channel is a combination of the flux of the source and the response of the receiving system to that flux. The bandpass solution should characterise how the system's response to the same amount of flux differs as a function of the frequency.

Using 1934-638 for the bandpass calibration is very convenient because its flux density is known as a function of frequency, and it does not change with time. It is possible then for Miriad to relatively simply determine the system response. We use the **mfcal** task to determine the bandpass shape:

```
miriad% inp mfcal
  Task:   mfcal
  vis      = 1934-638.2100/
  line     =
  stokes   =
  edge     =
  select   =
  flux     =
  refant   =
  minants  =
```

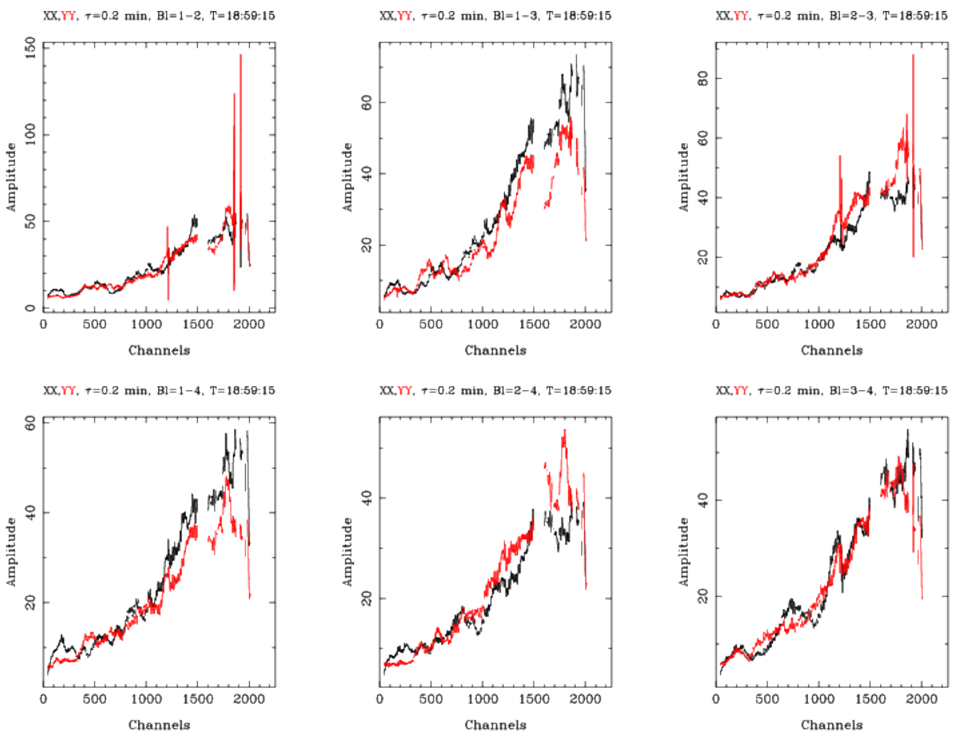

Figure 1: A plot of the spectrum from 1934-638 made before any calibration is performed.

```
  interval =
  options  =
  tol      =
```

This set of inputs will use `1934-638.2100` to determine both the bandpass shape and an initial complex gain solution as a function of time. The output should show that it is using the 1934-638 flux scale, and that it iterates towards a stable solution; it shouldn't take more than a couple of iterations to finish.

The output of **uvspec** should now look very different, as shown in Figure 2.

How the flux of 1934-638 changes with frequency is now readily apparent, as is quite a lot of RFI.

4 RFI flagging

We now try to flag this RFI in the easiest possible way. We begin by using the automatic flagging machinery in the task **pgflag**:

```
miriad% inp pgflag
  Task:   pgflag
  vis      = 1934-638.2100/
  line     =
  select   =
  stokes   = xx,yy
  device   = /xs
  device2  =
  mode     =
  flagpar  =
  command  = <b
  options  =
```

This tells **pgflag** to run the automatic flagger (`command=<b`) and flag both the `xx` and `yy` products based on the `yy` product.

After you run this command, **pgflag** will display a waterfall plot (channel as the x-axis, time as the y-axis, and amplitude as the colour of each pixel) in the PGPLOT device, and it will begin iterating over the baselines; its progress will be shown in the terminal, and will show how much of the data is flagged as it goes. After it has finished, you should change `stokes=yy,xx` and run it again, so it will flag based on the `xx` product as well.

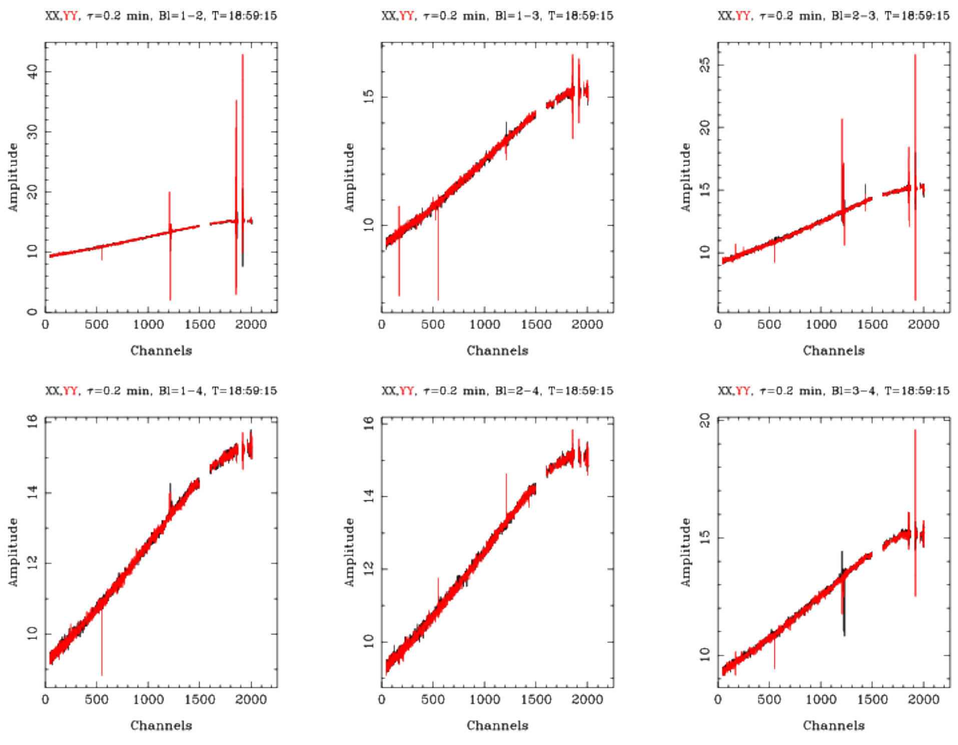

Figure 2: A plot of the spectrum from 1934-638 made after initial calibration.

We now run **uvspec** again to see how the data has changed; this is shown in Figure 3. You can see that it seems that all the major RFI appearing above the main flux is gone, although RFI below the main flux is still there.

To finish off the RFI flagging we are going to use the **blflag** task:

```
miriad% inp blflag
  Task:   blflag
  vis      = 1934-638.2100/
  line     =
  device   = /xs
  stokes   = xx,yy
  select   =
  axis     = chan,amp
  xrange   =
  yrange   =
  options  = nofqav,nobase
```

You will see a PGPLOT screen that looks like Figure 4. The erroneous visibilities are fairly obvious in this plot, and **blflag** allows us to quickly remove them. You could click on each bad visibility, but it is usually quicker to use the polygon selection facility. You should always keep in mind the philosophy that it is better to flag good data to ensure you flag all the bad data.

We will select a polygon that encompasses all the bad data below the main flux. To begin the polygon selection, press ‘`p`’ inside the PGPLOT window and then use the mouse to place corners of the polygon. Once the polygon has been formed, press the ‘`x`’ key to highlight all the points that will be flagged. Figure 5 shows a possible flagging polygon.

Pressing the ‘`r`’ key will redraw the screen without the flagged visibilities. You can continue to do flagging until you are happy with the remaining visibilities and then exit the task by clicking the right mouse button in the PGPLOT window.

You can then run **uvspec** again to see whether the spectrum is sufficiently RFI-free. Once you are happy with the flagging, it is time to redo the initial calibration with **mfcal**. Sometimes this may reveal that more RFI is present, and you will need to continue iterating between flagging and **mfcal** until you are happy with the result. It is not important that you get rid of all the RFI at this point, as only really strong RFI will corrupt the average flux.

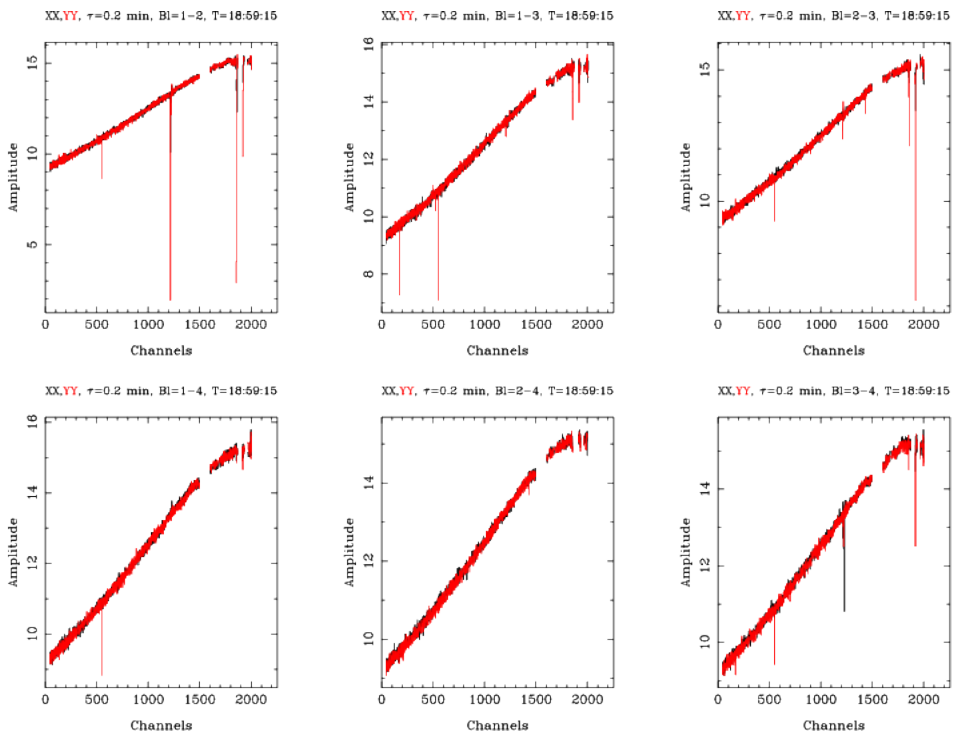

Figure 3: A plot of the spectrum from 1934-638 made after initial calibration and flagging using **pgflag**.

5 Gain calibration

Now we need to do a proper gain calibration over time, and to determine polarisation leakages. The MIRIAD task **gpcal** is used for this purpose:

```
miriad% inp gpcal
  Task:    gpcal
  vis       = 1934-638.2100/
  select    =
  line      =
  flux      =
  refant    =
  minants   =
  interval  = 0.1
  nfbin     = 4
  tol       =
  xyphase   =
  options   = xyvary
```

The inputs for this task reflect that we want to solve for how the gain changes on short timescales (`interval=0.1`, which selects individual 10 second cycles) and that we will allow the system response to vary from one 512 MHz chunk to another (`nfbin=4`). Although the gain response may not, and hopefully will not, change much over the bandwidth, the polarisation leakages probably will. This task will now determine a leakage solution for each 512 MHz bin. We could make **gpcal** determine solutions for up to 16 bins (128 MHz chunks, which is the same bandwidth as the original ATCA had), but we will be imaging 512 MHz chunks in this tutorial, so we stick with `nfbin=4`.

When you run this task, the solutions it obtains will be displayed to the terminal. If you take a careful look at the leakage solutions you should note that they don't change much over the bandwidth. Once again, 1934-638 is a very useful calibrator for this type of calibration, since it has essentially zero linearly-polarised flux (Stokes Q,U = 0). This means that any measured linear polarisation must be because of leakage between the two receptors in the antenna.

We can now look at the quality of the calibration solution. We use the **uvplt** task first, to look at the amplitude vs time:

```
miriad% inp uvplt
```

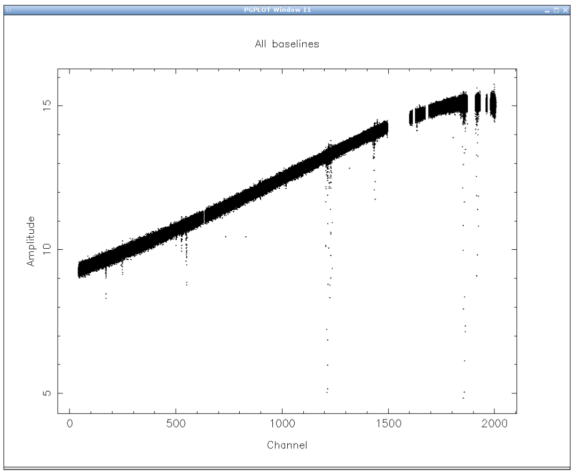

Figure 4: The screen you should see after running **blflag** on 1934-638.

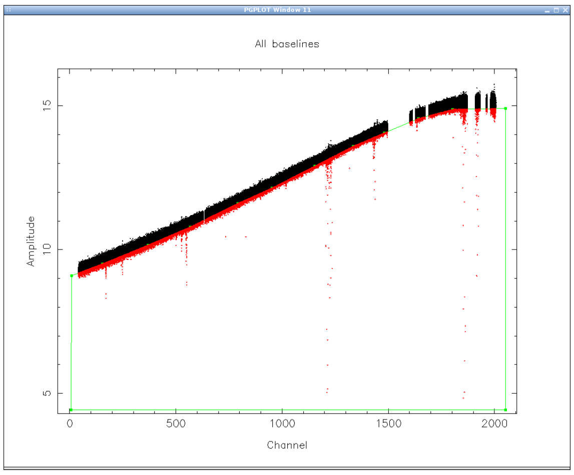

Figure 5: The screen you might see after selecting a polygon to flag.

```
Task:   uvplt
vis      = 1934-638.2100/
line     =
select   =
stokes   = xx,yy
axis     = time,amp
xrange   =
yrange   =
average  =
hann     =
inc      =
options  = nofqav
subtitle =
device   = /xs
nxy      =
size     =
log      =
comment  =
```

The output of this task can be seen in Figure 6. A plot of phase vs time (the same inputs to **uvplt** above, but change `axis=time,phase`) is shown in Figure 7. The amplitude plot shows a large amplitude spread that stays roughly constant with time. This large amplitude spread can also be seen in Figure 3, and is because the flux of 1934-638 varies by quite a bit over the 2 GHz frequency range. The phase plot shows a tight grouping around 0 degrees, which is what it should be. There are some outlying phase points, but nothing to be too concerned about at this stage.

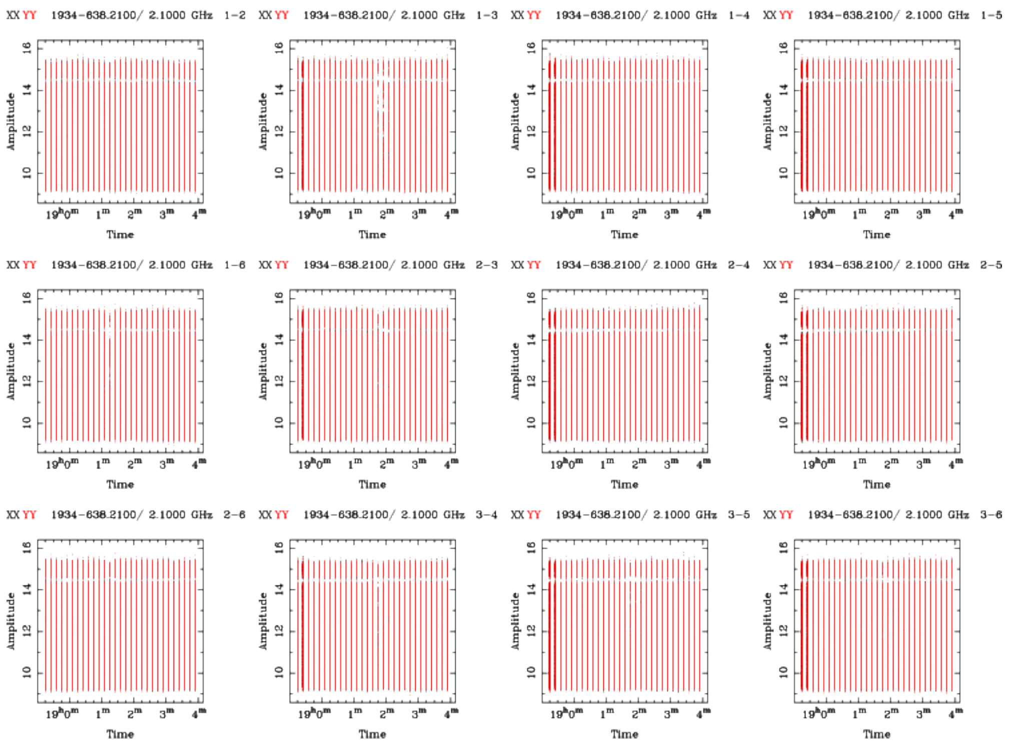

Figure 6: The amplitude of 1934-638 versus time, after calibration.

There is another plot that can show us roughly this same information in a single panel:

```
miriad% inp uvplt
  Task:    uvplt
  vis       = 1934-638.2100/
  line      =
  select    =
  stokes    = xx,yy
  axis      = real,imag
  xrange    =
  yrange    =
  average   =
  hann      =
  inc       =
  options   = nofqav,nobase,equal
  subtitle  =
  device    = /xs
  nxy       =
  size      =
  log       =
  comment   =
```

The options here tell **uvplt** to put a point on the plot for each channel and time visibility (if you left out `options=nofqav` **uvplt** would instead only plot a point for each time visibility, that being the average over all the channels), to put the points for all baselines on the same panel (`options=nobase`), and to make the scaling on each axis the same (`options=equal`).

Figure 8 shows the output of this task. We see clump of points that has a spread along the real axis and is tightly constrained around 0 along the imaginary axis. For a well calibrated point source that has a varying amplitude over the frequency range you're plotting, this is what we expect. Can you guess at what this plot would look like for a point source that had a roughly constant amplitude across the band?

At this point, the bandpass and flux calibration for 1934-638 is complete. We now proceed to calibrate the phase calibrator, which for this observation is 2058-425. We begin by copying over the calibration solution from 1934-638:

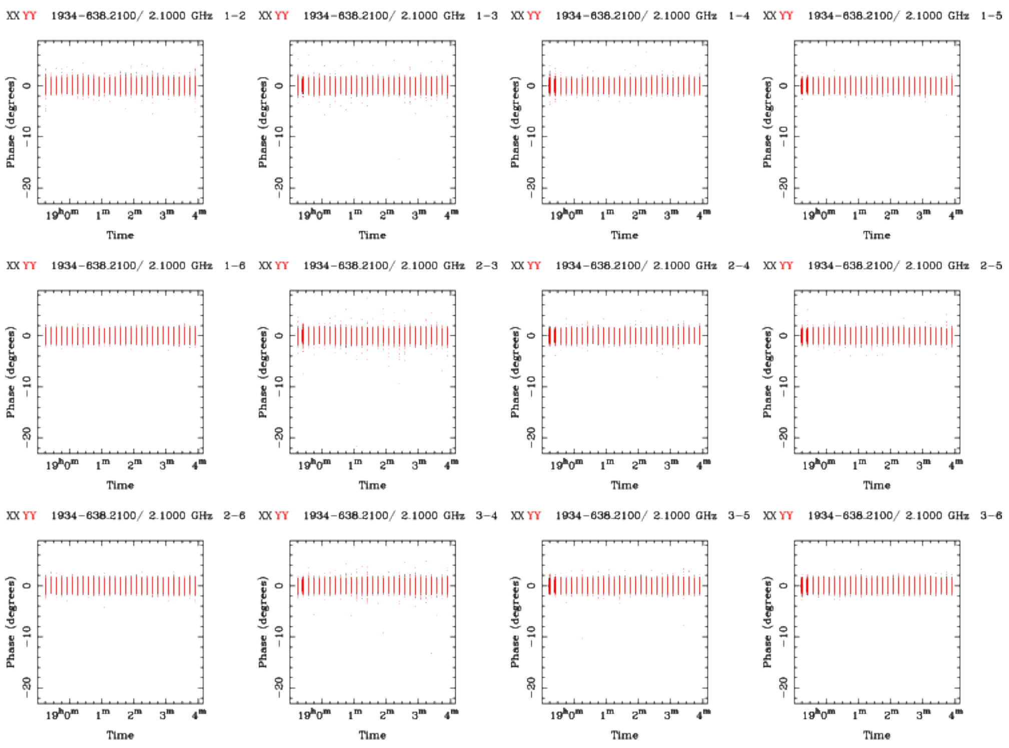
Figure 7: The phase of 1934-638 versus time, after calibration.

```
miriad% inp gpcopy
  Task:   gpcopy
  vis      = 1934-638.2100/
  out      = 2058-425.2100/
  mode     =
  options  =
```

At this point you can flag 2058-425 the same way that you did for 1934-638, with **pgflag** and **blflag**. Once you are happy with the flagging (you can check for RFI with **uvspec**), you can calibrate:

```
miriad% inp gpcal
  Task:   gpcal
  vis      = 2058-425.2100/
  select   =
  line     =
  flux     =
  refant   =
  minants  =
  interval = 0.1
  nfbin    = 4
  tol      =
  xyphase  =
  options  = xyvary,qusolve
```

The `qusolve` option here can be used to derive the fractional Q and U fluxes for this calibrator. This is only possible if the source has been observed over a large enough range of parallactic angle coverage. If it hasn't (it has in this case), **gpcal** will complain about the problem being degenerate and will fail. You will also get another leakage solution, albeit one where the solution derived from 1934-638 will be used as the initial guess. The final leakage solutions should be very close to that found for 1934-638, but you can optionally specify `options=nopol` to prevent **gpcal** from attempting to make another leakage solution.

Let's look at the real vs imaginary plot for this calibrator, in Figure 9. It looks much more round than the corresponding plot for 1934-638, but has essentially the same properties; this calibrator has also been quite well calibrated.

The only thing left to do is to reestablish the flux scaling from the flux calibrator. During calibration, the absolute values of the gains may be lost, but we can fix this with the **gpboot** task:

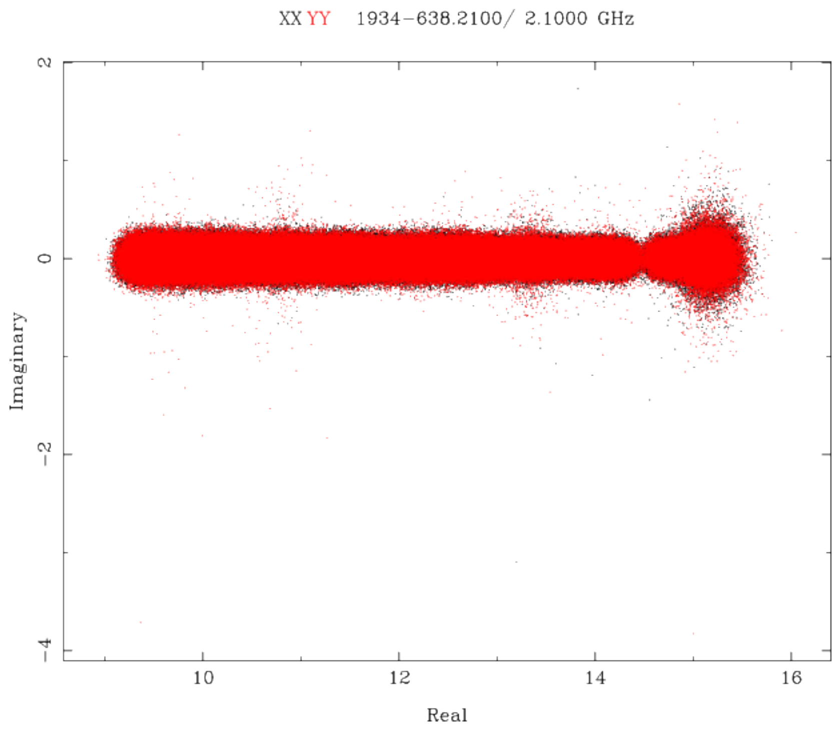

Figure 8: The real and imaginary parts of the calibrated visibilities for 1934-638.

```
miriad% inp gpboot
  Task:   gpboot
  vis      = 2058-425.2100/
  cal      = 1934-638.2100/
  select   =
```

Executing this command will display the scaling factors that were applied to bring the gains from the phase calibrator back to the absolute flux scale present in 1934-638. These factors will be displayed both for the entire set, and for each individual bin; the values should be ~ 0.95. With this step, we can now consider the calibration complete[1].

6 Imaging Preparation

We now have to copy the calibration tables from the phase calibrator to the source that we want to image, using **gpcopy**.

The phase calibrator brackets the observations of the project source 2051-377, and we assume that the gains change slowly enough that they can be linearly interpolated for using the solutions found on the phase calibrator. The MIRIAD tasks do this for any particular time it needs to determine a solution for by taking the closest solution before and after and linearly interpolating. Our calibration tasks have been making a solution for each cycle on the phase calibrator (and each cycle is 10 seconds), so the interpolation only takes into account 20 seconds worth of data. The phase calibrator is visited for 2 minutes each time, so we should use all this data. We do this by averaging the gain solutions using the task **gpaver**:

```
miriad% inp gpaver
  Task:   gpaver
  vis      = 2051-377.2100/
  interval = 2
  options  =
```

As long as our assumption that the gains vary slowly holds, the noise on the gain solutions will be smaller after this step.

We can use **pgflag** and **blflag** to flag this data as well, at least to remove the most obvious RFI.

Once you're happy with the flagging, you can apply the calibration solutions to the data using **uvaver**:

[1]We may see in the advanced tutorial that the calibration can be refined even further through the use of self-calibration.

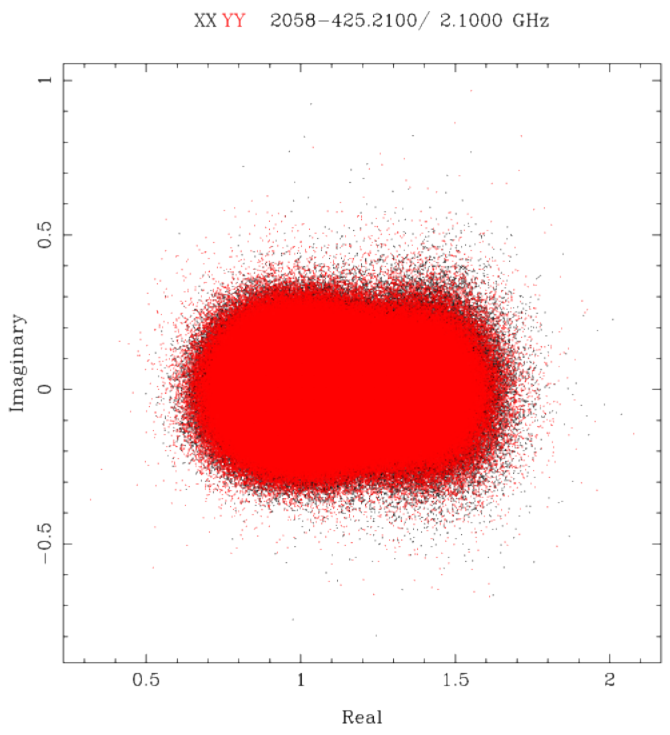

Figure 9: The real and imaginary parts of the calibrated visibilities for 2058-425.

```
miriad% inp uvaver
  Task:   uvaver
  vis      = 2051-377.2100/
  select   =
  line     =
  ref      =
  stokes   =
  interval =
  options  =
  out      = 2051-377.uvaver.2100
```

In normal use, any calibration that has been performed on a dataset is stored in calibration tables (which are just files within the dataset directory), and these tables can be altered, deleted or combined without actually changing the data. In this way, you can quickly investigate what each calibration table does, because you can opt not to apply it in most MIRIAD tasks. Applying the calibration solutions using **uvaver** alters the data, which is why a new dataset is created. Once you have applied the calibration, you can use the new dataset to calibrate potentially smaller effects, usually through the process of self-calibration.

We now have to split the dataset into smaller frequency chunks before we can image it. The large fractional bandwidth represented by the ATCA 16cm band means that the size, shape and position of the synthesised beam and its sidelobes will vary enormously across from one end of the band to the other. This will make it very difficult to effectively deconvolve the image, so we make the task more manageable by restricting the fractional bandwidth:

```
miriad% inp uvsplit
  Task:   uvsplit
  vis      = 2051-377.uvaver.2100
  select   =
  options  =
  maxwidth = 0.512
```

The `maxwidth` parameter is set to the maximum bandwidth in any particular dataset, in GHz. This command will make four new datasets, each 512 MHz wide, centred on 1332, 1844, 2356 and 2868 MHz.

7 Imaging

We will look at the imaging process for only one of the sub-bands that we made, but the process will be the same for all the other sub-bands as well. We choose to image the 2868 MHz band first:

```
miriad% inp invert
  Task:   invert
  vis      = 2051-377.2868/
  map      = 2051-377.2868.imap
  beam     = 2051-377.2868.ibeam
  imsize   =
  cell     =
  offset   =
  fwhm     =
  sup      =
  robust   = 0.5
  line     =
  ref      =
  select   =
  stokes   = i
  options  = mfs,double
  mode     =
  slop     =
```

The **invert** task makes two output files, one being the dirty map (specified with the `map` parameter) and the dirty synthesised beam (the `beam` parameter). We choose to make a Stokes I image, and we specify `options=mfs` to make a single-plane continuum image. The `robust=0.5` parameter indicates that we want to use robust weighting, while the `options=double` makes it possible to effectively deconvolve this image with the standard **clean** task later.

After executing this task, make a note of the theoretical rms noise level that **invert** expects; it should be around 4.6×10^{-5} Jy.

You can view the output dirty map `2051-377.2868.imap` using either KVIS or the MIRIAD task **cgdisp**:

```
miriad% inp cgdisp
  Task:   cgdisp
  in       = 2051-377.2868.imap/
  type     = p
  region   =
  xybin    =
  chan     =
  slev     =
  levs1    =
  levs2    =
  levs3    =
  cols1    =
  cols2    =
  cols3    =
  range    =
  vecfac   =
  boxfac   =
  device   = /xs
  nxy      =
  labtyp   = hms,dms
  beamtyp  =
  options  = wedge
  3format  =
  lines    =
  break    =
  csize    =
  scale    =
  olay     =
```

The image, as displayed by **cgdisp** is shown in Figure 10, while the central quarter of the beam (using `region=quarter` in **cgdisp**) is shown in Figure 11. If your first impression is that Figures 10 & 11 look very similar, you'd be correct. If we assume there is a point source at the field centre (there should be, we pointed the telescope at a known source), and the synthesised beam is effectively the response of the telescope to a point source (more commonly known as the point-spread function, or PSF), it should not be surprising that the image would look very like the beam. If you look carefully at the image though, you should see other point sources which also look like the beam, albeit fainter copies of it.

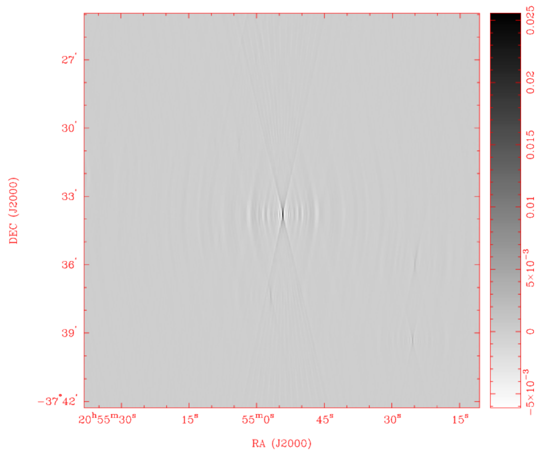

Figure 10: The dirty map made for the 2051-377 field, centred at 2868 MHz.

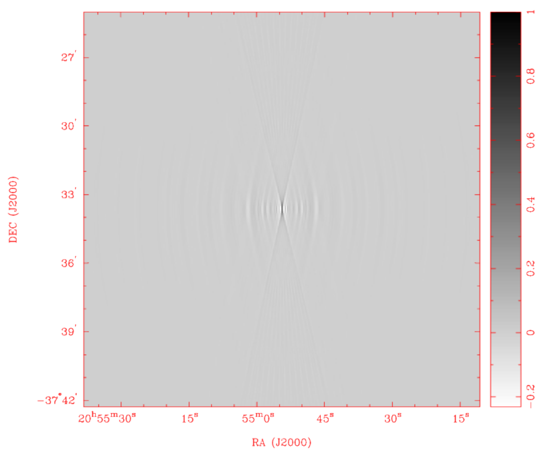

Figure 11: The central quarter of the dirty beam made for the 2051-377 field, centred at 2868 MHz.

Through the process of deconvolution, we can determine what the real sky looks like by disentangling it from the response of the telescope. There are a few MIRIAD tasks which deconvolve via different methods, but the one we will use here is called **clean**:

```
miriad% inp clean
  Task:   clean
  map      = 2051-377.2868.imap
  beam     = 2051-377.2868.ibeam
  model    =
  out      = 2051-377.2868.imodel
  gain     =
  options  = negstop,positive
  cutoff   = 5e-5
  niters   = 3000
  region   =
  phat     =
  minpatch =
  speed    =
  mode     =
  clip     =
```

This task requires the dirty map and beam from **invert** as the `map` and `beam` input respectively, and it outputs a model of the sky as the `out` dataset. The clean process is an iterative one, where the model components are found and subtracted from the dirty map. The conditions for stopping this process can be set in a number of ways. The `cutoff` parameter stops the process if the absolute value of the amplitude of the largest component found in the cycle is less than this value; in this case we set it to be approximately the expected rms noise level in the image. For Stokes I images, it can be handy to set `options=negstop,positive` since it will prevent **clean** from finding negative amplitude components (Stokes I is strictly positive). Finally, to prevent the process from running too long, we can set the `niters` parameter, in this case to 3000, to absolutely limit the number of iterations that may be performed.

Running this task should produce something like the following output:

```
Begin iterating
 Clark  Iterations: 39
 Residual min,max,rms:   -9.926E-04   2.293E-03   7.271E-05
 Total CLEANed flux:    2.866E-02
 Clark  Iterations: 914
 Residual min,max,rms:   -6.813E-04   2.364E-04   5.034E-05
 Total CLEANed flux:    5.928E-02
 Clark  Iterations: 3000
 Residual min,max,rms:   -6.433E-04   1.526E-04   4.619E-05
 Total CLEANed flux:    8.862E-02
 Stopping -- Maximum iterations performed
```

The information here is important in diagnosing how the procedure is progressing. At each major cycle (where the full beam subtraction is performed), the residual image statistics are printed; the minimum and maximum values, and the rms should all be decreasing (in the absolute sense, for the minimum) as long as **clean** is proceeding well. Meanwhile, the sum of all the flux found in the clean components should be increasing.

The model itself is just a collection of delta functions of varying amplitudes, each one representing a point-source component of the sky. To make an image from this model, we use the **restor** task:

```
miriad% inp restor
  Task:   restor
  model    = 2051-377.2868.imodel
  beam     = 2051-377.2868.ibeam
  map      = 2051-377.2868.imap
  mode     =
  fwhm     =
  pa       =
  out      = 2051-377.2868.irestor
```

You need to supply the dirty map and beam as inputs, along with the model from **clean**. From the dirty map, **restor** subtracts the dirty beam convolved with the model components, leaving what is termed the residual image. You can make **restor** output this residual image by specifying `mode=residual`. A clean beam is created by approximating the size and shape of central lobe of the synthesised beam, without the sidelobes that corrupt the dirty map. With this clean beam, **restor** creates the clean image by adding back to the residual image the model components convolved by the clean beam. This is what is output when you execute this task with the inputs given above.

Using **cgdisp** to display this clean image, this time setting `range=0,0,log` so we can see detail at different levels, we get something like that shown in Figure 12.

This image is a lot better than the dirty map. We can see a number of sources now that were previously hidden underneath the sidelobes from the bright central source. The image is by no means perfect however, as dirty beam structure is still visible, coming from what appears to be a couple of sources just outside the imaging region. We will not concern ourselves with these artifacts here, but will instead leave them to an advanced tutorial.

Now that we can clearly see our source, we need to measure its flux density. We can do this using the **imfit** task:

```
miriad% inp imfit
  Task:   imfit
```

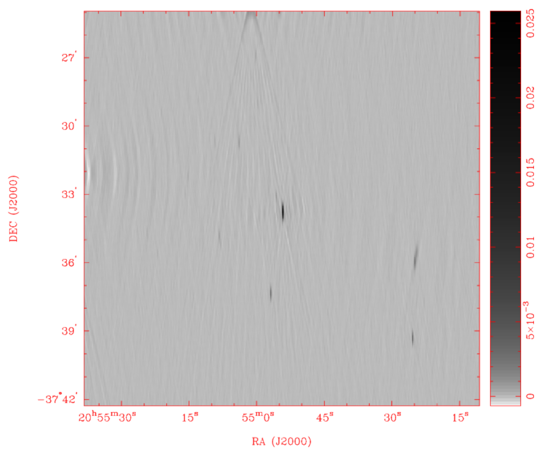

Figure 12: The clean image, centred at 2868 MHz.

```
  in       = 2051-377.2868.irestor
  region   = quarter
  clip     =
  object   = point
  spar     = 1,0,0
  fix      =
  out      = 2051-377.2868.iresidual
  options  = residual
```

The inputs here tell **imfit** to look for a point source (`object=point`), centred roughly on the phase centre (`spar=1,0,0`). We further restrict it to only considering the central quarter of the image to prevent the task from erroneously fitting some source on the outskirts of the field. We also instruct **imfit** to make a residual image, ie. with the fitted source removed.

The output of this task should look something like this:

```
-------------------------------------------------
Object 2051-377
RMS residual is 8.88E-05 (theoretical image noise is 4.60E-05)

Using the following beam parameters when
deconvolving and converting to integrated flux
  Beam Major, minor axes (arcsec):    23.73     2.30
  Beam Position angle (degrees):       0.4

Scaling error estimates by  4.4 to account for
noise correlation between pixels

Source  1, Object type: point
  Peak value:                 2.6930E-02 +/-  1.2575E-04
  Offset Position (arcsec):      -2.450    -9.613
  Positional errors (arcsec):     0.006     0.066
  Right Ascension:                20:54:54.194
  Declination:                    -37:33:48.613
-------------------------------------------------
```

This tells us that the source has a flux density of 26.9±0.1 mJy, and that the position is offset from the phase centre by about 2.45 arcseconds in R.A. and -9.61 arcseconds in Dec. The residual image looks like Figure 13. The fit isn't perfect, which could be for a number of reasons, but the amount of leftover flux will be of the order a few mJy, which is less than 10%. Improving this accuracy will be left to the advanced tutorial.

7.1 The rest of the band

This tutorial will not explicitly deal with the other three chunks of 512 MHz, but the process of imaging and measurements will be the same. Once you have done that, you should be able to use your measurements to determine the spectral index of the central source.

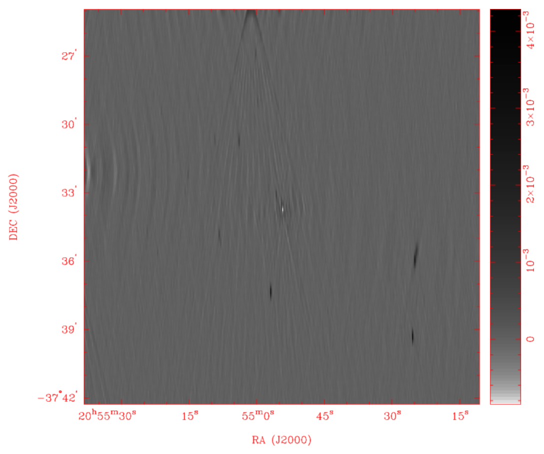

Figure 13: The clean image, centred at 2868 MHz, after a point source model of the central source is subtracted by **imfit**.

8 Summary

After this tutorial, you should be familiar/comfortable with using MIRIAD to:

- Load data from RPFITS format with **atlod**.
- Calibrate using **mfcal**, **gpcal**, **gpcopy** and **gpboot**.
- Flag bad data with **uvflag**, **pgflag** and **blflag**.
- Determine calibration and flagging effectiveness with **uvspec** and **uvplt**.
- Alter and split datasets with **uvaver** and **uvsplit**.
- Imaging and deconvolution with **invert**, **clean** and **restor**.
- Displaying images with **cgdisp**.
- Measuring source properties in images with **imfit**.

The advanced continuum tutorial, which also deals with this data, will introduce you to tasks which will help you get even better images and measurements via self-calibration, multi-frequency cleaning and clean boxes.