Miriad data reduction tutorial – Continuum

The aim of this tutorial is to introduce you to the most commonly used tasks in Miriad and to show you how best to use them to reduce a CABB continuum dataset. This tutorial will go through the steps required in broad detail, but many of the finer points will be left to you to figure out, using information in the Miriad manual. Some of the trickier parts will be filled in already. You may also email [Jamie.Stevens@csiro.au](mailto:Jamie.Stevens@csiro.au) for assistance, or go to the ATCA Forum at http://atcaforum.atnf.csiro.au.

1. Get the data!

The data we will use for this tutorial comes from the ATCA Target of Opportunity (ToO) experiment CX184. We will be using CABB data from 9 February 2010, which is now out of its proprietary period and available for anyone to use. From the Australia Telescope Online Archive (ATOA, <http://atoa.atnf.csiro.au>), please obtain this data and put it somewhere you can start reducing it.

1. Start Miriad.

While you are waiting for the data to download, check that you can start Miriad by going to the directory where you are downloading to the data to and typing 'miriad' (leave out the quotes). If this doesn't work, it's likely that your environment variables are not correctly set up – talk to your demonstrator about this. If it has worked, then you may start reading about the reduction process by visiting the Miriad Users Guide at <http://www.atnf.csiro.au/computing/software/miriad/userguide/userhtml.html>

1. Unpack the data, using either the tar command (tar xvf datafiles.tar) or zip (unzip datafiles.zip).
2. Load the data into Miriad format using the task 'atlod'.

To bring up the inputs to atlod, type 'inp atlod' at the 'miriad' prompt. You should see something like the following:

Task: atlod

in = \*.CX184

out = cx184.uv

ifsel =

restfreq =

options = birdie,rfiflag,noauto,xycorr,opcorr

nfiles =

nscans =

edge =

At any time for any task you may bring up its help page by typing 'help' at the prompt (which will bring up the help on the current task), or by typing 'help atlod' (for example). Alternatively you may bring up the help page on the web site:

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

For now, we want to load all the files we just got from ATOA into a new uv file. As a hint, the options we will want to use are 'options=birdie,rfiflag,xycorr,opcorr,noauto'. You should read about what these options do before loading the data. When you’re ready to run a Miriad task, type ‘go’ + Enter.

1. We now want to split the uv data into separate source/frequency sets. To do this we will use the task 'uvsplit':

Task: uvsplit

vis = cx184.uv

select =

options =

maxwidth =

Split the data and you should find that there are the sources 0420-014, 0434-188, 0440-003, 0458-020, 1934-638 and kteri, and there should be frequencies 1750, 5500, 9000, 17000 and 19000 (all values in MHz).

1. These data were obtained to determine whether a newly discovered source in x-rays was detectable at radio wavelengths, and what its spectral properties were. This is also what we will try to determine with the reduction for this tutorial.

First though we will need to plan our reduction. For each frequency, you must do a separate reduction and measurement process, and thus will need to specify a number of sources to use at each frequency. Identify the:

1. bandpass calibrator (a calibrator with a high flux density at the required frequency)

2. flux density calibrator (a calibrator that has a known flux density at the required frequency)

3. phase calibrator (a point source nearby to the project source)

4. project source

for each frequency and make a note of this.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1750 | 5500 | 9000 | 17000 | 19000 |
| Bandpass cal | 1934-638 | 1934-638 | 1934-638 | 0420-014 | 0420-014 |
| Flux density cal | 1934-638 | 1934-638 | 1934-638 | 1934-638 | 1934-638 |
| Phase cal | 0440-003 | 0434-188 | 0434-188 | 0458-020 | 0458-020 |
| Project Source | kteri | kteri | kteri | kteri | kteri |

1. The first step in any reduction is to determine a solution for the bandpass using the bandpass calibrator. This is done with the Miriad task 'mfcal':

Task: mfcal

vis =

line =

stokes =

edge =

select =

flux =

refant =

minants =

interval =

options =

tol =

For these data, it is only really necessary to set the 'vis' parameter and type 'go'. Choose a frequency (not 1750!) to start with and proceed. You should see some output which indicates the number of iterations required to converge upon a solution; good data will converge within 1-3 iterations, while if the data has problems it will take longer.

1. You should now check your solution in a number of ways. The first task we will use is 'uvplt' which will allow you to plot various uv variables against each other. Since it is likely in this data that there will be problems, uvplt will be very useful:

Task: uvplt

vis =

line =

select =

stokes =

axis =

xrange =

yrange =

average =

hann =

inc =

options =

subtitle =

device =

nxy =

size =

log =

comment =

We will look at our bandpass calibrator's amplitude vs time. An important thing to note is that, by default, uvplt will average all the frequencies together into a single point when plotting; most of the time this is not what we will want, so please remember to include 'options=nofqav' to disable this behaviour. Also, for now, it will be fine to use the standard 'device=/xs' PGPLOT device. The last thing to note is that we should look at the data products that the telescope observed, which are 'stokes=xx,yy' (we can ignore the cross-polarisation products xy and yx).

When you make your plot, you may notice that the amplitude on one or more baselines will be appear very wrong! During the process of setting up the array at the beginning of observing, the delays, amplitudes and phases will most likely be incorrect, and the observer will take some time in making them usable; this setup will more than likely be performed while looking at the bandpass calibrator. These 'setup' data are still recorded to file as normal and not marked in any special way, so if you don't have the observing logbook (or the observer didn't mark the times to discard in the logbook) you will need to determine for yourself which data should be discarded. By trying to determine a bandpass solution using this setup data may make a very weird uvplt. To see what the data looks like without the bandpass (and initial gain) solution, specify 'options=nopass,nocal' (although don't remove the 'nofqav' option!). If you can see any change in amplitude during the time, you will need to flag out that data in the next step.

1. There are many tasks in Miriad to flag data, but the most useful one for discarding setup data is the all-purpose 'uvflag'. You should, from the uvplt you just made, know the range of times that you want to flag as bad. Use the uvflag task:

Task: uvflag

vis =

select =

line =

edge =

flagval =

options =

log =

To mark data as bad you will need to set 'flagval=flag'. Although you may go backwards and mark data as good if you accidentally flag out too much data, it is fraught with peril as some fraction of the data will already be marked bad due to internally-generated RFI, and off-source flagging. By 'unflagging' this data, you will usually make your task harder rather than easier; the lesson here is to double check your inputs before flagging! Flag out your setup data and then redo steps 7 and 8 until the solution converges quickly and the plot of amplitude vs time looks sensible.

1. Once you are happy with the amplitude vs time plot, check that the spectral quality of the bandpass solution is suitable by using the task 'uvspec':

Task: uvspec

vis =

select =

line =

stokes =

interval =

hann =

offset =

options =

axis =

yrange =

device =

nxy =

log =

You can keep the same device setting, and also stokes, but you should change 'axis=chan,amp' to look at the spectrum. You will also not generally need any options to begin with.

What you will see is a plot of the spectrum as a function of amplitude vs channel number, and you will get one spectrum (well two if you count the different coloured polarisation quantities) for each baseline at each cycle time. You will need to cycle through these spectra by pressing 'Enter' repeatedly in the controlling window. What you are looking for is a spectrum that is smooth and continuous, discounting any breaks caused by ranges of channels that have been flagged (probably due to RFI). The spectrum itself may or may not have a slope – the Miriad manual attempts to explain why this is; do you understand it? You may be interested in looking at what the spectrum looks like without bandpass correction: specify 'options=nopass' to really see the difference! Once you're happy that you have a good bandpass solution, you can move on.

1. The next step is to calibrate the gains and leakages with the task 'gpcal':

Task: gpcal

vis =

select =

line =

flux =

refant =

minants =

interval =

nfbin =

tol =

xyphase =

options =

This task is operated much like mfcal, but we usually specify much shorter time intervals to solve for the changing gains. Here, you should set 'options=xyvary' and 'interval=0.1' (which will mean it will try to make a gain solution for every 10 second integration). This task will produce a lot more output than mfcal does, but it is doing the same iterative procedure, and the number of iterations required should not be more than about 4. Make sure to take account of any warnings emitted by gpcal and ensure they will not cause trouble for you. If you are happy with the output of gpcal, move on to examine the data again.

1. Now we will look at the phases as a function of time with uvplt. For a good gain solution, the phases should stay centred at 0 degrees for the entire time, and shouldn't have too large a spread (10-20 degrees is OK).

Another useful diagnostic is to look at a real vs imaginary plot. When you do this you should use 'options=nofqav,nobase,equal' to keep the plot square and put all the baselines onto a single plot. For a source with little flux variation over the band, this plot should appear as a circle centred on an imaginary value of 0 and a real value of the flux of the source. The radius of this circle shouldn't be more than 1 or 2. If the flux of the source does vary across the band, you will commonly see a “dumbbell” or “cigar” shape instead of a circle – this is normal. If you see points a long way from the circle, or the circle has a much larger radius, you will need to do some more flagging. Useful tasks for this are blflag, and mirflag, although these will not be described at the moment, since you shouldn't need them for this data.

If you have made all these plots and are happy that they look like good data, then you have finished with your bandpass calibration and can move on.

1. You will now deal with your flux density calibrator. If you have used the flux density calibrator as the bandpass calibrator as well, then you may skip to step 15.

First, you must copy the bandpass solution from the bandpass calibrator to the flux density calibrator with the task 'gpcopy':

Task: gpcopy

vis =

out =

mode =

options =

Now that your flux density calibrator has the bandpass solution from the bandpass calibrator, all you need to do is calibrate its gains using gpcal, in the same way as shown in step 11. You may also need to flag the data, and be sure to check the solutions you get in uvplt and repeat the process until you are satisfied with your solution.

1. Since you determined your bandpass solution using a calibrator whose flux density behaviour is not known, the bandpass solution will not actually be strictly correct – there may be a missing slope term. But we can correct this problem using the flux density calibrator.

We begin by matching the bandpass calibrator's gain scaling to that of the flux density calibrator, using the task 'gpboot':

Task: gpboot

vis =

cal =

select =

Here, 'vis' will be the bandpass calibrator dataset, while 'cal' will be that of the flux density calibrator. After you type 'go' you will see a message telling you by how much the gains were scaled on the bandpass calibrator dataset to agree with the flux density calibrator; this number shouldn't be too far away from 1, but depending on the array calibration during the observations, there's no reason why it might not be.

After this, we need to correct the bandpass slope, and we do this with the task 'mfboot':

Task: mfboot

vis =

line =

select =

flux =

mode =

clip =

device =

options =

This task takes a number of inputs that may not be immediately obvious. The 'vis' parameter should be set to both the flux density calibrator dataset and the bandpass calibrator dataset, separated by a comma. The 'select' keyword should be set to select the flux density calibrator by name, eg. 'select=source(1934-638)'. You should also set 'mode=scalar' whenever your flux density calibrator is not a planet; when it is a planet, you should leave mode unset. You can, if you like, set device to see what mfboot is doing, although it isn't necessary, and sometimes it won't produce any plot output. When you run this task, you will be told again how much the gains were scaled by (hopefully this number should be close to 1), and by what amount the bandpass solution's slope was shifted by. If these numbers are not 1.0 and 0.0 respectively, it is often a good idea to rerun mfboot: it should only take a couple of runs before it will no longer need to correct the data.

At this point, both your bandpass calibrator and flux calibrator are well calibrated, and your bandpass solution will be “correct”, and you can continue on to the next stage.

1. We now want to take our “correct” bandpass solution and apply it to our phase calibrator. We do this with gpcopy, which is described in step 13 (which you may not have read if your bandpass calibrator was the same as your flux density calibrator).
2. To get an idea of how the gains changed during the observation, we use the phase calibrator. To calibrate its gains, use the task gpcal in the same way as we did for the bandpass and/or flux calibrators in the previous steps. If your phase calibrator has enough parallactic angle coverage, you may also want to include 'options=qusolve' in gpcal to get some accurate polarisation measurements. Make sure to check with uvplt that the solutions look valid.
3. When you use gpcal to create a gain solution, it will not necessarily have the correct flux scaling, so we need to compare it to a gain solution that does have the correct scaling to fix it. We do this with gpboot, using the bandpass calibrator dataset (or flux calibrator dataset if you’d like, there will be no difference since at the moment both datasets have the same flux scaling) as the ‘cal’ input (see step 14 for more info). You will not need to use mfboot at this stage since the bandpass solution won’t need to be corrected. The scaling from gpboot should not be too far away from 1.
4. At this point it is instructive to look at the gain solutions that gpcal has made, using the Miriad task ‘gpplt’:

Task: gpplt

vis =

device =

log =

yaxis =

options =

nxy =

select =

yrange =

To look at the gain solutions, you should specify ‘options=gains’, and to see how the amplitude scaling changes with time specify ‘yaxis=amp’, while to see how the phases are varying with respect to the reference antenna, specify ‘yaxis=phase’. Look at both the amplitude and phase gains and ensure that you understand what it is showing you before moving on.

1. From step 18, you should notice that for each visit to the phase calibrator, gpcal has made several solutions for the amplitude and phase gains. This is OK since it shows us how the gains are varying over short timescales, but the primary purpose for these gain solutions is to allow us to interpolate to determine what the gains were doing while we were pointing at our project source. Miriad does this interpolation by taking the last gain solution before the time it is interpolating to, and the first gain solution after, and doing a linear fit. It would therefore make sense to take the average gain solution from the phase calibrator as these references, since the noise on average points will be lower than on single points. To average the gain solutions over time we use the task ‘gpaver’:

Task: gpaver

vis =

interval =

options =

We should set the ‘interval’ here to be equal to the time we spent on the phase calibrator during each visit – you will need to determine this value!

Once you have done this averaging, go back to gpplt and plot the gains vs time again, and notice how they have changed.

1. We are now ready to make some images of the project source. First though you need to copy over the gain solution from your phase calibrator to your project source.
2. We will use the Miriad task ‘invert’ to make the dirty map of the project source:

Task: invert

vis =

map =

beam =

imsize =

cell =

offset =

fwhm =

sup =

robust =

line =

ref =

select =

stokes =

options =

mode =

slop =

The invert task parameters ‘vis’, ‘map’, ‘beam’,‘stokes’ and ‘options’ need to be set to begin with; for the moment you can leave all the other parameters unset. We will make an image in Stokes I, or total intensity. We will also be making a continuum map, so ‘options=mfs’ is required. Lastly, since we will want to deconvolve this image at some point, it is easier to do this with a double-sized beam; specifying ‘options=double’ will do this for us.

When you run this task, you will see the weighting scheme that is being used (take note of this) and the theoretical RMS noise level in the image – this value will be useful when doing a CLEAN.

When the image has been made, take a look at it in kvis (a non-Miriad program), or if you’re feeling adventurous, you could try using Miriad’s cgdisp! Make sure you understand why the dirty map looks the way it does before moving on.

1. The purpose of the CLEAN algorithm (and other deconvolution algorithms) is to separate the components of the image that are due to the antenna response from the actual emission from the sky. Miriad’s primary deconvolution task is called clean:

Task: clean

map =

beam =

model =

out =

gain =

options =

cutoff =

niters =

region =

phat =

minpatch =

speed =

mode =

clip =

The ‘map’ and ‘beam’ inputs come from the invert parameters, and you will need to specify the ‘out’ parameter to be the model dataset you want to create. For ‘options’, I recommend ‘negstop,positive’, and you should set the ‘cutoff’ parameter to be 3 times (at least) the theoretical RMS noise level you were told by invert. For a simple field like this, you can set ‘niters=3000’; clean should finish its model before that number of iterations in any case.

Once you have made your model with clean, take a look at it in kvis or cgdisp – it doesn’t look much like an image does it? You should however find that most of the pixels that aren’t zero are near the source(s) that you saw in the dirty map. What do you think this model represents?

1. To make an image out of this model we use the Miriad task ‘restor’:

Task: restor

model =

beam =

map =

mode =

fwhm =

pa =

out =

All you need to do is set ‘model’, ‘beam’, ‘map’ and ‘out’ for this task and run it. You will be told the size and angle of the synthesized beam – is it sensible?

Now we look at the “clean” image. What do you notice about it compared to the dirty map?

1. At this stage, you should now have an image that clearly has a reasonably bright source near or at its centre. If you don’t you may have to speak to your demonstrator.

If you can see the source, then it’s time to make some measurements. The first measurement we will make is to determine the noise level present in the image. We do this with the task ‘imhist’:

Task: imhist

in =

region =

options =

cutoff =

xrange =

device =

log =

Most of the parameters here will remain unset, and hopefully you can figure out which ones do need to be set and how! Run it and you should see a histogram appear on your PGPLOT device (hint, hint) showing the distribution of flux densities in your clean image; imhist will also try to fit a Gaussian curve to this histogram, and it should fit quite well. In the controlling terminal, you will get lots of information about the image noise. The most important figures are that of the “Mean” (which should be very close to 0), and the “rms”, which is measured in Jy/beam. How does this measured value compare to the theoretical noise level that invert stated?

Next, we want to determine the flux density of the source that we can see in the image, and we use ‘imfit’ for that:

Task: imfit

in =

region =

clip =

object =

spar =

fix =

out =

options =

This task is a little trickier to configure than imhist, but we basically want to tell it to search the central quarter region of the clean image (hints everywhere) for a point source right at the centre of the image. Ask your demonstrator if you get stuck here. Finally, we will want to produce a “residual” image to check whether the fit this task performs is any good.

When you run imfit, it will give you a peak value (we’re looking for a point source after all), a position for the source (as an offset and an absolute RA & Dec), and their associated uncertainties. You should note all these numbers down for later. At the top of the output you should see it gives you an RMS residual number, which should be very close to the RMS you just measured with imhist, if the fit is a good one. You should now check that the residual image looks good; there should be no source at the centre anymore, and the noise level where the source was shouldn’t be noticeably different. If you’ve achieved this, congratulations!

1. Now do it all again, this time for the other frequencies! Get the flux of the source at each frequency before you proceed. (Hint, there may be issues at 1750 MHz, so do it last or skip it if you’d rather do some of the interesting things in step 27!)
2. With the data that you have gotten now, what is the spectral index of the source? Does it seem reasonable?
3. Alright, you’ve finished all that and you want more things to do. Try any of the following things if you get time:
   1. Make an image with natural weighting instead of uniform weighting. It should change what the dirty image looks like quite dramatically, and should reduce the RMS noise level in your clean image. Does it change your flux measurements? (You only need to do this for one frequency to see the effect).
   2. Does the spectral index stay constant over the entire frequency range? One way to check this is to use uvsplit to break up the band of one of the lower frequencies (where the fractional bandwidth is largest) and make a few measurements to check.
   3. Is this source really a point source? The imfit results might suggest that it is, but try using ‘uvamp’ to check.
   4. Make some polarisation images (ie. Stokes Q and U) of the project source. Does this source appear to be polarised, and to what extent?
4. A very interesting thing to do is to check that the data quality is as you’d expect for the observing parameters (ie., integration time, uv coverage etc.).

It can be difficult to do, but you can make a simulated dataset (of a simple field like this one) match almost exactly to an observed dataset using only the source strength measured from imfit and the output of the ATCA sensitivity calculator. This will also teach you about the Miriad task ‘uvgen’. Your task is to pick one of the frequencies that you have just reduced and try to replicate the image with a simulation.