MIRIAD Spectral Line Data Processing Tutorial

Jess O'Brien
(Substantially based on a Dave Rayner's Tutorial written for Continuum Data Processing with Miriad)

May 13, 2003

Contents

1. Introduction

This paper provides a tutorial introduction to processing simple ATCA "spectral line" data with MIRIAD. The tutorial aims both to demonstrate the fundamental steps for calibrating and imaging synthesis data, and also to show what real ATCA data looks like.

The tutorial assumes a basic knowledge of synthesis telescopes and imaging; you should know what are meant by visibilities, gain, bandpass, uv-plane, Fourier transform and deconvolution. No prior experience with Miriad is assumed.

1.1. A quick rant

In general, there is no "right" method for calibrating ATCA data. Nearly everyone has their own preferences for options and what tasks to do in what order, which can make processing ATCA data quite confusing for beginners (and so-called experts!). The calibration process followed in this tutorial follows that described in the *Miriad User's Guide*. Although the finer points of synthesis imaging are still a bit of a "black art", the improvements which can be obtained by tweaking the calibration and imaging process

1.2. For more information

- Miriad User's Guide: http://www.atnf.csiro.au/computing/software/miriad
- Synthesis Imaging in Radio Astronomy II, Taylor, Carilli & Perley.

2. Miriad

The Miriad package is a collection of tasks for processing radio astronomical data.Data processing in Miriad proceeds stepwise, at each step you can tun a task which will either modify a data-set (eg. delete corrupted visibilities), create a new data-set, or display some aspect of a data-set (eg. plotting visibilities vs. time). Often, the output data-set of one task becomes the input data-set for the next step.

Miriad tasks are really Unix executables, which are involed with a series of named arguments. They can be run from the Unix command line or by other programs 9eg. shell scripts - see Appendix A), but are usually run using Miriad shell. In the examples which follow, Miriad task names are usually written in `CAPITALS`.

Miriad data-sets are stored on disk as Unix directories - you can copy, rename or delete miriad data-sets like any other directory. However, you should not modufy the internal contents (the items) of a data-set, except via the Miriad tasks.

Note that Miriad data-sets do not contain revisions; you can not "undo" the changes! In most cases this si not a serious drawback, because many tasks generate new output data-sets. Thus, when you make a mistake you usually only have to re-do the last or last few calibration steps.

2.1. The Miriad Shell

The Miriad shell is a convenient environment for specifying arguments and options for miriad tasks, and executing them. It is an old, text-style interface. To start the Miriad shell, type:

```
miriad
```

The essential Miriad shell commands are:

help	Get help on a topic, eg. `help miriad` pr `help uvplt`
inp *task*	Select a task. Displays a list of arguments for the specified task, eg. `inp uvplt`. With no task argument, **inp** displays the list of arguments for the currently selected task.
key=value	Set the argument for the current tast eg. `vis = 1934-638.1413`.
unset *key*	Unset the argument ie. let is take the default value.
go	Execute the currently selected task.

3. A tutorial example: HI emission of the edge-on Sdm galaxy IC5052

This source is at declination $\delta = -69°$, sufficiently far south not to be shadowed.

3.1. Loading ATCA data into Miriad

The task to convert ATCA RPFITs data into a Miriad data-set is `ATLOD`. We also give it the rest frequency of the line we are interested, HI in this case. For example, to load data from a disk, change to your data directory, then:

```
cd /DATA/NELLE_2/synwork
mkdir [new dir]
cd [your dirname]
inp atlod
```

```
Task:                atlod
in                   = ../*.C894
restfreq             = 1.420405752
options              = birdie,noauto,bary,noif,xycorr
out                  = sum.uv
```

3.2. Splitting

Although you could work on `sum.uv`, it's more convenient to split the multi-source data-set into several data-sets, each a separate source and frequency. This is also important for calibration, as Miriad data-sets contain only a single set of calibration tables, it is rather poor at handling the calibration of data-sets containing multiple sources and frequency bands. For calibration purposes, it is best to work on datasets containing a single source and single frequency band

```
Task:                uvsplit
vis                  = sum.uv
```

`UVSPLIT` creates files in the current working directory of the form *source.freq* eg. `1934-638.1376`. It is rather unforgiving if you already have a files with one of the names that it wants to use. Make sure your directory is free of files that may usurp `UVSPLIT`'s name choice.

If I'ved observed simultaneously in two different frequencies, I usually create a subdirectory for each frequency (eg. `1418`), and distribute the data-sets accordingly.

3.3. Calibration Strategies

Reminder of basic theory

The feeds are modelled as having a composite gain comprising the time-variable antenna gain, the bandpass function (or gain varibility with frequency), the time-varying delay term and the time and frequency independent antenna leakage term. As both the antenna gains and the delay term vary with time, the calculated calibration is stored in Miriad as the `gains` item of a visibility dataset. The antenna bandpass functions are stored as the `bandpass` item, and the antenna leakage which is the leakage of one polarisation into another, is stored as the `leakage` item in a calibrated data-set.

Regardless of whether a feed is linearly or circularly polarised, its instantaneous response to a signal is a linear combination of two fo the four Stokes parameters that describe the wave. In the equatorial frame of the source, ideal linear feeds respond according to

$$\begin{aligned} \mathrm{XX} &= \mathrm{I} + Q \\ \mathrm{YY} &= \mathrm{I} - Q \\ \mathrm{XY} &= \mathrm{U} + i\mathrm{V} \\ \mathrm{YX} &= \mathrm{U} - i\mathrm{V} \end{aligned}$$

There are two distinct calibration situations with the ATCA, depending on whether the correlator configuration produced all four polarisation products (XX, YY, XY and YX) or just two (XX and YY). The former is done in continuum mode and in some spectral line modes, whereas the latter is true of other spectral line modes.

In the solution process, we must consider the time variability of the various parameters. The gains vary fairly rapidly with time, and at L band we typically solve for them every 30-60 minutes, with the assumption that they are steady over a solution interval of a few minutes - the calibrator scan. For dual polarisation systems the two polarisation bands are assumed to have independent gains and band passes, although the delay is common.

Calibration of Data with XX, YY, XY and YX

If all four polarisation products are measured, a full polarimetric calibration is possible to solve for the leakage terms, the gains, a bandpass function and calibrator Q and U.

The leakage terms, bandpass function and the calibrator Q and U are assumed to be steady over the entire length of the observation. Our experience so far appears to support this. It is not possible to solve for a calibrator's V (circularly polarised flux density) However, a calibrator's V is unlikely to be significant (< 0.1 %).

For polarimetric observations, as the polarisation of 1934-638 is known (in fact it is $< 0.2\%$), and the flux density is strong, it is possible to derive polarisation leakages from it given only a short observation (5-10 minutes). If you have good parallactic angle coverage of the secondary, it is possible to simultaneously determine the polarisation

leakages as well as Q and U for the secondary ("good parallactic angle coverage" means at least 5 observations spread over a parallactic angle range of 100°). This will be the normal case for a 12 hour observation of one source. Alternatively if there is poor parallactic angle coverage of the secondary (e.g. resulting from snapshot observations), the instrumental polarisation derived from 1934-638 can be used, and the Q and U of the secondary can still be determined. Use of the primary calibrator to determine the instrumental polarisation is only possibel for 1934-638, for which the polarisation is known. This approach cannot be used if an alternative primary calibrator is used.

For details of calibrating when all four polarisations are observed see the 'Miriad User's Guide' (www.atnf.csiro.au/computing/software/miriad)

Calibration of Data with XX and YY only

This is much simpler. You have insufficient information to solve for instrumental polarisation and the calibrator Q and U. You have no choice but to assume that these are zero. This means that we can skip a number of steps in the calibration process.

All these assumptions will result in some error in the calibration - this is fundamentally unavoidable if only XX and YY are measured. Provided the secondary calibrator is weakly polarised, the error is unlikely to be too substantial.

The primary calibrator is used to determine the absolute flux density scale, and where it has higher signal-to-noise than the secondary calibrator it is also used to calibrate the bandpass.

3.4. Correlator setup

For my observations I have used one of the more popular spectral line correlator configurations for HI work: `FULL_8_512-128`. This configuration produces 512 channels covering a total bandwidth of 8 MHz, sampling the XX and YY polarisations only. As the gain on each antenna is frequency dependent, the channels appear to have different flux levels. To account for this effect we do a bandpass calibration.

3.5. Examining and flagging the visibilities

The first thing you should always do is to look at the data. Was there interference? Is the Cacal data included? A detailed observing log will help here.

To view visibilities in Miriad use `UVPLT`.

```
Task:               uvplt
vis                 = 1934-638.1418
stokes              = xx,yy
axis                = time,amp
device              = /xs
```

The XX visibilities are the correlations between the X-polarised feed on one antenna and the X feed on another.

The visibility data for the primary calibrator should be almost constant. The delays are only calibrated after the cacal was run. In this case the cacal data is included in the dataset. The good post-cacal data is that after the phases have been brought to zero. Remove all the preceeding data by finding the time range of the data to be removed

and using the task UVFLAG. After this is removed it will be easier to discriminate any interference.

Use select parameter in UVPLT to find the range to flag - help select shows the syntax required to specify different time ranges..

When using UVFLAG run first with options = noapply to check you've specified all the right parameters, as it can be difficult to reverse the flaggin if you accidentially flag the wrong visibilities.

```
Task:                uvflag
vis                  = 1934-638.1418
select               = time(hh:mm:ss,hh:mm:ss)
flagval              = flag
options              = noapply
```

Having done this, re-inspect the visibilities with UVPLT. You can see there is a gap in the antenna 6 observations. This was data where th pointing drifted off source due to an ACC failure - this data was automatically flagged by ATLOD. There is no obvious need to further flag at this point.

3.6. Flux density and bandpass calibration

The absolute flux density scale for your observations is determined by an observations of a calibrator with known flux density - this is called the primary calibrator. At the ATCA, 1934-638 is almost always used as the primary calibrator.

The two Miriad tasks that calculate the gains, MFCAL and GPCAL know the flux density of 1934-638, and will scale the gains accordingly. MFCAL determines both the gains and the bandpass function, while GPCAL derives just the gains.

Use the secondary calibrator as the bandpass calibrator only when it is relatively bright and sufficient data has been collected, otherwise you are better off using 1934-638 as this has a well-known spectral index. If you have sufficient secondary data: you should determine the bandpass calibration of the secondary and use this if it better quality than that of the primary. This is identical to the determining the bandpass calibration for the primary, except that the flux density, and its variation with frequency, will generally not be known for the secondary. In this case mfcal assumes a spectral index of 0 and a flux density to make the rms antenna gains equal to 1.

In our case, we will use 1934-638 as our secondary is quite faint. To derive the bandpass dunction we use the task MFCAL to calibrate the flux density of 1934-638 and calibrate the bandpass which should remain constant over the duration of the observation.

```
Task:                mfcal
vis                  = 1934-638.1418
interval             = 0.1
```

The interval used is the time-step for the gains; setting it to 0.1 mintues, which is less than the ATCA integration time of 10 seconds, forcing MFCAL to solve for the gasins once per integration cycle. MFCAL usually issues a few warnings about correlations flagged and the flux scale used.

Quality of solutions

To assess the quality of the flux solution, we use UVPLT to plot Stokes I.

```
Task:                uvplt
vis                  = 1934-638.1418
axis                 = re,im
device               = /xs
options              = nobase
```

The 'nobase' option displays all the baselines on the one plot. To check that the solution is centred for all baselines, display the baselines separately by removing the options setting:

```
unset options
```

If there are significant outliers or large arcs this bad data will need to be flagged.

Similarly it is also good to check the amplitudes and phases yielded from the gains calibration are sensible. Use `axis=time,amp` and `axis=time,phase` to plot calibrated data on each baseline. As the calibrators are non-varying point sources the amplitude should be flat and constant on all baselines. Phases should be close to zero.

The best tool for such flagging is BLFLAG:

```
Task:                blflag
vis                  = 1934-638.1418
stokes               = xx,yy
axis                 = time,amp
select               = [as needed]
device               = /xs
```

Press `h` when the cursor is in the pgplot window, and BLFLAG lists the available commands. Use the left-mouse-button to click visibilities you wish to flag, the right-mouse-button to move to the next baseline, or if you wish to remove the flagging for the current baseline hit `c`.

To examine the bandpass solution, plot the antenna bandpass functions with the task GPPLT

```
Task:                gpplt
vis                  = 1934-638.1418
yaxis                = amp
device               = /xs
options              = bandpass
```

If there are spikes or dips in the bandpass you will need to flag separate channels. To do this you will need the task TVFLAG which allows interactive flagging of channels. This task also requires two additional programs: `xpanel` and `xmtv`. For `xmtv` to display the necessary full colour range you will need to remove all excess netscapes and xterms on your computer before starting up xmtv.

```
xmtv &
xpanel &
```

Then run TVFLAG if required.

```
Task:                tvflag
vis                  = 1934-638.1418
server               = xmtv@nelle
```

You can use the `select` parameter to just bring up the required visbilities. For more info, type `help select`.

To flag a particular channel, click 'Select Channel' and 'Flag Bad', then click the channel you wish to flag. 'Exit' moves on to the next baseline, while 'Abort' quits the task and doesn't apply any of the selected flagging.

After this further flagging undertaken, update the calibration solutions by running `MFCAL` again.

After this calibration of the primary calibrator is complete.

3.7. The Secondary Calibrator

Usually 1934-638 is only observed for the first 5-10 mintues of an experiment, and to calibrate the gains a calibrator much closer to your target is observed at least every hour to monitor the time dependent gains.

The secondary calibrator for this example is 2102-659. As before, first look at the data with `UVPLT` and flag out any bad visibilities. You will notice a few bad times. Flag out these bad times with `BLFLAG`.

Most of the visibility amplitude plots are fairly constant with time. It is very common for secondary calibrators to show extended structure on short basellines when observed at 20cm, or these baselines can be particularily affected by the solar interference. Miriad determines the gains assuming the calibrator is unresolved, and any baselines which show structure will cause the derived gains to be in error. If only one or two baselines show structure, just leave them out of the solution. As the gains are $antenna - based$, leaving out two baselines will not degrade the solution very much providing there are still baselines to every antenna.

So calibrating the secondary proceeds as follows: first copy the bandpass from the primary, then calibrate the gains using the secondary and correctly scale the flux with the primary.

To copy the bandpass and the flux scale across from the primary to the secondary calibrator data-set:

```
Task:               gpcopy
vis                 = 1934-638.1418
out                 = 2102-659.1418
```

3.8. Gains calibration

Then determine the time-variable gains from the secondary calibrator.

```
Task:               gpcal
vis                 = 2102-659.1418
interval            = 0.1
options             = noxy,nopol
```

Now examine the gains solution of the secondary for reasonableness. First, plot the stokes I solution as you did for the primary.

```
Task:               uvplt
vis                 = 2102-659.1418
axis                = re,im
device              = /xs
options             = nobase
```

Then check the gains calibration, by plotting the antenna gains as a function of time with GPPLT.

```
Task:                gpplt
vis                  = 2102-659.1418
yaxis                = amp
device               = /xs
options              = gains
```

Provided you equalised the gains at the start of your observation, your antenna gains should have an amplitude close to 1, although the phases should reflect atmospheric phase stability. If you see glitches in the antenna gains in time, it is probably due to bad data. Go back and look at the data near this time, and perhaps do some more flagging

Lastly, check how well the calibrated secondary data fits a point source with UVFLUX. As we have only XX and YY polarisations we have only measured the Stokes I parameter.

```
Task:                uvflux
vis                  = 2102-659.1418
stokes               = i
```

The first numeric column given for each Stokes parameter is the theoretical scatter. The second and third numeric columns are the mean visibility amplitude and phase. The phase should be near 0 for I, but could be either 0 or 180 degrees for Q and U (assuming there is signal in these!). V should be noise. The fourth column gives the actual scatter. This should be close to the theoretical value. Do not be concerned if it is a factor of 2 or so bigger. If it is more than a factor of a few greater than the theoretical, you probably still have bad data or a bad gain solution.

As we have already copied the flux scale across from the primary to the secondary calibrator, the gains of the secondary now reflect the true flux density of the secondary and calibration of the secondary is essentially complete.

If we had used the secondary to calibrate both the gains and the bandpass we would need to correct for the flux density of the secondary calibrator. The task to do this is gpboot.

3.9. Averaging the gains solution

Miriad's gain interpolation routines are pretty simple, and only use the nearest two gain solutions. This means that despite deriving the gains every 10 second integration, when you interpolate the gain solution across the target visibilitiesm only the first and last 10 seconds would be used. To overcome this, we take the average the gain solution of the secondary calibrator over the duration of each calibrator scan (3 - 5 minutes).

```
Task:                gpaver
vis                  = 2102-659.1418
interval             = 5
options              = vector
```

After running GPAVER the secondary calibrator visibilities will have a lot more scatterm especially the phase. This isn't important; the aim is to get a good gain solution for the target.

3.10. IC5052 - the target at last!

First things first, we just check the visilities for interference or other problems.

```
Task:               uvplt
vis                 = ic5052.1418
axis                = time,amp
device              = /xs
options             = nobase
```

Now copy the bandpass and gains across from the secondary calibrator to the target:

```
Task:               gpcopy
vis                 = 2102-659.1418
out                 = ic5052.1418
```

3.11. Removing the continuum emission

As we are primarily interested in the emission line data, to accurately image this we need to subtract the continuum baseline from all the spectra. This is done with the task `UVLIN`. If there are bright continuum sources in one's image, these will also generate sidelobes. Removing these sources in the uv-plane with `UVLIN` is the safest and recommended method.

However first we inspect the spectra of the narrow line emission with `UVSPEC`. You can see that the continuum is well-modelled by a linear baseline, except at the bandpass edge and in the region of line emission. Use the shortest baseline to get the estimate the full velocity coverage of the line emission.

```
Task:               uvspec
vis                 = ic5052.1418
interval            = 10000000
axis                = channel,amp
device              = /xs

Task:               uvlin
vis                 = ic5052.1418
out                 = ic5052.uvl
interval            = 10000000
chans               = [choose these]
options             = sun,twofit
```

If there exist very bright radio sources within the primary beam it is worthwhile shifting the phase center to the position of the radio source. To do this use `offset = x,y` where `x` and `y` are the east and north arcsec offsets of the radio source from the pointing center.

Alternatively if the sun is causing sunstantial interference throughout the target observation, use the `options = sun,twofit` option can greatly improve the continuum subtraction.

After the continuum subtraction, check the baseline spectra with `UVSPEC`.

And now you're ready to Fourier transform the visibilities to make an image.

3.12. Imaging

To make a "dirty image" the visibilities have to be interpolated onto a rectangular grid, weighted and Fourier transformed. The Miriad task which performs both of these functions is `INVERT`:

```
Task:               invert
vis                 = ic5052.uvl
map                 = ic5052.imap
beam                = ic5052.beam
line                = velocity,30,500,6.596,6.596
robust              = +0.4
stokes              = i
imsize              = 512,512
options             = double
```

This will make a dirty cube with 30 channels, starting at a recession velocity of 500 kms^{-1}. The `robust = +0.4` is an intermediate weighting scheme that sacrifices a small amount of sensitivity for an improved beam shape. See Lecture 7 in Taylor, Carilli & Perley for more information.

To view the dirty channel maps, either load it into kview, or run `CGDISP`:

```
Task:               cgdisp
in                  = ic5052.imap
type                = p
range               = 0,0,lin
labtyp              = arcmin
device              = /xs
options             = wedge,full
csize               = 1.2
```

This generates a pretty awful image, where sidelobe response from the bright sources limits the sensitivity across the image.

3.13. Deconvolution

Deconvolution algorithms attemp to remose the sidelobes around strong sources, by filling in the uv-plane. See Chapter 8 of Taylor, Carilli & Perley for more information. Deconvolution will make a big difference to the image sensitivity.

The safest way to deconvolve is to use `CLEAN`. It isn't a very good idea to just set `CLEAN` loose on an image without contraints - before running `CLEAN` you should specify those regions you think contain emission. The simplest first-pass way is to create a polygonal region with `CGCURS` using the total intensity (also called zeroth moment map) created from the dirty image with `MOMENT`. This way we have to define only one region which we will use for all velocity channels, instead of one for each channel, which is much more time-consuming.

```
Task:               moment
in                  = ic5052.imap
out                 = m0.dirty
axis                = 3
mom                 = 0
clip                = 0.001
```

```
Task:               cgcurs
in                  = m0.imap
type                = p
range               = 0,0,lin
labtyp              = arcmin
device              = /xs
options             = region
```

When you run CGCURS use the left-mouse-button to define vertices of the polygon, and the right-mouse-button to indicate you are finished. CGCURS creates or appends this region to the file cgcurs.region. You probably want to rename this file to avoid appending to it later by accident.

To run CLEAN use the following parameters:

```
Task:               clean
in                  = ic5052.imap
beam                = ic5052.beam
out                 = cln.imap
niters              = 500
region              = @cgcurs.region
gain                = 0.1
```

CLEAN's output will be a clean component model.

How do you know how many iterations to use? I start off with a conservative number, usually niters=500, and then I look to see how much side-lobe structure remains in the image. If CLEAN has removed all the side-lobes, or it the remaining structure doesn't look like sidelobes anymore, then its time to stop cleaning. As a general rule, faint complicated sources require many more iterations than a simple nrea-Gaussian point-like structure.

To look at what's left after you've cleaned, you convolve the CLEAN component model output with the dirty beam, and subtract it from the original dirty cube. Both these steps are performed by RESTOR:

```
Task:               restor
model               = cln.imap
map                 = ic5052.imap
beam                = ic5052.beam
mode                = residual
out                 = residual.imap
```

You can view the residual cube using CGDISP or kview. The residuals after 500 iterations, show there is still sidelobes left to clean.

When the non-sidelobes start to dominate the residuals, its time to stop cleaning. One must be careful not to clean too deeply as this will leave a negative bowl around your target emission - this is definitely not desirable.

To work out which features are sidelobes and which are not, it may help to look at the dirty beam with CGDISP or kview. For compact structures, the sidelobes in the residual image will have the same form as the beam.

We've already seen several types of images of the visibility dataset - dirty images, clean component images, and residual images. For further analysis or publication, the acceptable image is the clean-component image, convolved with a Gaussian to remove high-spatial frequencies, and with the residual image added in as well. This combination

is often called the "restored" image. To produce this image, use `RESTOR` again, but this time with `options = clean`:

```
Task:                restor
model                = cln.imap
map                  = ic5052.imap
beam                 = ic5052.beam
mode                 = clean
out                  = restor.imap
```

You can view this image with `CGDISP` or `kview`. Although it's an improvement on the dirty image, further cleaning still has to be done to improve the dynamic range. On source dynamic range is the peak flux density of the source, divided by the rms of the noise in the image *in the region around the source.*

4. Appendix: Invoking Miriad programs from the Unix shell

Miriad programs are invoked using the syntax:

```
taskname firstkey=firstargument secondkey=secondargument ...
```

Often, the Unix shell will recognise and try to interpret special characters eg. * or (). To protect these, you should enclose the key=value pairs in quotes like:

```
maths 'in=<vmap>+(0.00029*<imap>)' 'out=vmap_corr'
```