Spectral line tutorial - Miriad
CSIRO S&A Radio School 2023

Karen Lee-Waddell

September 27, 2023

1 Miriad

Miriad is a radio interferometry data reduction package that can be downloaded from the ATNF website: `https://www.atnf.csiro.au/computing/software/miriad/`. It is mostly used to process ATCA data.

1.1 General commands

→ miriad (start Miriad, run within the processing folder)

→ inp *taskname* (load a task and see the inputs, values can carry over between tasks)

→ help (details about loaded task)

→ *parameter* = <*value*> (set a parameter value)

→ unset *parameter* (clear a parameter value)

→ tput *taskname* (save parameter values)

→ tget *taskname* (reloads a task)

→ exit (exit Miriad, save parameter settings)

You can also put all the parameter setting into a single command, the examples of which have been provided in green.

2 Processing data

2.1 Import and split data

You can start by downloading raw data cubes from Australia Telescope Online Archive (`https://atoa.atnf.csiro.au/query.jsp`).

For this tutorial, we will be using data from the IMAGINE legacy survey (PI: A. Popping), ATCA project code C3157. NGC 1512 is a barred spiral galaxy with a very extended neutral hydrogen (HI) disk with a mass of $M_{HI} \sim 6 \times 10^9$ $M_\odot$.

Observations details:
- date = 2 Oct 2018
- ATCA configuration = 750C (with zoom mode)
- flux calibrator = 1934-638
- phase calibrator = 0438-436

IMAGINE used a redundant observing strategy to mitigate downtime due to CABB blocks dropping out. For this particular dataset, IF2 is the "better" dataset.

atlod converts the files into Miriad uv format using the following parameters:

→ inp atlod

→ in = *.C### (for this tutorial, project code = C3157)

→ out = rawdata.uv (new filename, which will include all datasets within the folder)

→ ifsel = # (select 4 for IF2 in zoom mode)

→ restfreq = 1.420405752 (for HI, in GHz)

→ options=bary,birdie,noauto,rfiflag (barycentric velocity frame, flag 'birdies', autocorrelations and known RFI channels)

→ go (to excute atlod, same command for all tasks)

command line: atlod in=*.C3157 out=rawdata.uv ifsel=4 restfreq=1.420405752 options=bary,birdie,noauto,rfiflag

Use uvindex to index the uv data:

→ vis = rawdata.uv (previously imported data)

→ log = rawdata.uvlog (to save a log file of the indexed data)

→ unset options (to clear previous option settings, same syntax to clear any parameter settings)

command line: uvindex vis=rawdata.uv log=rawdata.uvlog

Use uvflag to flag edge channels:

→ vis = rawdata.uv

→ edge = # (number of channels to flag at start and end of spectral window, use 40 for CABB data from ATCA)

→ flagval = flag

→ unset log (to clear previous option settings, same syntax to clear any parameter settings)

command line: uvflag vis=rawdata.uv edge=40 flagval=flag

uvsplit splits the data into calibrator and science targets:

→ vis = rawdata.uv

→ select = -shadow(d) (discards data affected by shadowing, d = diameter of dish + 15% = 25)

→ options = mosaic (since these data was observed in mosaic mode with proper naming convention)

command line: uvsplit vis=rawdata.uv select=-shadow(25) option=mosaic

The output files will be named using the format: *source_name.freq_details*

2.2 View visibility data

To look at spectra of a visibility dataset, use uvspec:

→ vis = *source.freq*

→ stokes = i (for Stokes-I polarization)

→ interval = ### (time averaging in minutes, choose a higher number – i.e. 1000 – to average all data)

→ device = /xw (to open a new display window, use *filename*/ps to write a postscript file)

→ nxy = 5,3 (plot all 15 baselines on the screen at once)

command line: uvspec vis=*source.freq* stokes=i interval=1000 device=/xw nxy=5,3

For an overall summary:

→ options = avall,nobase (average all baselines)

→ options = avall,nobase,ampscalar (average all baselines, plot amplitude using scalar averaging)

→ nxy = 1,1 (single output plot)

command line: blflag vis=*source.freq* stokes=i interval=1000 device=/xw options=avall,nobase,ampscalar nxy=1,1

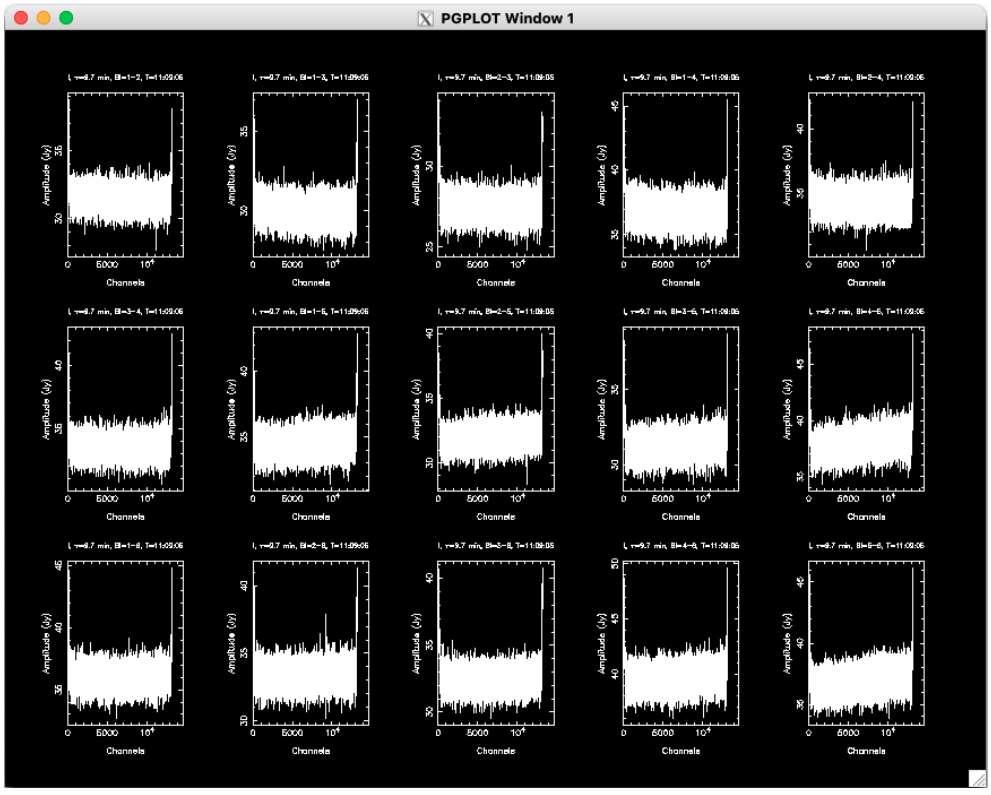

Figure 1: uvspec output of flux calibrator 1934-638 data, showing each baseline

2.3 Visually inspect and flag data

For the most part, you want to flag spurious signals in the bandpass and phase calibrators data. For this tutorial, flux calibrator = 1934-638 and phase calibrator = 0438-436.

Use either or both of these following tasks to manually flag "spikes" and/or outliers in the data. After flagging, inspect the data again using uvspec. Iterate the flagging and inspecting cycle as necessary.

Note: there are more automated/algorithmic methods used for flagging data, but sometimes its just more fun to do things manually (plus this dataset is fairly clean).

blflag allows for interactive flagging using the cursor:

→ vis = *source.freq*

→ device = /xw

→ axis = time,phase (for phase plot, unset for amplitude plot)

→ options = nobase (plot all baselines on one plot)

command line: blflag vis=*source.freq* device=/xw axis=time,phase options=nobase

Interaction commands: carriage return displays help menu with key commands.

```
----------------------------------------
Single key commands are
 Left-button  Delete nearest point
 Right-button Next baseline
 <CR>  Help
 ?     Help
 a     Delete nearest point
 c     Clear flagging of this baseline
 e     Exit, preserving edits
 h     Help, these messages
 p     Delete point in polygonal region
 q     Quit, discarding edits
 r     Redraw
 u     Unzoom
 x     Next baseline
 z     Zoom in
----------------------------------------
```

Figure 2: blflag commands

pgflag displays waterfall plots and allows for interactive flagging:

→ vis = *source.freq*

→ device = /xw

command line: pgflag vis=*source.freq* device=/xw

Interaction commands: `http://www.atnf.csiro.au/computing/software/miriad/doc/pgflag.html`

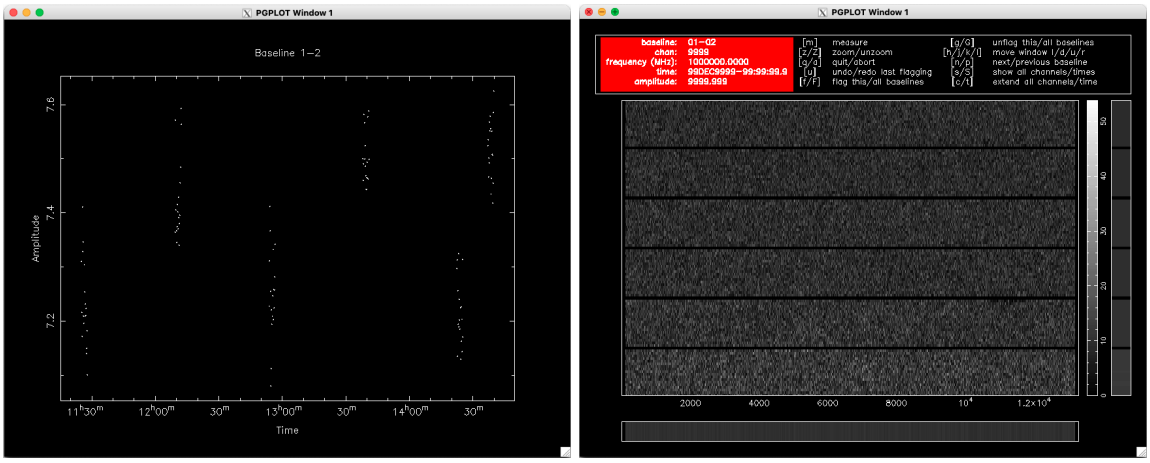

Figure 3: Left panel: blflag window showing baseline 1-2 of the phase calibrator 0438-436 data. The phase calibrator is observed at regular intervals. Right panel: pgflag window also showing baseline 1-2 of the phase calibrator 0438-436 data.

2.4 Bandpass calibration

Use mfcal for multifrequency bandpass calibration:

→ vis = *bp_cal.freq*

→ edge = # (number of channels to be dropped from the start and end of each spectral window, pick 10-20)

→ refant = # (reference antenna, pick a "good" antenna, default is 3)

→ interval = # (length of calibration solution interval, in minutes, default is 5)

→ options = interpolate (interpolates bandpass values for flagged channels)

→ unset stokes

command line: mfcal vis=*bp_cal.freq* edge=10 options=interpolate

If the source is a known calibrator, mfcal will use values from the database. Otherwise, specify the details using flux parameter.

Inspect with blflag and flag as required:

→ vis = *bp_cal.freq*

→ device = /xw

→ stokes = i

→ axis = real,imag

→ options = nobase

command line: blflag vis=*bp_cal.freq* device=/xw stokes=i axis=real,imag options=nobase

2.5 Phase calibration

Use gpcopy to copy the bandpass solutions to the phase calibrator:

→ vis = *bp_cal.freq*

→ out = *phase_cal.freq*

command line: gpcopy vis=*bp_cal.freq* out=*phase_cal.freq*

Use gpcal for phase calibration:

→ vis = *phase_cal.freq*

→ refant = # (reference antenna, pick a "good" antenna, default is 3)

→ interval = # (length of calibration solution interval, in minutes, default is 5)

→ options = xyvary,nopol (assume XY phase remains constant, do not solve for polarization leakages)

command line: gpcal vis=*phase_cal.freq* options=xyvary,nopol

Calibrator fluxes can be compared to the values in the ATCA database: `https://www.narrabri.atnf.csiro.au/calibrators/calibrator_database.html`

uvflux will determine some statistics about the visibilities:

→ vis = *phase_cal.freq*

→ stokes = i

command line: uvflux vis=*phase_cal.freq* stokes=i

Inspect with blflag and flag as required:

→ vis = *phase_cal.freq*

→ device = /xw

→ stokes = i

→ axis = real,imag

→ options = nobase

command line: blflag vis=*phase_cal.freq* device=/xw stokes=i axis=real,imag options=nobase

gpplt plots the gain corrections:

→ vis = *phase_cal.freq*

→ device = /xw

→ yaxis = phase

→ options = xygains

command line: gpplt vis=*phase_cal.freq* device=/xw yaxis=phase options=xygains

gpboot corrects the gain table by comparing amplitudes of the datasets:

→ vis = *phase_cal.freq*

→ cal = *bp_cal.freq*

command line: gpboot vis=*phase_cal.freq* cal=*bp_cal.freq*

2.6 Apply calibration solutions

Use gpcopy to copy the phase/gain solutions to the science data:

→ vis = *phase_cal.freq*

→ out = *science.freq*

command line: gpcopy vis=*phase_cal.freq* out=*science.freq*

Examine the spectral of the calibrated science data using uvspec and note which channels have HI:

→ vis = *science.freq*

→ stokes = i

→ interval = ###

→ options = avall,nobase

→ axis = felocity (x-axis is velocity, in standard optical convention)

→ device = /xw

→ nxy = 1,1

command line: uvspec vis=*science.freq* stokes=i interval=1000 options=avall,nobase axis=felocity device=/xw nxy=1,1

2.7 Subtract continuum in uv domain

Inspect with blflag and flag if required:

→ vis = *science.freq*

→ device = /xw

→ stokes = i

command line: blflag vis=*science.freq* device=/xw stokes=i

uvlin can subtract out the continuum:

→ vis = *science.freq*

→ out = *science*.line

→ chans = #,#,#,# (select line-free channels)

→ order = 1 (linear fit)

command line: uvlin vis=*science.freq* out=*science*.line chans=#,#,#,# order=1

uvlin can also make a continuum only dataset (optional):

→ vis = *science.freq*

→ out = *science*.cont

→ chans = #,#,#,# (select line-free channels)

→ order = 1

→ mode = continuum

command line: uvlin vis=*science.freq* out=*science*.cont chans=#,#,#,# order=1 mode=continuum

3 Imaging and data analysis

Visualisation applications such as CASAviewer (imview in CASA), CARTA, and kvis/Karma are useful to open and inspect the image cubes. Other visualisation tools could also be used but may require conversion to a different format (e.g. fits).

3.1 Making image cubes

invert make a 'dirty' image:

→ vis = *science*.line

→ map = *science*.dirtymap (output map name)

→ beam = *science*.beam (output beam, i.e. PSF)

→ imsize = # (chose number that is not a power of two and is large enough to image the entire primary beam)

→ cell = # (pixel size, should have 5 pixels across the synthesized beam size)

→ sup = 0 (natural weighting sidelobe suppression, better to vary robust parameter)

→ line = felocity,total#,start#,average#,step# (line type, number of channels/bins, starting channel/velocity, number of channels/velocity range to average together, step size. Note: units of values depend on the line type specified. To combine multi-epoch data, use velocity/felocity rather than channel)

→ select = *subset_options* (can specify a subset of the data, e.g. leave out antenna 6 with select = -antennae(6))

→ options = mosaic (if observations are in mosaic mode, also specify offset to set the image centre)

command line: invert vis=*science.line* map=*science*.dirtymap beam=*science*.beam imsize=400 cell=10 sup=0 line=felocity,150,600,4,4 options=mosaic

Tips: averaging channels (to about 4 km/s per channel) and/or pixels (ensuring there are at least 5 pixels across the beam minor axis) should improve signal-to-noise. Also try imaging with natural weighting (sup = 0) without and with antenna 6. Then try robust weighting (unset sup and robust = 0.5 or robust = 0) without and with antenna 6.

clean and mossdi extract 'clean' components, the latter is used for observations made in mosaic mode:

→ map = *science*.dirtymap

→ beam = *science*.beam

→ out = *science*.ccomp (clean component output)

→ gain = # (minor iteration loop gain, default is 0.1)

→ cutoff = 0.02 (clean cut-off, start with 5σ, vary as required. Invert will give theoretical RMS)

→ niters = 100000 (number of iterations, pick a high number)

command line: clean map=*science*.dirtymap beam=*science*.beam out=*science*.ccomp cutoff=0.02 niters=100000

command line: mossdi map=*science*.dirtymap beam=*science*.beam out=*science*.ccomp cutoff=0.02 niters=100000

restor makes a 'clean' image:

→ model = *science*.ccomp

→ beam = *science*.beam

→ map = *science*.dirtymap

→ out = *science*.cleanmap

command line: restor model=*science*.ccomp beam=*science*.beam map=*science*.dirtymap out=*science*.cleanmap

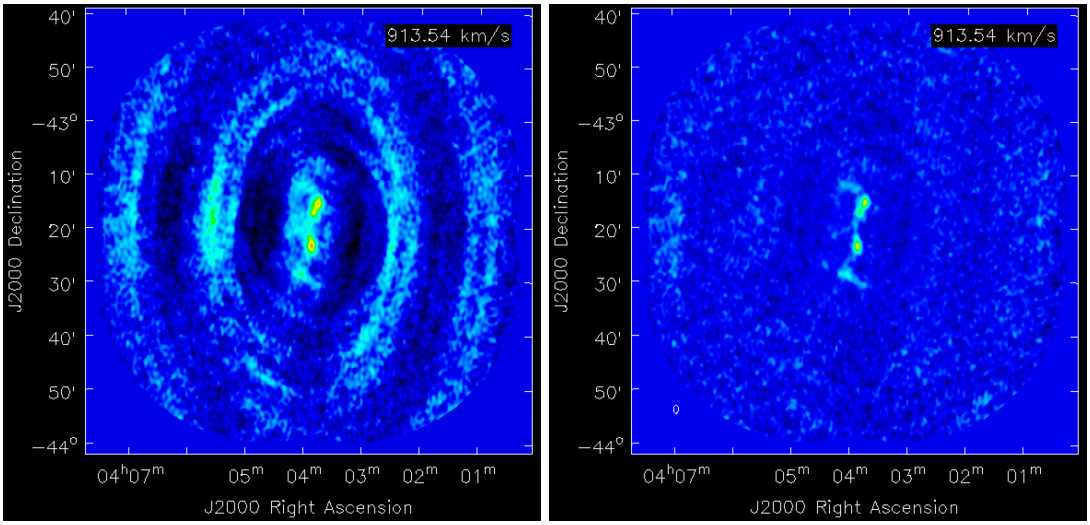

Figure 4: Left panel: single channel of 'dirty' cube. Right panel: same channel in 'cleaned' cube.

If the data is not a mosaic, linmos applies a primary beam correction:

→ in = *science*.cleanmap

→ out = *science*.cleanmap.pb

command line: linmos in=*science*.cleanmap out=*science*.cleanmap.pb

3.2 Moment and channel maps

moment produces moment maps:

→ in = *science*.cleanmap

→ out = *science*.mom0

→ mom = 0 (moment zero, total intensity map)

→ axis = 3

→ clip = -1000,0.003

command line: moment in=*science*.cleanmap out=*science*.mom0 mom=0 axis=3 clip=-1000,0.003

→ out = *science*.peakint

→ mom = -2 (peak intensity map)

command line: moment in=*science*.cleanmap out=*science*.peakint mom=-2 axis=3 clip=-1000,0.003

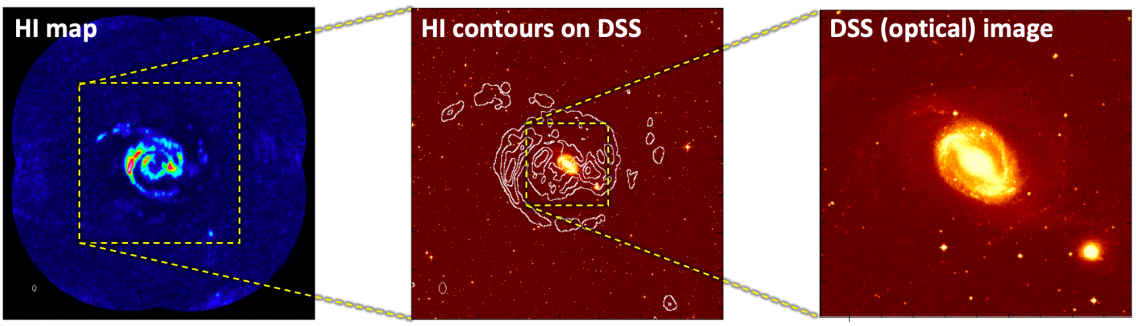

Figure 5: HI map and optical image of NGC 1512

cgdisp produces channel maps (i.e. subplots of individual or averaged channels):

→ in = *science*.cleanmap (can specify multiple files)

→ type = c (contour map)

→ region = boxes(#,#,#,#)(#,#) for $(x_{min},y_{min},x_{max},y_{max})(z_1,z_2)$

→ chan = #,# (two values, first = channel increment, second = number of channels to average)

→ cols1 = # (contour colours, 1 = foreground colour)

→ device = channel.ps/vcps (outputs a colour postscript file)

→ nxy = #,# (number of sub plots)

→ labtyp = hms,dms (spatial axes in RA, Dec)

→ beamtyp = b,l,2 (outline of synthesised beam in bottom left)

→ options = 3value,blacklab,nofirst (label each subplot, black labels, one set of x-axis labels,)

→ 3format = f5.0 (label formatting)

→ csize = 0.5,0.7,0.7,0.7 (character sizes)

command line: cgdisp in=*science*.cleanmap type=c region=boxes(100,100,400,400)(38,117) chan=4,4 cols1=1 device=channel.ps/vcps nxy=4,6 labtyp=hms,dms beamtyp=b,l,2 options=3value,nofirst,blacklab 3format=f5.0 csize=0.5,0.7,0.7,0.7

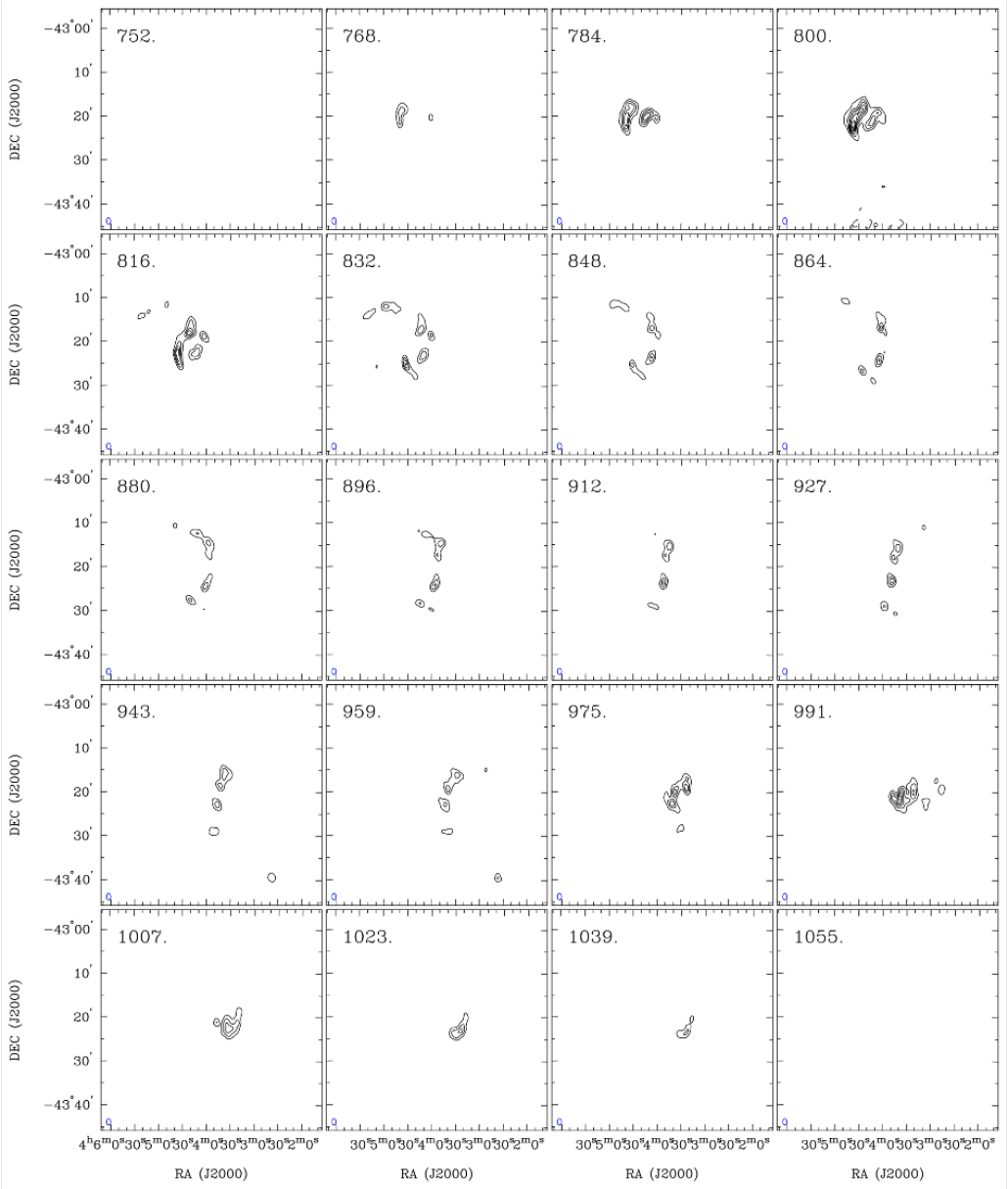

Figure 6: Channel maps of NGC 1512

3.3 Spectral analysis

mbspect plots spectral profile and measures fluxes:

→ in = *science*.cleanmap

→ coord = #,# (central coordinates of source, default is centre of the map)

→ width = #,# (dimension of box, in pixels, around the source. Must be odd numbers)

→ xaxis = felo ('optical' velocity)

→ yaxis = sum (summed and normalised values)

→ xrange = #,# (x-axis range for the plot)

→ order = 0 (order of polynomial fit)

→ options = measure (measure the spectral parameters from the plotted spectrum)

→ mask = #,#,... (x-axis range to be excluded from continuum fit, give pairs of numbers)

→ profile = #,# (x-axis range for profile measurements)

→ device = /xw

command line: mbspect in=*science*.cleanmap width=101,101 xaxis=felo yaxis=sum order=0 option=measure mask=750,1070 profile=760,1060 device=/xw

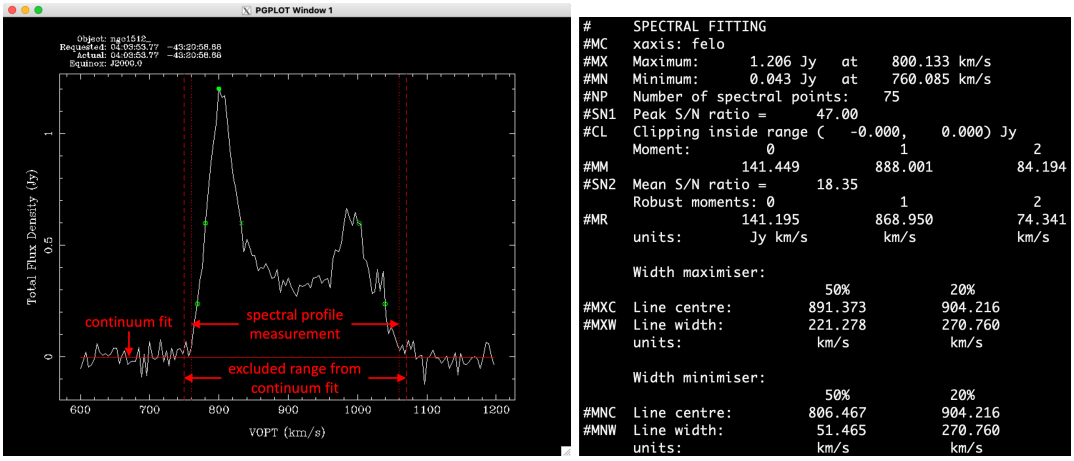

Figure 7: mbspect outputs. Left panel: spectrum with lines showing regions being fitted/measured. Right panel: values measured from spectral fitting.

3.4 Export to fits

Convert image made in Miriad to fits format with fits:

→ in = *imagename*

→ op = xyout

→ out = *imagename*.fits

command line: fits in=*imagename* op=xyout out=*imagename*.fits