Introduction to MIRIAD

Doug Friedel
What is MIRIAD
A collection of programs/tasks that use a common interface
Can be called from the command line or from inside the MIRIAD command shell
Can be used to:

- display, flag, calibrate, analyze u-v data
- generate images
- display, mask, analyze images
General task categories:

- Calibration
- Plotting
- Profile analysis
- u-v analysis
- Deconvolution
- Map manipulation
- Flagging
- Map combination
What is a MIRIAD File
MIRIAD files are directories

Two types of MIRIAD files: $u-v$ and image
u-v data
"streaming" data model
file contains:
  • header - listing of some variable names
  • history - ascii text of what has been done to the file
  • variable - listing of metadata and data type
  • visdata - actual u-v data
  • flags - binary flag for every channel in every baseline in every integration item files
    • calibration tables (gains, bandpass, leakage)
    • blfmask - blanking and flagging mask
each record contains:

- time stamp (JD)
- u,v(w) baseline lengths in ns
- baseline number
- complex data (one entry per channel)
u-v data
"streaming" data model
file contains:
  • header - listing of some variable names
  • history - ascii text of what has been done to the file
  • variable - listing of metadata and data type
  • visdata - actual u-v data
  • flags - binary flag for every channel in every baseline in every integration item files
  • calibration tables (gains, bandpass, leakage)
  • blfmask - blanking and flagging mask
Image data (beam, data cube) file contains:

- header - metadata for image
- history - same as with u-v data
- mostable - listing of mosaic parameters
- mask - flags for the image (one per pixel)
- image - actual image data (one value per pixel)
Using MIRIAD
Getting help
list all tasks by category

```
doc -t $MIRDOC/prog/*/doc
```

display documentation for a given task

```
doc <task>
<task> -k
```
There are several ways of executing MIRIAD commands:

From inside the miriad shell
- can see all arguments
- type `help` for documentation on current task
- good for iterating on a single task
- use `inp <task>` or `tget <task>` to go to a task
- `<keyword>=<value>` to set the input values
- type `go` to execute the task

From the command line
- all arguments on 1 line
- can use variables
- good for scripts
Most MIRIAD tasks share common key words

**u-v files**

- **vis** - input visibility file(s)
- **line** - selection criteria based on channels or velocity
  
  line=channel|velocity|width, <channel>, <start>, <width>, <step>
  
  - line=chan, 32, 1, 2, 2: select 32 output channels, starting at channel 1, averaging every 2 channels together
  
  - line=vel, 25, -50, 2, 1.5, 1.5: select 25 output channels, starting at -50 km/s, averaging every 1.5 km/s together

- **select** - selection criteria based on other parameters
  
  - time, ra, dec, antenna, window, source, uvrange, list, etc. (miref select will give the full list)
  
  - a "-" in front inverts the selection
  
  - window(2,3) all data from windows 2 & 3
  
  - source(ORIONKL) data from ORIONKL
  
  - ant(2,3)(5,6) all data but autocorrelations
  
  - time(02:25:15, 03:00:00) all data between these times (inclusive) in UT

- **options** - extra processing options
  
  - nopass: do not apply passband calibration
  
  - nocal: do not apply gain calibration
  
  - flagged: only work with flagged data

**image files**

- **in** - input image file(s)
- **region** - selection criteria for images
  
  - image, box, polygon: abspixel, relpixel, arcsec, kms
  
  - region=box(100, 120, 150, 170) (1, 5)
    
    - a box with corners at (100, 120) & (150, 170) pixels and planes 1-5
  
  - region=images(2, 15)
    
    - image planes 2-15
  
  - region=arcs, box(-10, -10, 10, 10)
    
    - a box 20" on a side centered on the reference pixel

- **options** - extra processing options
  
  - very variable by task
• **vis** - input visibility file(s)

• **line** - selection criteria based on channels or velocity

  `line=channel|velocity|wide,<nchan>,<start>,<width>,<step>`

  `line=chan,32,1,2,2` select 32 output channels, starting at channel 1, averaging every 2 channels together

  `line=vel,25,-50.2,1.5,1.5` select 25 output channels, starting at -50.2 km/s, averaging every 1.5 km/s together

• **select** - selection criteria based on other parameters

  time, ra, dec, antenna, window, source, uvrange, lst, etc. (mirhelp select will give the full list)
  multiple selections can be given, separated by a “,”
  a “-” in front inverts the selection

  `window(2,3)` all data from windows 2 & 3
  `source(ORIONKL)` data from ORIONKL
  `ant(2,3)(5,6)` data from baselines 2-5, 2-6, 3-5, & 3-6
  `-auto` all data but autocorrelations
  `time(02:25:15,03:00:00)` all data between these times (inclusive) in UT

• **options** - extra processing options

  `nopass` do not apply passband calibration
  `nocal` do not apply gain calibration
  `flagged` only work with flagged data
• **in** - input image file(s)
• **region** - selection criteria for images
  image, box, polygon     abspixel, relpixel, arcsec, kms
  region=box(100,120,150,170)(1,5)
    a box with corners at (100,120) & (150,170) pixels and planes 1-5
  region=images(2,15)
    image planes 2-15
  region=arcsec,box(-10,-10,10,10)
    a box 20" on a side centered on the reference pixel

• **options** - extra processing options

  very variable by task
Tasks for Inspecting Your Data
**listobs** - gives a good overview of the data that were taken

**uvindex** - gives detailed info of spectral setup

**uvlist** - gives window setup

**varlist & varplt** - plot header data
uvplt - plot actual data vs time, uvdistance, etc.

uvspec - plot raw u-v spectra

uvflag/csflag - flag data good or bad

uvbflag - flag data based on other reasons
Tasks for Calibrating Your Data
Passband Calibration

As the astronomical signals transit to the correlator a frequency dependent phase and amplitude shifts are induced on the signal. Most of this is due to the hardware (filters, fiber lengths, etc) However a small fraction is due to the atmosphere.

There are several ways to correct for this:
- Use an astronomical source with no spectral features (e.g. quasar)
- Use the internal noise source
- Use the auto-correlations

`mfcal` - calculates the passband correction
`gpcopy` - copy the gains from the calibrator to other objects
`uvcat` - apply the gains
Gain Calibration

**bootflux** - determine the flux of a calibrator

**selfcal** - calculates the amplitude and/or phase gains for your calibrator
Other Calibrations

Paired Antenna Calibration System (PACS)
- Phase calibration for long baseline observations
- Uses the 3.5m dishes to track phases of calibrators at 1cm throughout the observation
- Phase changes are bootstrapped to the observing frequency (3mm or 1mm)

Polarization calibration
- correct for instrumental affects on polarization data
Tasks for Imaging Your Data
Making Maps

invert - used to make the maps
  - produces a “dirty map” and the synthesized beam pattern
  - it has many options that affect how the map is produced:
    - pixel size in arcseconds
    - image size in pixels
    - weighting schemes, natural, uniform, and anything in between
    - selection parameters
    - line selection parameters
    - options (systemp, mosaic, double, ...)

![Image of mapping software interface]
Cleaning Maps

clean - for single pointing maps with objects near the phase center
mossdi - for mosaics, complex (not near phase center) or unknown source structure

Both tasks have similar inputs:
regions to clean (default is inner quarter)
way of controlling how deep to clean (gain, niters, cutoff)

restor is used to produce the final cleaned maps
Visualizing Maps

cgcurs & cgdisp - good for looking at maps plane by plane
imspec - look at spectra from a specific region

Non-MIRIAD tasks:
- ds9
- karma
- IMViz (http://lai.astro.illinois.edu/IMViz/)
More info can be found at: http://carma.astro.umd.edu/miriad/ including CARMA cookbook

MIRIAD tasks are written in fortran77 with C i/o routines
    Feel free to edit the tasks or create new tasks to suit your needs (many people have)
    To recompile just run $MIR/src/sys/bin/mir.prog <task>

The current MIRIAD distribution contains wrappers for python access to the underlying subroutines

Other wrapper types also exist (e.g. Ruby)

Source code for programs is in $MIRPROG
All CARMA data are run through the pipeline when they arrive in the archive.
The intent is to give the PI a "first look" at their data.
The pipeline can process:
  - single pointing
  - multipoint mosaic
  - spectral line
  - continuum
  - SZ data
Python master scripts calling miriad tasks or directly accessing raw data via python wrappers to subroutines
- Baseline solution

- Flagging (shadowing, systemp, bad gains, etc.)

- Bandpass correction
  - 125, 250 & 500 MHz windows are done with astronomical source
  - 62 MHz prefer astronomical source
  - All other modes done with noise source

- Bootflux

- Gain calibration

- Make maps:
  - Continuum
  - Spectral line widows
  - For Sunyaev-Zel'dovich data make short and long spacing maps

- Most processing is done in parallel
What you get:

- Calibrated u-v data from all objects
- Continuum image of source(s) and primary gain cal
- Data cubes from each window from source(s)
- Pipeline python files
- Csh script that can be re-run on your own machine that will reproduce what the pipeline did, and can be adjusted by the user to customize it for their data

All data taken this week will be run through a local copy of the pipeline so you can have a reference maps and scripts for your data reduction.