Miriad

Multichannel Image Reconstruction,
Image Analysis and Display

CARMA Cookbook

Peter Teuben

20 Jun 2008 - 16:12

See http://carma.astro.umd.edu/miriad

Contents

Chapter 1

Introduction

This manual, and other relevant information, is also available on the *MIRIAD* home page

`http://carma.astro.umd.edu/miriad`

and serves as a cookbook for reduction and analysis of CARMA data using the *MIRIAD* package. It is assumed that the reader has some familiarity with the underlying **Unix** operating system (be it Linux, Solaris or Darwin) and *MIRIAD* itself. A *wiki* page

`http://carma.astro.umd.edu/wiki/index.php/CARMA_Cookbook`

maintains links and other useful information for the cookbook. You will be able to download example scripts from there as well.

1.1 Users Guide

All general information, and many procedures also relevant for CARMA, can already be obtained from two existing Users Guides:

1.1.1 Miriad Users Guide

Throughout the cookbook the reader is assumed to be familiar with the (ATNF) *MIRIAD* Users Guide, in particular Chapters 2 (the `miriad` shell), 3 (plotting and the `device=` keyword), 4 (what *MIRIAD* datasets really are). Chapter 5 on visibility data is in particular important, it deals with the different types of *calibration tables*, and the `select=` keyword. Chapter 6 on image data is much shorter but also important to read. Chapter 10 on flagging is also important. However, note the ATNF version of *MIRIAD* is specialized for that array and different from the CARMA version, and here we will assume you are using the CARMA version of *MIRIAD*.

1.1.2 SMA Users Guide

The recently written SMA Users Guide contains lots of useful information as well, in a cookbook style, which can be complementary to the current CARMA cookbook and the ATNF *MIRIAD* Users Guide.

1.1.3 CARMA cookbook

Procedures specific to CARMA will be addressed in this cookbook. Most notably, the UV variables (Appendix C) in this version of the manual should be considered the appropriate ones for CARMA and other versions may show missing or conflicting information for the moment.

1.2 Future

This cookbook is currently a live document, and it will change rapidly over the coming months. Also be sure to be subscribed to the relevant mailing lists: `miriad-dev` for Miriad development issues, `carma_users` for CARMA observatory[1]. Miriad data versions (the filler changes from time to time). Developments around flagging and blanking, baseline and band dependent integration times, polarization *etc.*. Our **bugzilla** has a module for Miriad as well, though again, this is probably only useful for developers (though the developers should maintain a list of active bugs on a more user friendly webpage).

1.3 Links

- Wiki page for Miriad and CARMA Cookbook: `http://carma.astro.umd.edu/wiki/index.php/Miriad`
- Miriad bugzilla (part of CARMA bugzilla) at `http://www.mmarray.org/bugzilla`
-

1.4 Revision History

- 20-apr-2007: first draft
- 15-feb-2008: Draft for version 2 of this document

Acknowledgements

Stuart Vogel, Stephen White, Jin Koda, Joanna Brown etc.. And the fine crew of the first Miriad "Party" where much of this material was first fine tuned.

[1] subscription details on web....

Chapter 2

Work Flow

This Chapter guides you through the following basic steps of reducing your CARMA data. Each step has a seperate Section devoted to it.

1. Get your data from the CARMA Data Archive
2. Organize your working directory tree
3. Inspect data logs and quality
4. Correct data for obvious errors (flagging, baselines, etc.)
5. Calibrate phases and amplitudes
6. Mapping and Deconvolution

The PI will normally have received an email with a "`CARMA track finished`" subject line. It will have attached the observing script and logfile, and it will remind you how to download the data from the CARMA data archive. This Chapter guides you through the steps of getting your data, inspecting the data quality, and calibration of the data. Some comments on mapping and Deconvolution (standard procedures in Miriad and covered in depth elsewhere) are in place as well.

2.1 CARMA Data Retrieval

CARMA visibility data are normally multi-source MIRIAD datasets, where all data from a single track (a typical "8" hour observation) are in a single Miriad dataset.

Note that you will use the `carmaweb` username and need to get a password to gain access to CARMA data! The website `http://cedarflat.mmarray.org/observing` gives instructions on how to get this password.

2.1.1 Not at CARMA

The CARMA data archive at `http://carma-server.ncsa.uiuc.edu:8181/` provides a web interface to retrieve your data. Once you have located your data, there are two primary ways to download the data to your personal computer. The first is through a Java Web Start (jsp) application. Once launched (you may have to teach your browser where `javaws`[1] is located) and once the list of datasets has been displayed

[1] javaws is part of the Java Development Kit (JDK) and if not present, you may have to install it

in DaRT, click on the Download button in that java application to start the transfer. The second method would be to right click on the dataset names and use your browser to download the data. Notice that these file are gnuzip compressed tar files, and need to be un-tarred to become a real *MIRIAD* dataset (i.e. a directory):

```
  % tar zxf cx012.SS433.2006sep07.1.miriad.tar.gz
  % itemize in=cx012.SS433.2006sep07.1.miriad
Itemize: Version 22-jun-02
  obstype  = crosscorrelation
  nwcorr   = 573768
  ncorr    = 8606520
  vislen   = 48204056
  flags      (integer data, 277630 elements)
  visdata    (binary data, 48204052 elements)
  wflags     (integer data, 18509 elements)
  vartable   (text data, 632 elements)
  history    (text data, 1095356 elements)
```

The drawback of this download scheme is that you initially need about twice the diskspace. You can also use the streaming capabilities of programs like `wget` or `curl` to transfer and un-tar on the fly, if you know how to construct the URL from the dataset names you saw on that Data Archive page:

```
  % set base=http://carma-server.ncsa.uiuc.edu:8181/data/
  % set data=cx012.SS433.2006sep07.1.miriad.tar.gz
  % wget -O - $base/$data | tar zxf -
or
  % curl $base/$data | tar zxf -
```

This procedure will only work for recent data that are in the cache, for example all the "Tracks Transferred within The Last Two Days" listed on the Quick Searches page fall under this category.

2.1.2 At CARMA

At CARMA (and OVRO) there is limited bandwith to the outside world, and CARMA data should probably be directly copied via one of the `cedarflat` machines on `/misc/sdp/sciencedata/`, or in case of older data, the directory `/misc/sdp/archive_sciencedata/` should contain data older than about a month. Eventually those data will also disappear and can then only be retrieved via the Data Archive.

Notice one subtle naming convention: currently the site uses `.mir` names, where the data archive uses `.miriad`! The data archive returns gzip compressed tar files, whereas the site only uses miriad data (i.e. directories).

2.2 Organizing your CARMA Data Tree

Once you start downloading your data, the question immediately arises where to put these data and how to organize your data. Recall each observation results in a single multi-source MIRIAD dataset containing all your sources and calibrators. We recommend that you place each of these datasets in a separate directory, since your data reduction scripts likely will look very similar and this can result in a more efficient way to organize your reduced files.

```
    MyProject / Day1 / cx002.foo.1/visdata
                                  flags
                                  ...

              / Day2 / cx002.foo.2/visdata
```

...

...

These "Day" directories are also a good place to put your observing script and logfile, as it was emailed back to you after the track was finished. With this scheme, after calibration is done in each "Day" directory, you might wind up with a script that combines all these data in the following way:

```
% invert vis=Day1/n1234c,Day2/n1234c,Day3/n1234c map=n1234.mp beam=n1234.bm
...
```

where each "Day" directory is assumed to have its cleaned and calibrated MIRIAD dataset named `n1234c`.

2.3 Quality Check

After your data was taken, a *quality* script ran at the observatory inspecting your data and giving it a grade. The output of `quality` is normally also inspected by the resident observer(s). Once you have downloaded your data from the archive, it is important for you to first check that all the data that `quality` has reported, is actually also present in your dataset (most notably check the full timerange and all the sources reported). A few Miriad programs are available for this, described in the next section, though in theory you could also run `quality` yourself. Directions for running quality are located in the Observer Help Pages.

You should be able to find your quality output from the CARMA webpage[2] by going to "Observing with CARMA" and following "Obtaining and reducing your data" to "Quality output". Copy the output into your directory and inspect it.

2.4 Data Inspection

There are several ways to get a useful summary of what is in your CARMA multi-source dataset. MIRIAD programs `listobs`, `uvindex` and `uvlist` all have options to deal with this. As stressed before, it is a good idea to double check if your dataset matches the one that your quality report saw.[3]

2.4.1 listobs

`listobs` gives an overall summary of the data: antenna positions w.r.t. the center of the array, baselines, sources observed, frequency setup, and chronology of the observation. An example follows:

```
  % listobs vis=cx012.SS433.2006sep07.1.miriad
Opening File: cx012.SS433.2006sep07.1.miriad
                SUMMARY OF OBSERVATIONS
------------------------------------------------------------------------------
------------------------------------------------------------------------------
Input file: cx012.SS433.2006sep07.1.miriad
------------------------------------------------------------------------------
         Antenna and Baseline Information
         --------------------------------
            Antenna Locations (in nsec)            Antenna Locations (in m)
                 X           Y           Z             E           N           U
Antenna  1:   -29.9213    -98.3795     41.9036       -29.493      15.429       0.472
```

[2] See: `http://cedarflat.mmarray.org/observing/quality/`

[3] On occasion a glitch in the data transfer process has resulted in incomplete datasets

```
Antenna  2:   -45.4240    117.2736     60.7055        35.158      22.729       0.188
Antenna  4:   -66.6784     21.1092     89.9173         6.328      33.557       0.423
Antenna  5:   -87.5850   -149.6982    121.2388       -44.878      44.825       1.123
Antenna  6:  -123.4471    -95.6647    168.5641       -28.680      62.626       1.162
Antenna  7:    64.6595     77.8763    -87.9020        23.347     -32.710      -0.538
Antenna  8:    43.7087     42.8510    -59.7407        12.846     -22.188      -0.422
Antenna  9:    35.7017    -64.6192    -48.4809       -19.372     -18.048      -0.287
Antenna 10:    92.9788    -34.3857   -125.6435       -10.309     -46.855      -0.636
Antenna 12:    89.1192   -145.3247   -119.1938       -43.567     -44.615      -0.386
Antenna 13:    17.1266     40.2879    -24.0221        12.078      -8.840      -0.277
Antenna 14:    -8.3007     65.6849      9.9469        19.692       3.880      -0.174
Antenna 15:     3.0289    142.9872     -5.0120        42.866      -1.746      -0.188
--------------------------------------------------------------------------------
           Baselines in Wavelengths
           ------------------------
      for Decl = 0 deg. Source at Transit
                 U           V           W          UVdistance
Bsln   1- 2:  -19941.18    -1738.59     1433.51    20016.82
Bsln   1- 4:  -11048.97    -4439.77     3398.88    11907.61
Bsln   1- 5:    4745.37    -7336.03     5332.09     8737.04
...
Bsln  13-15:   -9496.48    -1757.84     1303.60     9657.80
Bsln  14-15:   -7148.05     1383.23    -1047.63     7280.65
--------------------------------------------------------------------------------
            Observed Sources Coordinates and Corr Freqs
Source          RA          Decl          Vlsr             Corfs in MHz
MARS               12 00 24.86 0 47 14.93      0.00E+00     0.0
3C273              12 29 06.70 2 03 08.60      0.00E+00     0.0
1830+063           18 30 05.94 6 19 16.00      0.00E+00     0.0
SS433              19 11 49.56 4 58 57.60      0.00E+00     0.0
noise              19 11 49.56 4 58 57.60      0.00E+00     0.0
--------------------------------------------------------------------------------
                          Frequency Set-up
   Source: SS433                      UT: 235551              LST: 151057
Line Code: unknown    Rest Freq:   92.4688 GHz     IF Freq:     0.000 MHz
Velo Code: VELO-LSR   Anten Vel:      0.00 km/s   First LO:   95.0000 GHz
--------------------------------------------------------------------------------
Source               UT      LST     Dur Elev       Sys Temps (K)
                    hhmmss   hhmmss    min deg     1    2    4    5    6    7    8    9   10   12   13   14   15
MARS               232853.5 144355.2 10.0  49.  340  384  295  295  339  490  543  531  339  446  256  352  627
3C273              233938.0 145441.5 10.0  54.  332  376  290  289  331  473  541  520  323  434  247  339  611
1830+063           235255.5 150801.2  2.0  39.  374  432  346  342  394  539  556  591  370  484  292  393  694
SS433              235551.0 151057.1  6.7  30.  437  491  406  394  450  615  689  674  458  581  361  470  776
noise              000242.0 151749.3  0.0  31.  427  482  390  383  443  641  695  696  446  590  381  489  808
1830+063           000350.0 151857.5  2.0  42.  392  430  353  351  404  542  557  596  386  495  297  398  716
SS433              000644.5 152152.4  6.7  32.  432  486  413  395  454  621  655  653  439  558  351  459  800
noise              001335.5 152844.6  0.0  34.  396  448  358  353  408  601  647  649  410  559  351  458  774
1830+063           001439.5 152948.7  2.0  44.  339  385  298  295  340  500  562  539  335  453  260  356  637
...

1830+063           072914.5 224535.1  2.0  26.  427  488  400  387  444  635  690  676  442  582  359  463  792
SS433              073207.0 224828.1  6.7  36.  380  437  349  345  389  581  624  620  397  524  310  415  713
1830+063           073938.5 225600.8  2.0  23.  460  517  428  414  465  662  728  728  476  608  382  494  832
--------------------------------------------------------------------------------
```

Here, inspect if the antenna positions are indeed the ones obtained from those from the latest baseline solutions. See also Section D.4.6. Also pay attention to the system temperatures, near rising and setting you might see some increased values, and some antennas are better than others, but there should be no outliers.

2.4.2 uvindex

`uvindex` provides a useful display showing the track length, LO settings, etc. The output of `uvindex` should be compared with the log sent by e-mail and the actual data length. Sometimes there is a failure

in filling the data properly, and the data archive center should be contacted. [4] An example output of `uvindex` follows:

```
  % uvindex vis=cx012.SS433.2006aug25.1.miriad
UVINDEX: version 14-apr-06

Summary listing for data-set cx012.SS433.2006aug25.1.miriad

       Time        Source        Antennas Spectral Wideband  Freq  Record
                    Name                 Channels Channels Config   No.

06AUG25:23:26:45.5 MARS             15        90        6      1       1
06AUG25:23:37:39.5 3C273            15        90        6      1    1981
06AUG25:23:50:14.0 1830+063         15        90        6      1    3961
06AUG25:23:53:33.0 SS433            15        90        6      1    4357
06AUG26:00:00:15.0 noise            15        90        6      1    5677
06AUG26:00:01:26.5 1830+063         15        90        6      1    5743
06AUG26:00:04:17.5 SS433            15        90        6      1    6139
06AUG26:00:10:59.5 noise            15        90        6      1    7459
06AUG26:00:12:09.5 1830+063         15        90        6      1    7525
06AUG26:00:15:04.0 SS433            15        90        6      1    7921
06AUG26:00:21:46.0 noise            15        90        6      1    9241
...
06AUG26:03:03:58.0 1830+063         15        90        6      1   36037
06AUG26:03:06:49.0 SS433            15        90        6      1   36433
06AUG26:03:13:31.0 noise            15        90        6      1   37753
06AUG26:03:14:42.0 1830+063         15        90        6      1   37819
06AUG26:03:17:32.5 SS433            15        90        6      1   38215
06AUG26:03:24:14.5 noise            15        90        6      1   39535
06AUG26:03:25:29.5 1830+063         15        90        6      1   39601
06AUG26:03:27:09.5 Total number of records                         39996

------------------------------------------------

Total observing time is  3.27 hours

The input data-set contains the following frequency configurations:

Frequency Configuration 1
  Spectral Channels  Freq(chan=1)  Increment
               15      92.46875    -0.031250 GHz
               15      93.46875    -0.031250 GHz
               15      92.96875    -0.031250 GHz
               15      97.53125     0.031250 GHz
               15      96.53125     0.031250 GHz
               15      97.03125     0.031250 GHz
  Wideband Channels  Frequency     Bandwidth
                      92.23438    -0.468750 GHz
                      93.23438    -0.468750 GHz
                      92.73438    -0.468750 GHz
                      97.76562     0.468750 GHz
                      96.76562     0.468750 GHz
                      97.26562     0.468750 GHz

------------------------------------------------

The input data-set contains the following polarizations:
There were 39996 records of polarization RR

------------------------------------------------

The input data-set contains the following pointings:
 Source                   RA            DEC             dra(arcsec) ddec(arcsec)
1830+063               18:30:05.94   6:19:16.00             0.00         0.00
3C273                  12:29:06.70   2:03:08.60             0.00         0.00
MARS                   11:29:54.84   4:11:08.34             0.00         0.00
```

[4] currently: Lisa Xu, lisaxu@ncsa.uiuc.edu

```
SS433                  19:11:49.56   4:58:57.60             0.00         0.00
noise                  19:11:49.56   4:58:57.60             0.00         0.00

-----------------------------------------------

The input data contain the following AzEl offsets
   Date             vis# ant   dAz   dEl (ArcMin)

06AUG25:23:26:45.5      1   1  0.00  0.00
-----------------------------------------------
```

2.4.3 uvlist

A useful listing of the spectral windows can be obtained with `uvlist`

```
% uvlist vis=cx012.SS433.2006sep07.1.miriad options=spectra
 rest frequency     :  92.46875  92.46875  92.46875  92.46875  92.46875  92.46875
 starting channel   :          1        16        31        46        61        76
 number of channels :         15        15        15        15        15        15
 starting frequency :  92.46875  93.46875  92.96875  97.53125  96.53125  97.03125
 frequency interval :  -0.03125  -0.03125  -0.03125   0.03125   0.03125   0.03125
 starting velocity  :     0.000 -3242.095 -1621.047-16413.105-13171.010-14792.058
 ending velocity    :  1418.416 -1823.678  -202.631-17831.522-14589.427-16210.474
 velocity interval  :   101.315   101.315   101.315  -101.315  -101.315  -101.315
```

where you can see the 3 lower sideband windows and 3 upper sideband windows. Notice the rest frequency in this example appear a little odd, being identical in all windows. See also Section 2.5.3 for a discussion on this. Note that `uvlist` only displays the first selected frequency setting. If your source and calibrator have a different setup, use `select=source()` to look at the appropriate setting for each object.

2.4.4 uvflag - initial check of flagged data

You should inspect how much data was flagged by the online system. As of March 2007, blanking has been enabled at CARMA, and depending on conditions and the threshold setting chosen in the observing script user-defined parameters, one can easily wind up with too much flagged data. Unflagging should of course be done with caution. Currently the default threshold is 20%, i.e. if more than 20% of the (0.5 second) frames of an integration are blanked, that integration itself is flagged.

```
% uvflag vis=cx012.SS433.2006sep07.1.miriad options=noapply flagval=flag
...
Total number of records selected: 95628; out of 95628 records
Antennas used: 1,2,4,5,6,7,8,9,10,12,13,14,15
Counts of correlations within selected channels
channel
Good:        5894250
Bad:         2712270
wide
Good:         392950
Bad:          180818
```

Just for the record, this is a *very* high fraction of flagged data. Normally you might see 0.1-1%. If you have a high percentage of flagged data, you might want to look through the Nightly Report or Observing Logs to discover the cause - antenna out-of-array, Rx problem, etc. If you discover serious data problems not accounted for, you should inform the observers - `obs@mmarray.org` - who can investigate.

2.4.5 Visual: uvplt, uvspec, varplt, uvimage, closure

Probably the most important thing to remember at various stages of your calibration is careful and consistent data inspection. After all calibration is done, your phases and/or amplitudes should be flat as function of frequency and/or time. Use `uvflag` and friends where needed to edit out discrepant data that could throw of calibration routines.

For the calibrator(s), inspect the run of phase and amplitude as a function of time and channel. In this example we are using the "sma" flavor of `uvplt` and `uvspec`:

```
% smauvplt  vis=cx012.SS433.2006sep07.1.miriad select="source(3c273)" axis=time,phase interval=3
% smauvplt  vis=cx012.SS433.2006sep07.1.miriad select="source(3c273)" axis=time,amp   interval=3

% smauvspec vis=cx012.SS433.2006sep07.1.miriad select="source(3c273)" axis=chan,phase interval=999
% smauvspec vis=cx012.SS433.2006sep07.1.miriad select="source(3c273)" axis=chan,amp   interval=999
```

Recall that "`mirhelp taskname`" will give you a full description of other options.

To inspect the amplitudes in a different manner, construct a 3D cube from the visibility data and view this with any of the FITS or MIRIAD image viewers that are available. Here is an example using `ds9`:

```
% ds9 &
% uvimage vis=cx012.SS433.2006sep07.1.miriad out=cube1 select="-source(noise),-auto"
% histo in=cube1
% mirds9 cube1
```

Another useful program

```
%
% closure vis=cx012.SS433.2006sep07.1.miriad select="source(3c273),-auto" device=/xs
```

To inspect a wide range of uv variables use `varplt` (a list can be found in Appendix C):

```
% varplt vis=cx012.SS433.2006sep07.1.miriad device=/xs yaxis=systemp nxy=5,3 yrange=0,2000 options=compress
```

shows C3 and C11 are not online. Autoscaling showed C2 has a bad point. But overall something bad happened around 5h UT. See Figure 2.1

2.5 Initial Data Correction

2.5.1 Archive based corrections

The CARMA Data Archive will typically re-fill data from its basic constituents (the *visbrick* and the *monitor points*) whenever the data is requested. This could mean that the data used by the `quality` script might be different from that obtained from the Data Archive.

You can save a checksum of your data and/or use the version of the data that is stored inside the visibility data. That way you will be able to decide if your data reduction will have to be redone.

```
% uvlist vis=cx012.SS433.2006sep07.1.miriad options=var,full | grep version
```

```
UVLIST: version  4-may-06
version :0.1.2

% mdsum cx012.SS433.2006sep07.1.miriad
518864276e75f081e68156fbf3ac12a3  cx012.SS433.2006sep07.1.miriad.tar.gz
```

Appendix D lists the various problems that could have occured with your data at different stages of the commissioning of CARMA in 2006/7. Especially if you are re-calibrating your data after some new insight, it makes sense to check if you should re-fetch the data.

2.5.2 Baseline correction

You should always check if you need to (re)apply baseline corrections[5] Although your data may come with initially pretty decent baseline solutions, often after a few weeks in a new array configuration improved baselines are available. In the first few days up to several weeks after a move, baselines can settle and may need to be re-applied from the newly computed ones. Normally these are stored in a small ascii table with equatorial values in nanoseconds. (*cf.* `uvgen baseunit=1`. Antpos datafiles can be found[6] at `http://cedarflat.mmarray.org/observing/baseline/`, as well in your local MIRIAD distribution in `$MIRCAT/baselines/carma`.. To apply a new baseline, apply the program `uvedit` to your multisource data set. Be sure to apply the new baseline to all sources:

```
uvedit vis=xxx.mir out=yyy.mir apfile=$MIRCAT/baselines/carma/antpos.070115
```

In rare cases, a new and better solution is found a month or so after your data were taken. Check the status of the baseline solution on the above mentioned web page. It is a good idea to apply an appropriate solution if you are not sure which solution has been applied to your data. No harm is done if you apply a solution that has already been applied.

Notice that for data taken during a move (which can take several days and the array will be in some hybrid configuration) an antpos file will be available for each day. Please check the time validity carefully, either by filename, or comments in the file.

Errors due to baselines can be seen as slopes in phase vs. time. See Figure ...

2.5.3 Rest Frequency (bugzilla 409)

Certainly during the initial campaigns, CARMA data were written with a rest frequency equal to the starting frequency in the first window of the LSB. This is most likely wrong for your data. Look again at the output of `uvlist`:

```
% uvlist vis=xxx.mir options=spec

rest frequency     : 100.27057 100.27057 100.27057 100.27057 100.27057 100.27057
starting channel   :         1        16        31        46        61        76
number of channels :        15        15        15        15        15        15
starting frequency : 100.27057 100.73054 101.19050 104.33300 103.87304 103.41307
frequency interval :  -0.03125  -0.03125  -0.03125   0.03125   0.03125   0.03125
starting velocity  :   -23.654 -1398.978 -2774.302-12170.599-10795.275 -9419.951
ending velocity    :  1284.502   -90.822 -1466.146-13478.755-12103.431-10728.107
velocity interval  :    93.432    93.432    93.432   -93.432   -93.432   -93.432
```

To fix this, you can set the `restfreq` variable to the (in this case CO 1-0) line you are interested in:

[5] all data prior to 22-jan-2008 should always be corrected
[6] At CARMA, also `/array/rt/baselineSolutions/antpos.YYMMDD`

```
% uvputhd in=xxx.mir hdvar=restfreq varval=115.271203 out=yyy.mir
```

The drawback of this procedure is that the uv variable is now "promoted" to a (miriad) header variable, and in the process loosing any potential time variability as well as (in this case 6) dimensionality.

TODO:explain difference between puthd and uvputhd

2.5.4 Linelength Correction

The linelength system monitors changes in the delays through the optical fibers to the antennas. The delays vary as the fibers change temperature. The delay variations are small, typically less than 0.05 nsec on time scales of hours, but they are enough to cause significant phase drifts of the local oscillators on the receivers. Since these changes are measured accurately by the linelength system, the corrections should be applied.

Phase corrections from the linelength system are stored in the Miriad uv variable `phasem1`, which is an antenna based variable.

To apply the linelength corrections, use the Miriad program `linecal`, which writes an antenna based calibration table in the dataset that can be applied. However, don't expect perfection - the linelength system cannot correct for differences in the thermal expansion of the antennas (particularly BIMA vs OVRO) or for changes in the temperature of the phaselock electronics. Schematically we will do:

```
linecal  vis=$data
gpplt    vis=$data  yaxis=phase nxy=5,3 device=/xs options=wrap
uvcat    vis=$data    out=$data.lc
```

Here the `gpplt` commands displays the actual phase corrections that are going to applied in `uvcat`. You might see many wraps, but remember it is only the differences in phases that matter, these are antenna based phases.

TODO: careful with select=-source(noise)

2.5.5 Other UV variables

Some data bugs cannot be fixed by refilling the data from the archive. For example at some point in the past the latitude was erroneously set to 0 (latitude=0) In this case programs such as `puthd` will work fine for variables that do not depend on time. In the first example we see how to fix the latitude (stored as 0 in the SS443 dataset) such that the ENU coordinates were printed correctly:

```
% puthd in=cx012.SS433.2006sep07.1.miriad/latitud value=0.6506654009 type=double
```

Developers and observers typically file these problems as bugs in our bugzilla. This particular bug was filed as bug # xxx and was caused by using multiple sub-arrays and one subarray polluting the data in another.

TODO: check on the uvedit problem with missing LO2.

2.5.6 Data Flagging and Editing

Chapter 10 in the Miriad Users Guide has an extensive discussion on flagging your visibility data. The two important programs that allow you to interactively flag are `uvflag` and `blflag`.

Programs such as `uvplt` and `varplt` can be used to inspect data and decide what baselines, antennae, time-ranges etc. need to be flagged. Another potentially useful way is a relatively new program `uvimage` which creates a Miriad image cube out of a visibility dataset. This 3 dimensional dataset can be viewed with programs like `ds9` or `karma`'s `kvis`, and guide you how to flag the data using `uvflag`. It is possible to come up with a procedure that ties keystrokes in `ds9` to the creation of a batch script that runs `uvflag` afterwards, and this is a likely change in upcoming versions of MIRIAD.

```
% uvimage vis=cx012.SS433.2006sep07.1.miriad out=visbrick1
UVIMAGE: version 22-dec-2006
Mapping amp
### Informational:  Datatype is complex
 Nvis= 95628 Nant= 13
 Nchan= 90 Nbl= 78 Ntime= 1226 Space used:  8606520 /  17432576 =   49.370327%
 number of records read=  95628

% mirds9 visbrick1
```

The most useful output mode is amplitudes (the default) where the cube will be constructed with channels along the X axis, baselines along Y and time along Z. The X axis is represented in `ds9` by different planes in `ds9`). As you move the *Data Cube* slider you will see different channel-baseline images of the visibility amplitudes at different times. Look for a change in noise, regions of pure 0s, vertical spikes (a.k.a. birdies), horizontal spikes (bad baselines or antennae). These will potentially all have to be flagged. Overall noise increase that is the result of a higher system temperature will be accounted for though (see `invert`).

2.5.7 Flagging Birdies and End Channels

An example of a birdie (often a antenna based single channel with high amplitudes) can be flagged easily with `uvflag` using `line=chan,`*n,start,width,step*:

```
#Birdies
 uvflag vis=$cfile "select=ant(1)" line=chan,1,32,1,1  flagval=flag
 uvflag vis=$cfile "select=ant(1)" line=chan,1,95,1,1  flagval=flag
 uvflag vis=$cfile "select=ant(1)" line=chan,1,158,1,1 flagval=flag
 uvflag vis=$cfile "select=ant(7)" line=chan,5,4,1,1   flagval=flag

#End channels
 uvflag vis=$cfile  line=chan,1,4,1,1   flagval=flag
 uvflag vis=$cfile  line=chan,1,186,1,1 flagval=flag
```

After this you should recompute the wide band averages using `uvwide`., if you plan to use them.

2.5.8 Flagging using tvflag

The interactive flagging program `tvflag` must be run on an 8-bit display, or PseudoColor display. Most modern desktops are so color rich, they cannot be effectively run on an 8-bit display, though `twm` and `fvwm` can. For example, on Linux you can start a second X session from another console (e.g. ctrl-alt-F2):

```
% startx -- -depth 8 :1
```

or use VNC:

```
% vncserver :1 -depth 8 -cc 3 -geometry 1024x768
% vncpasswd
% vncviewer :1

% xmtv &
% tvinit server=xmtv@localhost
% tvflag vis=vis0 server=xmtv@localhost

% vncserver -kill :1
```

2.5.9 Flagging based on tracking errors

The `axisrms` UV variable holds the tracking error (in arcsec, in Az and El) for each antenna in the array. It can be useful to automatically flag data when the tracking is above a certain error, or even antennae based (e.g. allow OVRO to have a smaller tolerance than the BIMA antennae).

```
% varplt vis=c0048.umon.1.miriad device=/xs yaxis=axisrms options=overlay yrange=0,100

% uvflag vis=c0048.umon.1.miriad 'select=-pointing(0,5)' flagval=flag options=noapply
```

The exact amount (5 arcsec in this example) is left to your own judgement, and you should probably also base this on the inspection of the graphical output of `varplt`. But in case you were wondering, the recommended value is 5.

2.6 Calibration

2.6.1 Passband Calibration

When a strong passband calibrator is available it can be used using `mfcal` to correct the passband of your others sources (not just the target source, but also for example the phase and amplitude calibrator). Choose a small interval to construct the antenna based passband, and inspected the solutions with `gpplt`:

```
% mfcal vis=pbcal.mir interval=0.5 refant=9

% gpplt vis=pbcal.mir device=/xs options=passband  yaxis=amp    nxy=5,3
% gpplt vis=pbcal.mir device=/xs options=passband  yaxis=phase  nxy=5,3
% gpplt vis=pbcal.mir device=/xs options=gains
```

The passband gains can then be copied to your other sources, for subsequent calibration and/or mapping:

```
% gpcopy vis=pbcal.mir   out=phasecal.mir options=nocal
% gpcopy vis=source1.mir out=source2.mir  options=nocal
```

2.6.2 Simple single calibrator

When a calibrator is strong enough in the same window as the source is observed, we can simply determine a selfcal solution[7] for the calibrator and apply this to the source:

Here is an annoted section of C-shell code exemplifying this:

[7] cf. also the gmakes/gfiddle/gapply approach for BIMA data

```
set vis=cx011.abaur_co.2006nov21.1.miriad

# check phase in W2 (narrow) and W3 (wide)
# TODO: lingo wwong here: W2/W3 vs. p,A
smauvplt vis=$vis device=/xs axis=time,phase line=wide,1,3 "select=-source(abaur)"
smauvplt vis=$vis device=/xs axis=time,amp   line=wide,1,3 "select=-source(abaur)"

# check bandpass
uvspec vis=$vis device=/xs "select=-auto,source(3c111)" axis=chan,amp interval=999
uvspec vis=$vis device=/xs "select=-auto,source(3c111)" axis=chan,pha interval=999

uvspec vis=$vis device=/xs "select=-auto,source(0530+135)" axis=chan,amp interval=999
uvspec vis=$vis device=/xs "select=-auto,source(0530+135)" axis=chan,pha interval=999

# use W5 , the narrow band in this case
rm -rf 0530+135
uvcat vis=$vis "select=-auto,source(0530+135)"  out=0530+135
selfcal vis=0530+135 refant=5 interval=5 line=wide,1,5,1 options=amp,apriori,noscale flux=4.6
gpplt vis=0530+135 device=1/xs yaxis=amp nxy=5,3 yrange=0,3
gpplt vis=0530+135 device=2/xs yaxis=pha nxy=5,3 yrange=-180,180

rm -rf abaur
uvcat vis=$vis "select=-auto,source(abaur),win(5)"  out=abaur
puthd in=abaur/restfreq type=double value=115.271203
gpcopy vis=0530+135 out=abaur
# copyhd in=0530+135 out=abaur items=gains,ngains,nsols,interval
```

2.6.3 Autocorrelation

Auto-correlations are handled by the datafiller as of January 31, 2007 (see also Appendix D). Visibility data auto-correlations are stored as baselines with the same antenna pair, and show up before the cross-correlations.

```
  % uvlist vis=c0048.umon.1.miriad recnum=20 line=wide,3
  ...
 Vis #    Time      Ant    Pol U(kLam)  V(kLam)   Amp  Phase   Amp  Phase   Amp  Phase
     1 05:02:30.7   1-   1 RR     0.00     0.00  104.786    0   105.167    0   106.508    0
     2 05:02:30.7   2-   2 RR     0.00     0.00  105.545    0   106.359    0   107.692    0
     3 05:02:30.7   4-   4 RR     0.00     0.00  105.542    0   106.023    0   107.893    0
 ...
    14 05:02:30.7  15-  15 RR     0.00     0.00  104.073    0   105.818    0   107.511    0
    15 05:02:30.7   1-   2 RR     0.00     0.00   98.883   34    97.093   46   100.276   59
    16 05:02:30.7   1-   4 RR     0.00     0.00   98.703   24    96.451   20    99.647   59
...
```

Some programs use these data for calibration purposes. For example `uvcal` has an option to normalize the cross-correllation data V_{ij} by the preceding auto-correlation data with $\sqrt{V_{ii}V_{jj}}$. This is a different method to (amplitude) bandpass calibrate.

```
  % uvcal vis=xxx.mir out=yyy.mir options=fxcal
```

2.6.4 Noise Source Passband Calibration

The noise source is only present in the LSB and can also be used to bandpass calibrate narrow calibrator modes. Only for data since early December 2006 has the signal of the Noise Source been sufficiently amplified to be useful for this calibration mode.

The following procedure uses the phases of a wide band signal (in Window 2) and applies them to a narrow band signal, in order to check phase transfer:

```
# bwsel can select out pieces of a track with the same BW settings

% selfcal vis=ct010.500_500_500.2006dec01.1.miriad select='source(3c279),win(2)' refant=9 interval=20
% uvcat   vis=ct010.500_8_500.2006dec01.1.miriad   out=3c279.8mhz.1dec select='source(3C279)'
% gpcopy  vis=ct010.500_500_500.2006dec01.1.miriad out=3c279.8mhz.1dec options=nopass
```

The phases in the `3c279.8mhz.1dec` data can now be compared to that of the noise source, and will still show offsets compared to that of the noise source.

The amplified noise source can effectively remove any passband variations. For example, by applying an mfcal solution on the narrow band of the noise source (skipping the first channel):

```
% mfcal vis=ct010.500_31_500.2006dec01.1.miriad interval=999 line=channel,62,2,1,1 refant=9 tol=0.001 \
  select='source(noise),win(2)'
```

If the signal of interest is in the USB, where there is no noise source, the data will have to be conjugated into the USB and headers faked in order for `mfcal` to apply the correction, after a slight manual copying of important header variables:

```
% uvcat vis=ct010.500_31_500.2006dec01.1.miriad out=noise.lsb select='source(noise),win(2)
% uvcat vis=ct010.500_31_500.2006dec01.1.miriad out=source.usb select='source(3C279),win(5)'
% uvcal vis=noise.lsb options=conjugate  out=noise.usb

# look at the parameters for the spectra in USB and LSB
% uvlist vis=source.usb options=spec
% uvlist vis=noise.lsb options=spec

# cheat and copy two important variables accross
# note sfreq varies with time, sdf does not
# see bandcal.csh for more automated methods
% uvputhd vis=noise.usb hdvar=sfreq varval=96.99336 out=noise2.usb
% rm -rf noise.usb
% uvputhd vis=noise2.usb hdvar=sdf varval=0.00049 out=noise.usb

# now calibrate USB
% mfcal vis=noise.usb interval=9999 line=channel,62,2,1,1 refant=9 tol=0.001
% gpcopy vis=noise.usb out=source.usb options=nocal
```

2.6.5 Phase Transfer

2.6.6 Absolute Flux Calibration

Although one can rely on known fluxes of strong calibrators such as 3C273 and 3C111, their actual flux varies with time and you will need to depend on what CARMA, or other observatories, have supplied for you. The best method is to add a planet for bootstrapping the flux of your flux calibrator, at least if a planet in available during your observation. An alternative way is to use a planet, if available, in your observation and bootstrap its flux to scale the flux of your phase or amplitude calibrator. [8]

It maintains a list of 11 "secondary flux calibrators", and publishes their fluxes as function of time. Fluxes are maintained in Miriad in a database that you can consult using the `calflux` program:

```
% calflux source=3c84
```

[8] http://www.astro.uiuc.edu/~wkwon/CARMA/fluxcal

```
  ...
Flux of: 3C84      03FEB13.00 at  86.2 GHz:  4.30 Jy; rms: 0.20 Jy
Flux of: 3C84      03MAR28.00 at  86.2 GHz:  4.30 Jy; rms: 0.20 Jy
Flux of: 3C84      03APR17.00 at  86.2 GHz:  4.20 Jy; rms: 0.30 Jy
Flux of: 3C84      03AUG17.00 at  86.2 GHz:  4.00 Jy; rms: 0.30 Jy
Flux of: 3C84      03AUG18.00 at  86.2 GHz:  4.10 Jy; rms: 0.30 Jy
Flux of: 3C84      03SEP25.00 at  86.2 GHz:  4.50 Jy; rms: 0.20 Jy
Flux of: 3C84      06DEC12.00 at  93.3 GHz:  6.57 Jy; rms: 0.99 Jy
...
Flux of: 3C84      07OCT03.00 at 224.0 GHz:  4.53 Jy; rms: 0.68 Jy
Flux of: 3C84      07OCT10.00 at  93.4 GHz:  7.41 Jy; rms: 1.11 Jy
Flux of: 3C84      07OCT10.00 at 224.0 GHz:  3.57 Jy; rms: 0.54 Jy
Flux of: 3C84      07OCT12.00 at  95.0 GHz:  6.60 Jy; rms: 0.10 Jy
Flux of: 3C84      07OCT12.00 at  90.7 GHz:  6.80 Jy; rms: 0.10 Jy
....
```

This calibration list is essentially the old BIMA flux calibrator history now appended with new CARMA values, so there is a gap between 2004 and 2006 when the BIMA dishes were moved from Hat Creek to Cedar Flat to the merged CARMA array.

Another source of information is the flux data maintained by ATCA[9] and SMA[10].

`xplore` is a tool outside of miriad that also contains time-flux tables for each source based on the same table.

`bootflux`... example

2.6.7 Absolute Flux Calibration: MARS

A special case has been reserved for the planet Mars, since it offers an option to fine-tune your calibration. The Miriad task `marstb` will interpolate a table of calculated values to a given frequency and date in the range 1999-2014, used as follows:

To find the model value:

```
   marstb epoch=08mar02 freq=95.0
Brightness temperature at  95.0 GHz: 187.675
```

to find the value of brightness temperature used in your data, read the variable PLTB using either the varplt log option or uvio if it is in your distribution, e.g.

```
   varplt vis=ct007.mars_88GHz.2006nov12.1.mir yaxis=pltb log=tblog
   more tblog
```

or

```
   uvlist vis=ct007.mars_88GHz.2006nov12.1.mir options=var,full | grep pltb
          pltb      206.6920013
```

To change the value of PLTB in your file, use uvputhd (makes new file):

```
   uvputhd vis=ct007.2008mar02_3mm.mars.1.mir hdvar=pltb varval=187.675 \
           type=r out=ct007.2008mar02_3mm.mars.1.mir_fixed
```

The model values are disk-averaged Planck brightness temperatures from the Caltech Thermal Model. As Mel noted, Mars isn't always a disk and dust storms can't be accomodated in this model, but the Caltech model values should be more reliable than CARMA's previous model (10% different in the example above from a month ago).

[9] http://www.narrabri.atnf.csiro.au/calibrators/
[10] http://sma1.sma.hawaii.edu/callist/callist.html

2.7 Mapping and Deconvolution

CARMA is a heterogeneous array, currently with 2 different types of antennae (10m and 6m), and as such will contribute 3 different types of baselines with OVRO-OVRO, BIMA-BIMA and OVRO-BIMA baselines. The latter is currently labeled in the visibility data as a `CARMA` (nominally about 8m) antennae, the first two simply being "pure" `OVRO` (10m) and `HATCREEK` (6m) [11].

If you want to map anything but a point source in the phase center, you MUST map your source in mosaic'd mode, even if you have a single pointing!

2.7.1 Mosaicing

`mospsf` needs to estimate the "average" beam appropriate for restoring.

```
  % invert ... beam=xxx.bm options=systemp,double,mosaic imsize=129,129 cell=1,1
  % imfit in=xxx.bm object=beam

  % mossdi ...
or
  % mosmem ...

  % restor ... fwhm=8,6 pa=40


TODO: needs more explanation
```

Even for a single pointing observation, your beam (dataset `xxx.bm` in the example) will currently contain 3 maps (i.e. an image cube). The first plane is probably the OVRO-OVRO beam, followed by the OVRO-BIMA beam, and finally the BIMA=BIMA beam.

It is also important to set the area to be cleaned carefully. Use a mosaic sensitivity map and use something like a 1.5σ cutoff. The mask thus generated can be copied into the cube to be cleaned.

2.8 Tips and Tricks

- In selfcal style applications (selfcal, mfcal, gmake) the reference antenna `refant=` should be choosen somewhat centrally in the array.
- In the selection of a pgplot graphics device for X11 it is recommended to use the persistent driver (`device=1/xs`, `device=2/xs`, ...), which allows for as screens as you want or your screen can handle.

[11] The future CARMA array with the additional SZA 8 antennae will thus have 6 different baseline types that contribute to a different primary beam

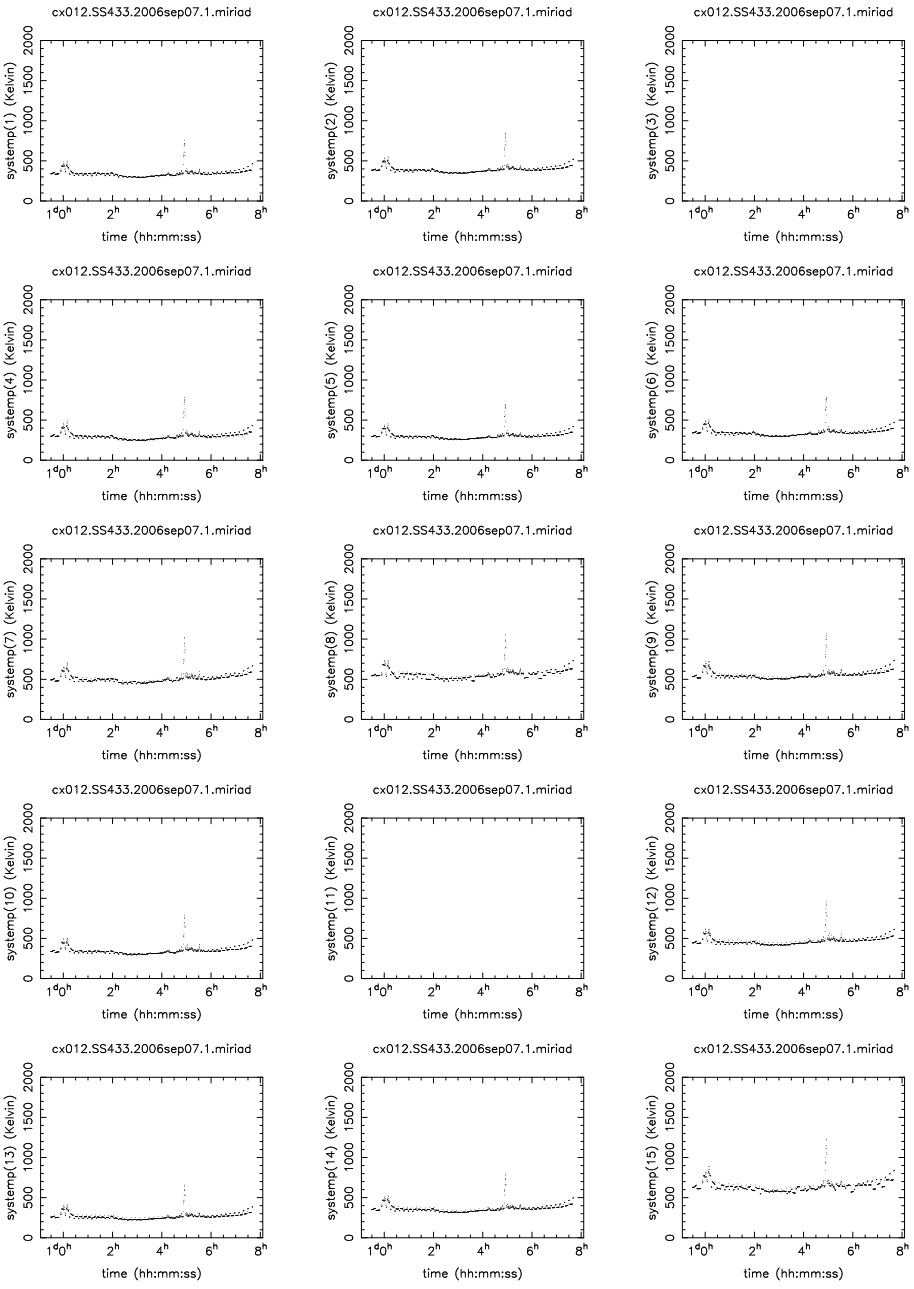

Figure 2.1: System temperature plot for 3C273 made with `varplt`, `cx012.SS433.2006sep07.1.miriad`

Chapter 3

Recipes

3.1 Calibration

3.1.1 Calibration-1

Simple calibration with a single correlator setting through all sources. A passband and phase (amplitude) calibrator is used in addition to the source of interest. useful for continuum and simple line observations in e.g. a 32 MHz window. There is also no need for NOISE source.

```
set vis=ct002blabla    # the observed multi-source dataset from CARMA archive
set bcal=3c273         # bandpass calibrator
set pcal=3c454         # phase calibrator
set src=ngc1234        # your source

# create and inspect an (antenna based) bandpass solution
mfcal vis=$vis select="source($bcal)"
gpplt vis=$vis ....

# apply, take out autocorrelations, selfcal does not handle them
uvcat vis=$vis out=$vis.bp select=-auto

# create and inspect phase and amp calibrator, assuming we trust the flux
selfcal vis=$vis.bp select="source($pcal)" options=amp,apriori,noscale
gpplt vis=$vis.bp ....

# map and deconvolve the source
invert vis=$vis.bp select="source($src)" map=map0 beam=beam0 options=mosaic
...
```

One can even imagine a more compact form, of combining the bandpass and phase calibrator:

```
set vis=ct002blabla    # the observed multi-source dataset from CARMA archive
set cal=3c273          # bandpass and phase calibrator
set src=ngc1234        # your source

# create and inspect an (antenna based) bandpass solution
mfcal vis=$vis select="-auto,source($cal)" interval=5
gpplt vis=$vis .... options=bandpass

# inspect
uvplot ...
uvspec ...
```

```
   # map and deconvolve the source
   invert vis=$vis select="source($src)" map=map0 beam=beam0 options=mosaic
   ...
```

3.1.2 Calibration-2

Narrow band (line) calibration in 2 or 8 MHz, with the NOISE source.

```
set archive=c0117.1B_225Arp220.1.miriad

set vis=vis2
set cal=cal1
set win=4,5
set refant=11
set flag=1
set linecal=1
set baseline=1
set show=0
set mosaic=1
set cont=0
set bpinterval=0.1
set cell=0.1



foreach cmdlinearg ($*)
   set $cmdlinearg
end

# select out the sources; from listobs' output we also get their purpose
#Source         Purpose    RA         Decl                          
#NOISE              B      12 29 06.70 2 03 08.60      0.00E+00     0.0
#3C273              B      12 29 06.70 2 03 08.60      0.00E+00     0.0
#3C345              G      16 42 58.81 39 48 36.99     0.00E+00     0.0
#ARP220             S      15 34 57.34 23 30 05.50     0.00E+00     0.0



if (! -d $archive) then
  carmadata -x $archive
endif


# get data from archive
rm -rf vis0
cp -a $archive vis0

# patch the frequency, but note it's a funny co(2-1) that makes the galaxy at VLSR=0
# puthd
# or
# uvputhd vis= out= hdvar=restfreq varval=226.42200

puthd in=vis0/restfreq value=226.42200 type=double




# flag data
if ($flag) then
```

```
  # C8 has terrible systemps (it's actually never present in the sky data :-)
  uvflag vis=vis0 select="ant(8)" flagval=flag
  # flag all ants after 1720
  uvflag vis=vis0 select="time(17:20,19:00)" flagval=flag
  # ant9 is really bad after 16:00 already
  uvflag vis=vis0 select="ant(9),time(16:00,19:00)" flagval=flag
endif









# linelength calibration
rm -r vis1
if ($linecal) then
  linecal vis=vis0
  uvcat vis=vis0 out=vis1
else
  uvcat vis=vis0 out=vis1
endif


# baseline corrections
rm -r vis2
if ($baseline) then
  uvedit vis=vis1 out=vis2 apfile=$MIRCAT/baselines/carma/antpos.071121
else
  uvcat vis=vis1 out=vis2
endif




rm -rf noise cal1 cal2 src
uvcat vis=$vis select="-auto,source(noise)"    out=noise
uvcat vis=$vis select="-auto,source(3C273)"    out=cal1
uvcat vis=$vis select="-auto,source(3C345)"    out=cal2
uvcat vis=$vis select="-auto,source(ARP220)"   out=src

# inspect passband calibrator
uvspec vis=cal1 device=1/xs interval=999 nxy=5,3 axis=channel,phase
uvspec vis=cal1 device=1/xs interval=999 nxy=5,3 axis=channel,amp
# and in vel space... notice with uvlist-spectra, do we need to patch?
uvspec vis=cal1 device=1/xs interval=999 nxy=5,3 axis=velocity,amp "select=win(4,5,6)"

# inspect phase calibrator
uvplt vis=cal2 device=/xs axis=time,phase "select=win($win)"
uvplt vis=cal2 device=/xs axis=time,amp   "select=win($win)"



# inspect amps of source, didn't see any bad points
uvplt vis=src  device=/xs axis=time,amp   "select=win($win)"

# make passband; use short interval, especially for 1mm (default is 5min)
# can even go as low as 10sec if you have enough signal
mfcal vis=cal1 "select=win($win)" refant=$refant interval=$bpinterval
# and inspect
gpplt vis=cal1 device=/xs yaxis=amp nxy=5,3 options=band yrange=0,2
gpplt vis=cal1 device=/xs yaxis=phase nxy=5,3 options=band

# stuff it in the phase cal
gpcopy vis=cal1 out=cal2 options=nopol,nocal
uvcat vis=cal2 out=cal3
rm -rf cal2
```

```
mv cal3 cal2
```

```
# make gain calibrator
selfcal vis=cal2 options=amp,apriori,noscale "select=win($win)"  refant=$refant
# and inspect
gpplt vis=cal2 device=/xs yaxis=phase nxy=5,3
gpplt vis=cal2 device=/xs yaxis=amp   nxy=5,3

# copy gain tables to the source
gpcopy vis=cal1 out=src options=nopol,nocal
gpcopy vis=cal2 out=src options=nopol,nopass

# map the source
rm -rf map0 beam0 beam0psf cc0 cm0

  invert vis=src map=map0 beam=beam0 select="win($win)" options=mosaic,double,systemp imsize=513 cell=$cell
  mossdi map=map0 beam=beam0 out=cc0  region=@arp220-clean.reg > mossdi.log
  mospsf beam=beam0 out=beam0psf
  imfit in=beam0psf object=beam region=quarter > imfit.log
  set bmaj = `grep Major    imfit.log | tail -1 | awk '{print $4}'`
  set bmin = `grep Minor    imfit.log | tail -1 | awk '{print $4}'`
  set bpa  = `grep Position imfit.log | tail -1 | awk '{print $4}'`
  restor map=map0 beam=beam0 model=cc0  out=cm0 fwhm=$bmaj,$bmin pa=$bpa
```

After this the

```
#
if ($cont) then
  # called with win=1,2 cont=1
  rm -rf cont
  moment in=cm0 out=cont mom=-1
else
  # called with win=4,5 cont=0
  echo CONTSUBS cannot do this yet
  rm -rf cm0cont cm0line
  replicate in=cont template=cm0 out=cm0cont
  maths exp=cm0-cm0cont out=cm0line
  #
  cgdisp type=c in=cm0line  "region=arcsec,box(-5,0,5,10)" nxy=6,6 slev=p,1 levs1=10,20,30
endif
```

3.1.3 Calibration-3

Switch correlator setup, with phase transfer.

3.1.4 gmake/gfiddle

Douglas Friedel wrote a script to split a dataset and runs `gmakes/gfiddle` on its parts. There are currently some issues with using these old BIMA g-routines. This will be looked into

The basic procedure is to get a dataset with two sidebands. Depending on your correllator setting you can use the `line=` and `select=` keywords in `gmakes` to get those:

```
  gmakes  vis=cal1  out=gvis1 line=wide,2,1,3,3
```

```
gfiddle vis=gvis1 out=gvis2 device=/xs nxy=5 fit=poly,0,2 clip=10
gapply  vis=cal1  out=gcal1 gvis=gvis2
```

after which you can check the gain and phase corrected calibrator for any more problems.

Now these gains can be applied to the source, after which it can be mapped.

```
gapply  vis=src   out=gsrc  gvis=gvis2
```

3.2 Bandpass calibration

The script below, `bandcal.csh`, is a working example how Jin Koda's M51 data can be passband calibrated. Courtesy Stuart Vogel.

```
 1:  #! /bin/csh -f
 2:  #
 3:  # 1. Uses noise source for narrow-band channel to channel bandpass calibration
 4:  #    Conjugate LSB for USB
 5:  # 2. Uses astronomical source for wideband and low-order polynomical narrow-
 6:  #    band passband calibration
 7:  # 3. Uses hybrid mode data for band-offset phase calibration
 8:  # 4. Generates temporal phase calibration from phase calibrator using
 9:  #    super-wideband (average of all three bands from both sidebands)
10:  # 5. Applies calibrations to each of the source data bands
11:  # 6. Glues source bands back together
12:  # 7. Flags bad channels in overlap region between bands.
13:
14:  # Assumes that a relatively bright quasar has been observed in the following
15:  # modes:
16:  # 1. 500/500/500
17:  # 2. nb / nb/ nb   nb=narrowband
18:  # 3. With 2 bands in narrowband and the other in 500.  aka "hybrid" mode
19:  #     Note - easy mod to script to use 1 band in nb, others in hybrid.
20:  #
21:  # SNV 2/18/2007
22:
23:  # Assumes data properly flagged so that self cal solutions are good!!
24:  # Make sure refant is a good choice!
25:
26:  # To-do list:
27:  # 1. this script assumes just one visibility calibrator
28:
29:  # User parameters
30:
31:  set vis = c0064.jk_m51co_c.4.miriad  # visibility file
32:  set refant=9                         # reference antenna
33:  set cal = 3C279                     # passband calibrator
34:  set viscal = 1153+495               # visibility calibrator
35:  set source = M51MOS                 # source
36:  set nb_array = ( 4 5 6 )            # spectral line bands to calibrate
37:  set wide_array = ( 5 6 5  )         # hybrid band with wide setup
38:       # For each element in nb_array, the
39:       # corresponding element in wide_array
40:                                      # should be the hybrid band that is wideband
41:  set superwidewin = "2,3,5,6"        # windows to use for super-wideband
42:  set superwidechan = "1,1,60"        # Channels for superwide
43:  set bw = 64                         # Spectral Line bandwidth
44:  set wideline = "1,3,11,11"          # line type for 500 MHz
45:  set narrowline = "1,3,58,58"        # line type for narrow band
46:  set sideband = "usb"                # Sideband (used for noise conjugation)
47:  set calint = 0.2                    # passband calibration interval (minutes)
48:  set vcalint = 42                    # visibility calibrator cal interval
49:  set order = 1                       # polynomial order for smamfcal fit
50:  set edge = 3                        # # of edge channels to discard in smamfcal
```

```
51:  set badants = "2,3,5"                    # bad antennas to flag
52:                                          # Do heavy uvflagging prior to script
53:  set badchan1 = "6,61,1,1"               # bad overlap channels between 1st 2 bands
54:  set badchan2 = "6,124,1,1"              # bad overlap channels between 2nd 2 bands
55:  set restfreq = 115.271203               # rest frequency of line
56:
57:  # End user parameters
58:
59:  uvflag vis=$vis select=anten'('$badants')' flagval=flag
60:
61:  rm -rf all.wide all.nb
62:  rm -rf $cal.wide* $cal.nb* $cal.hyb*
63:
64:  # Select all-wideband and all-narrowband data
65:  bwsel vis=$vis bw=500,500,500 nspect=6 out=all.wide
66:  bwsel vis=$vis bw=$bw,$bw,$bw nspect=6 out=all.nb
67:
68:  # First get super-wideband on passband calibrator and phase calibrator
69:  rm -r $cal.wide $cal.wide.0 $viscal.v.wide $viscal.v.wide.0
70:  uvcat vis=all.wide out=$cal.wide.0 \
71:          "select=-auto,source($cal)" options=nocal,nopass
72:  uvcat vis=all.wide out=$viscal.v.wide.0 \
73:          "select=-auto,source($viscal)" options=nocal,nopass
74:
75:  # mfcal passband on superwideband
76:  # Don't bother using noise source for superwideband
77:  mfcal vis=$cal.wide.0 interval=$calint refant=$refant
78:  echo "**** Plot super-wideband passband on $cal.wide.0 "
79:  gpplt vis=$cal.wide.0 options=bandpass yaxis=phase nxy=4,4 yrange=-360,360 device=bp$cal.wide.0.ps/ps
80:  gv bp$cal.wide.0.ps
81:
82:  # Inspect temporal phase variation on superwideband
83:  echo "**** Check temporal phase variations on superwideband $cal.wide.0 "
84:  gpplt vis=$cal.wide.0 yaxis=phase yrange=-360,360 nxy=4,4 device=p$cal.wide.0.ps/ps
85:  gv p$cal.wide.0.ps
86:
87:  # Apply superwideband passband for later use in band offset cal
88:  uvcat vis=$cal.wide.0 out=$cal.wide options=nocal
89:
90:  # Copy wideband passband to visibility calibrator
91:  gpcopy vis=$cal.wide.0 out=$viscal.v.wide.0 options=nocal,nopol
92:  uvcat vis=$viscal.v.wide.0 out=$viscal.v.wide options=nocal
93:
94:  # Determine phase gain variations on visibility calibrator using superwide
95:  rm -r $viscal.v.wide.sw
96:  uvcat vis=$viscal.v.wide out=$viscal.v.wide.sw select='win('$superwidewin')'
97:  selfcal vis=$viscal.v.wide.sw line=channel,$superwidechan \
98:       interval=$vcalint options=phase refant=$refant
99:  echo "**** Phases on the superwideband visibility calibrator $viscal.v.wide.sw"
100: gpplt vis=$viscal.v.wide.sw device=p$viscal.v.wide.sw.ps/ps yaxis=phase yrange=-360,360 nxy=4,4
101: gv p$viscal.v.wide.sw.ps
102:
103: # LOOP OVER EACH NARROW BAND
104:
105: set nblength = $#nb_array
106: if $nblength == 1 set list = 1
107: if $nblength == 2 set list = ( 1 2 )
108: if $nblength == 3 set list = ( 1 2 3 )
109:
110: foreach i  ( $list )
111:
112: set nb = $nb_array[$i]
113: set wide = $wide_array[$i]
114: rm -r all.hyb
115:
116: # Select hybrid data
117: # NB:  assumes only 1 band is in wideband mode; if two bands are in wideband
118: #      mode, change hybrid selection to select on nb and modify bw=
119: if ( $wide == 1 || $wide == 4 ) then
120:     if ($nb == 2 || $nb == 5) then
```

```
121:        bwsel vis=$vis nspect=6 bw=500,$bw,0 out=all.hyb
122:      else
123:        bwsel vis=$vis nspect=6 bw=500,0,$bw out=all.hyb
124:      endif
125:      endif
126:  if ( $wide == 2 || $wide == 5 ) then
127:      if ($nb == 1 || $nb == 4) then
128:        bwsel vis=$vis nspect=6 bw=$bw,500,0 out=all.hyb
129:      else
130:        bwsel vis=$vis nspect=6 bw=0,500,$bw out=all.hyb
131:      endif
132:      endif
133:  if ( $wide == 3 || $wide == 6 ) then
134:      if ($nb == 1 || $nb == 4) then
135:        bwsel vis=$vis nspect=6 bw=$bw,0,500 out=all.hyb
136:      else
137:        bwsel vis=$vis nspect=6 bw=0,$bw,500 out=all.hyb
138:      endif
139:      endif
140:  set test = `uvio vis=all.hyb | grep -i source | awk '{if (NR==1) print $4}'`
141:  if ($test == "") then
142:    echo
143:    echo "FATAL!  There appears to be no valid data in all.hyb"
144:    echo "This is likely to be because wide_array[$i] = $wide is not valid"
145:    echo "(i.e. band $wide is not really wideband), or one of the other "
146:    echo "bands is not really narrowband.  Use uvindex to sort this out"
147:    exit
148:  endif
149:
150:  echo "**** Be sure that bands are found by inspecting uvlist output!"
151:  echo "**** If no frequency info is found, that bwsel parameters are wrong"
152:  uvlist vis=all.hyb options=spectra
153:
154:  # Now we need to select single bands to process in this pass
155:  # Select by source and band
156:  # First get the two bands in all-wideband mode
157:  # Note that we use super-wideband calibrated file for the wide mode
158:  rm -rf $cal.win$nb* $cal.win$wide* $cal.wide.win$wide* $cal.wide.win$nb*
159:  rm -rf $cal.hyb.win$nb* $cal.hyb.win$wide* noise.nb.win$nb*
160:  uvcat vis=$cal.wide out=$cal.wide.win$wide "select=-auto,source($cal),win($wide)" \
161:  options=nocal,nopass
162:  uvcat vis=$cal.wide out=$cal.wide.win$nb "select=-auto,source($cal),win($nb)" \
163:  options=nocal,nopass
164:
165:  # Now select hybrid wideband band
166:  uvcat vis=all.hyb out=$cal.hyb.win$wide.0 "select=-auto,source($cal),win($wide)" \
167:  options=nocal,nopass
168:  # Now select the hybrid and all-narrowband narrow bands
169:  # nb bands require extra step (applying noise source)
170:  # we did not bother with noise source for wideband
171:  uvcat vis=all.hyb out=$cal.hyb.win$nb.00 "select=-auto,source($cal),win($nb)" \
172:  options=nocal,nopass
173:  uvcat vis=all.nb out=$cal.nb.win$nb.00 "select=-auto,source($cal),win($nb)" \
174:  options=nocal,nopass
175:
176:  # copy wideband passband determined from all-wideband mode to hybrid wideband
177:  gpcopy vis=$cal.wide.0   out=$cal.hyb.win$wide.0 options=nocal,nopol
178:  uvcat vis=$cal.hyb.win$wide.0  out=$cal.hyb.win$wide   options=nocal
179:
180:  # get the noise source data.  Use the noise source data obtained in all
181:  # narrowband mode, and assume it also can be applied to hybrid narrowband
182:
183:  if ($sideband == "USB" || $sideband == "usb" ) then
184:    echo " **** PROCESSING USB"
185:    rm -r noise.lsb noise.usb
186:    @ lsbnb = $nb - 3
187:    uvcat vis=all.nb out=noise.lsb "select=-auto,source(NOISE),win($lsbnb)" \
188:          options=nocal,nopass
189:    uvcat vis=all.nb out=noise.usb "select=-auto,source(NOISE),win($nb)" \
190:          options=nocal,nopass
```

```
191:    set sdf = `uvio vis=noise.usb | grep sdf | grep DATA | awk '{print $5}'`
192:    set sfreq = `uvio vis=noise.usb | grep sfreq | grep DATA | awk '{if (NR==1) print $5}'`
193:    uvcal vis=noise.lsb out=noise.nb.win$nb.00 options=conjugate
194:    puthd in=noise.nb.win$nb.00/sfreq value=$sfreq type=d
195:    puthd in=noise.nb.win$nb.00/sdf value=$sdf type=d
196:  else
197:    uvcat vis=all.nb out=noise.nb.win$nb.00 "select=-auto,source(NOISE),win($nb)" \
198:          options=nocal,nopass
199:  endif
200:
201:  # For narrowband windows, first do a passband cal using noise source
202:  mfcal vis=noise.nb.win$nb.00 refant=$refant
203:  echo "**** Passband cal using noise source"
204:  gpplt vis=noise.nb.win$nb.00 device=bpnoise.nb.win$nb.00.ps/ps options=bandpass yaxis=phase nxy=4,4 \
205:         yrange=-90,90
206:  gv bpnoise.nb.win$nb.00.ps
207:
208:  # Copy noise passband to astronomical all-narrowband and hybrid narrowbands,
209:  # and apply
210:  gpcopy vis=noise.nb.win$nb.00 out=$cal.nb.win$nb.00 options=nocal,nopol
211:  gpcopy vis=noise.nb.win$nb.00 out=$cal.hyb.win$nb.00 options=nocal,nopol
212:  uvcat vis=$cal.nb.win$nb.00 out=$cal.nb.win$nb.0 options=nocal
213:  uvcat vis=$cal.hyb.win$nb.00 out=$cal.hyb.win$nb.0 options=nocal
214:
215:  # use smamfcal with 1st order polynomial to
216:  # get passband on hybrid narrowband and copy to all narrowband
217:  smamfcal vis=$cal.hyb.win$nb.0 interval=$calint refant=$refant edge=$edge options=opolyfit \
218:            polyfit=$order
219:  echo "**** Hybrid narrowband passband on $cal.hyb.win$nb.0 "
220:  gpplt vis=$cal.hyb.win$nb.0 options=bandpass yaxis=phase nxy=4,4 yrange=-90,90 \
221:      device=bp$cal.hyb.win$nb.0.ps/ps
222:  gv bp$cal.hyb.win$nb.0.ps
223:
224:  # Copy narrowband passband from hybrid to all-narrowband mode
225:  gpcopy vis=$cal.hyb.win$nb.0    out=$cal.nb.win$nb.0 options=nocal,nopol
226:
227:  #Check that all-narrowband passband is flat
228:  rm -r test.pass
229:  uvcat vis=$cal.nb.win$nb.0 out=test.pass
230:  mfcal vis=test.pass refant=$refant
231:  echo "**** Narrowband passband (should be flat!)  on $cal.hyb.win$nb.0 "
232:  gpplt vis=test.pass options=bandpass yaxis=phase nxy=4,4 yrange=-90,90 \
233:        device=bptest.ps/ps
234:  gv bptest.ps
235:
236:  # Apply astronomical narrowband passband to hybrid and all-narrowband
237:  uvcat vis=$cal.hyb.win$nb.0    out=$cal.hyb.win$nb     options=nocal
238:  uvcat vis=$cal.nb.win$nb.0    out=$cal.nb.win$nb     options=nocal
239:
240:  # Selfcal on hybrid wideband to remove temporal variations
241:  # prior to band offset calibration
242:  selfcal vis=$cal.hyb.win$wide line=channel,$wideline \
243:       interval=$calint options=phase refant=$refant
244:
245:  # Copy selfcal solution over to narrow hybrid band and apply
246:  copyhd in=$cal.hyb.win$wide out=$cal.hyb.win$nb items=gains,ngains,nsols,interval
247:  uvcat vis=$cal.hyb.win$nb out=$cal.hyb.win$nb.a
248:
249:  # Selfcal on narrow band of hybrid to determine band offset
250:  selfcal vis=$cal.hyb.win$nb.a line=channel,$narrowline \
251:      interval=9999 options=phase refant=$refant
252:  echo "**** Band offset between hybrid-narrowband $cal.hyb.win$nb.a"
253:  echo "****  and hybrid-wideband $cal.hyb.win$nb"
254:  gplist vis=$cal.hyb.win$nb.a options=phase
255:  # Also copy band offset to text file
256:  gplist vis=$cal.hyb.win$nb.a options=phase >! mnband_offset.$cal.hybwin$nb.txt
257:
258:  # Test by applying to calibrator observed in all-narrowband mode
259:  copyhd in=$cal.hyb.win$nb.a out=$cal.nb.win$nb items=gains,ngains,nsols,interval
260:  uvcat vis=$cal.nb.win$nb out=$cal.nb.win$nb.a
```

```
261:
262:  # Remove antenna phase gain using super-wideband
263:  rm -r $cal.wide.sw
264:  uvcat vis=$cal.wide out=$cal.wide.sw select='win('$superwidewin')'
265:  selfcal vis=$cal.wide.sw line=channel,$superwidechan \
266:      interval=9999 options=phase refant=$refant
267:  # Copy super-wideband gain to narrowband and apply
268:  copyhd in=$cal.wide.sw out=$cal.nb.win$nb.a items=gains,ngains,nsols,interval
269:  uvcat vis=$cal.nb.win$nb.a out=$cal.nb.win$nb.a.sc
270:
271:  # Selfcal to check that phases are roughly zero
272:  # to within amount expected given temporal variations over interval
273:  # between superwideband and all-narrowband observerations
274:  selfcal vis=$cal.nb.win$nb.a.sc line=channel,$narrowline \
275:       interval=9999 options=phase refant=$refant
276:
277:  # List gains, which should be near zero except for temporal variations
278:  # over interval between wideband and narrow band observations of cal
279:  echo "**** Phase offset between super-wideband $cal.wide.sw "
280:  echo "**** and all-narrow narrow band $cal.nb.win$nb.a.sc "
281:  echo "**** Check that phases are near zero, limited by atmospheric flucatuations"
282:  gplist  vis=$cal.nb.win$nb.a.sc options=phase
283:
284:  # Now apply calibrations to source data
285:  rm -r  $source.win$nb* $source.win$nb.bcal
286:  # First select source data
287:  uvcat vis=all.nb out=$source.win$nb.00 \
288:     "select=-auto,source($source),win($nb)"  options=nocal,nopass
289:  # Copy and apply noise passband to source
290:  gpcopy vis=noise.nb.win$nb.00 out=$source.win$nb.00 options=nocal,nopol
291:  uvcat vis=$source.win$nb.00 out=$source.win$nb.0 options=nocal
292:  # Copy and apply astronomical passband
293:  gpcopy vis=$cal.hyb.win$nb.0 out=$source.win$nb.0 options=nocal,nopol
294:  uvcat vis=$source.win$nb.0 out=$source.win$nb options=nocal
295:  # Copy band offset to source
296:  copyhd in=$cal.hyb.win$nb.a out=$source.win$nb items=gains,ngains,nsols,interval
297:  rm -r $source.win_$i
298:  # Apply band offset using smachunkglue naming convention
299:  uvcat vis=$source.win$nb out=$source.win_$i
300:
301:  # end nb loop
302:  end
303:
304:  # glue together 3 bands
305:  set nblength = $#nb_array
306:
307:
308:  if ($nblength == 2) then
309:    set cfile=$source.$nb_array[1]$nb_array[2]
310:    rm -r $cfile
311:    smachunkglue vis=$source.win nfiles=$nblength out=$cfile
312:    uvflag vis=$cfile line=channel,$badchan1 flagval=flag
313:  else if ($nblength == 3) then
314:    set cfile=$source.$nb_array[1]$nb_array[2]$nb_array[3]
315:    rm -r $cfile
316:    smachunkglue vis=$source.win nfiles=$nblength out=$cfile
317:    # flag bad overlap channels
318:    uvflag vis=$cfile line=channel,$badchan1 flagval=flag
319:    uvflag vis=$cfile line=channel,$badchan2 flagval=flag
320:  else
321:    set cfile=$source.$nb_array[1]
322:    rm -r $cfile
323:    uvcat vis=$source.win out=$cfile
324:  endif
325:
326:  # put in restfreq
327:  puthd in=$cfile/restfreq type=double value=$restfreq
328:
329:  # copy super-wideband gains to source
330:  copyhd in=$viscal.v.wide.sw out=$cfile items=gains,ngains,nsols,interval
```

```
331:
332:  echo "Calibrated source file: $cfile"
```

3.3 Flux Calibration

3.3.1 Bootstrap Flux Calibration

In this example we will calculate the flux of a phase calibrator using a known flux calibrator. The flux is assumed from another source (it could be bootstrapped from a planet, or from an external list such as the SMA list of the CARMA flux table). We will assume we have both calibrators in a triple 500 MHz correlator mode for maximum sensitivity, and that all data have been flagged appropriately. We will also assume the phase calibrator is relatively bright to believe the time variance of the gains.

First a few handy definitions so we can shorten the examples:

```
set fluxcal = 3C84                  # flux calibrator  (also the miriad dataset name)
set viscal  = 0238+166              # phase calibrator (also the miriad dataset name)

set flux    = 6.6                   # flux of flux calibrator (SMA or Woojin)
set refant  = 9                     # referance antenna

set calint  = 0.2                   # passband calibration interval (minutes)
set vcalint = 25                    # visibility calibrator scan interval
set fcalint = 1                     # flux calibrator interval
set superwidewin  = "1,2,4,5"       # windows to use for superwide
set superwidechan = "1,1,60"        # channels for superwide
set lsbfluxchan   = "1,1,30,30"     # channels for calc lsb flux
set usbfluxchan   = "1,31,30,30"    # channels for calc usb flux
```

A note on setting the flux here. In the example below we do not use `options=apriori` in `selfcal` but instead set the flux value explicitly. Either way should work, but flux calibration tables are sometimes updated and can give slightly different (supposedly better of course) results.

First we will passband calibrate the flux calibrator. We will use a relatively short interval, to ensure phase wrapping in time does not wipe out the passband:

```
mfcal vis=$fluxcal interval=$calint refant=$refant
```

It is always good to inspect the calibration tables, both in frequency and time:

```
gpplt vis=$fluxcal options=bandpass yaxis=phase nxy=4,4 yrange=-360,360 device=/xs
gpplt vis=$fluxcal yaxis=phase yrange=-360,360 nxy=4,4 device=/xs
```

Notice that the first LSB and last USB window (spectral window 3 and 6) are not as well behaved as the others, and will be left out in the definition of the `superwide channel` (combining all wide band windows) .

The passband calibration table is now copied to the visibility calibrator, and a copy is made of this now passband corrected dataset:

```
gpcopy vis=$fluxcal out=$viscal options=nocal,nopol
uvcat vis=$viscal out=$viscal.wide options=nocal
```

Next, the antenna gains are determined from the flux calibrator. First we again make a passband corrected copy of all the good windows, after which we run an amplitude selfcal with the flux we think we know this source should have.

```
  uvcat vis=$fluxcal out=$fluxcal.gain options=nocal "select=win($superwidewin)"
  selfcal vis=$fluxcal.gain refant=$refant interval=$fcalint "select=source($fluxcal)" \
    options=noscale,amplitude flux=$flux
  gplist vis=$fluxcal.gain options=zeropha,amp > $fluxcal.gains

...
------------------------------------------------------------------------
Means:      1.39  0.98  1.04  1.14  0.00  1.10  1.08  1.29  1.17  1.38  1.44  1.09  1.33  1.35  1.26
Medians:    1.36  0.98  1.04  1.13  0.00  1.09  1.09  1.29  1.17  1.38  1.44  1.08  1.32  1.33  1.26
Rms:        0.09  0.03  0.03  0.07  0.00  0.02  0.03  0.02  0.05  0.04  0.04  0.03  0.04  0.05  0.03
------------------------------------------------------------------------
```

Since we will need these gain factors later on, a little Unix pipe will grab the medians into a file:

```
  grep Medians $fluxcal.gains | tr -d Medians: > $fluxcal.medians
  cat $fluxcal.medians
  1.36  0.98  1.04  1.13  0.00  1.09  1.09  1.29  1.17  1.38  1.44  1.08  1.32  1.33  1.26
```

Next the phase of the phase calibrator should be straightened out, and we use a phase-only selfcal with a fairly long integration time for this

```
  uvcat vis=$viscal.wide out=$viscal.sw "select=win($superwidewin)"
  selfcal vis=$viscal.sw line=channel,$superwidechan interval=$vcalint options=phase refant=$refant
```

Now the amplitude gains derived from the flux calibrator, can be applied to the phase calibrator, by replacing the amplitudes, and keeping the phases from the just determined selfcal solution:

```
  gplist vis=$viscal.sw options=replace jyperk=@$fluxcal.medians
```

A special program, `uvflux`, can now be used to gather some statistics on this phase calibrator. Since the calibrator is assumed to be a point source, all amplitudes should be the same (you could check this with e.g. `uvplt axis=uvd,amp`), and thus report the flux (6.18 Jy $\pm$ 2.59 for both LSB and USB in this example).

```
  uvflux vis=$viscal.sw options=nopol line=chan,$lsbfluxchan
  uvflux vis=$viscal.sw options=nopol line=chan,$usbfluxchan
  uvflux vis=$viscal.sw options=nopol > $viscal.flux

------------------------------------------------------------------------------
Source     Pol Theoretic   Vector Average      RMS      Average  RMS Amp  Number
                 RMS        (real,imag)     Scatter       Amp    Scatter  Corrs
------     --- -------- ------------------- -------  --------- --------  ------
0238+166    RR 1.3E+00  5.157E+00 -3.838E-03 3.0E+00  6.180E+00 2.59E+00 1935180
------------------------------------------------------------------------------
```

Finally, checking the time variance of the phase calibrator

```
  uvcat vis=$viscal.wide out=$viscal.wide.gain "select=win($superwidewin)"
  selfcal vis=$viscal.wide.gain refant=$refant interval=$vcalint "select=source($viscal)" \
          options=noscale,amplitude flux=$visflux
  gplist vis=$viscal.wide.gain options=zeropha,amp > $viscal.gains

  Time     Ant 1 Ant 2 Ant 3 Ant 4 Ant 5 Ant 6 Ant 7 Ant 8 Ant 9 Ant10 Ant11 Ant12 Ant13 Ant14 Ant15
19:11:10    1.33  0.99  1.01  1.07  0.00  1.07  1.10  1.34  1.19  1.39  1.48  1.06  1.28  1.37  1.24
19:13:34    1.34  0.98  1.05  1.06  0.00  1.06  1.09  1.36  1.21  1.40  1.48  1.06  1.28  1.35  1.28
19:40:47    1.41  1.01  1.01  1.12  0.00  1.07  1.10  1.45  1.23  1.40  1.53  1.08  1.32  1.45  1.31
```

20:09:41	1.36	0.98	1.05	1.12	0.00	1.12	1.19	1.58	1.28	1.44	1.68	1.09	1.38	1.47	1.39
20:42:34	1.44	0.98	1.05	1.19	0.00	1.12	1.11	1.66	1.31	1.53	1.75	1.14	1.44	1.51	1.42
20:53:42	1.50	0.95	1.03	1.14	0.00	1.09	1.07	1.63	1.29	1.56	1.78	1.15	1.42	1.54	1.41
21:22:06	1.38	1.01	1.05	1.15	0.00	1.07	1.16	1.67	1.32	1.63	1.75	1.16	1.37	1.54	1.40
21:51:16	1.41	0.99	1.05	1.10	0.00	1.06	1.11	1.67	1.39	1.60	1.74	1.14	1.37	1.50	1.43
22:21:57	1.45	1.04	1.09	1.15	0.00	1.09	1.14	1.71	1.51	1.64	1.76	1.16	1.40	1.57	1.50
22:34:31	1.54	1.05	1.07	1.14	0.00	1.07	1.10	1.70	1.48	1.56	1.65	1.15	1.39	1.55	1.47
23:02:50	1.47	1.04	1.06	1.08	0.00	1.05	1.14	1.85	1.61	1.64	1.74	0.00	1.46	1.57	1.55
23:31:26	1.56	1.10	1.07	1.17	0.00	1.09	1.22	2.21	1.72	1.70	1.90	1.25	1.52	1.65	1.64
00:00:10	1.50	1.09	1.03	1.13	0.00	1.06	1.41	4.50	1.86	1.72	1.91	1.28	1.55	1.61	1.60
Means:	1.44	1.02	1.05	1.13	0.00	1.08	1.15	1.87	1.42	1.55	1.70	1.14	1.40	1.51	1.43
Medians:	1.41	0.99	1.05	1.12	0.00	1.07	1.11	1.66	1.31	1.56	1.74	1.14	1.38	1.51	1.41
Rms:	0.07	0.05	0.02	0.04	0.00	0.02	0.09	0.82	0.21	0.11	0.14	0.07	0.08	0.09	0.12

3.4 Mosaiced Mapping and Deconvolution

Just make sure you have a good set of CPUs!

3.5 Simple Reduction

3.5.1 Simple Reduction - I

```
 1:  #! /bin/csh -f
 2:  #
 3:  # 3/7/07
 4:  #
 5:  # Script to reduce CARMA OPHB11 data
 6:  #   -- using 2 calibrators:
 7:  #            offsetcal to calibrate offset in phase b/w wide and narrow band
 8:  #                  cal to calibrate phases
 9:  #
10:  # pre-work:
11:  # set vis=c0014.OphB11_C.1.miriad
12:  # carmadata  $vis.tar.gz
13:  # puthd in=$vis/restfreq value=115.271204 type=double
14:  # uvcat vis=$vis out=1751+096W.raw 'select=source(1751+096),-auto,time(11:29:00,11:38:20)'
15:  # uvcat vis=$vis out=1751+096N.raw 'select=source(1751+096),-auto,time(11:08:00,11:30:00)'
16:  # uvcat vis=$vis out=1625-254.raw  'select=source(1625-254),-auto'
17:  # uvcat vis=$vis out=ophb11.raw    'select=source(ophb11),-auto'
18:
19:  umask 002
20:
21:  set task=flag                     # task to perform
22:  set vis==c0014.OphB11_C.1.miriad  # base project vis file
23:  set offsetcal=1751+096            # calibrator for phase offset b/w Wide and Narr. bands
24:  set cal=1625-254                  # primary phase cal name
25:  set source=ophb11                # source name
26:  set refant=8                     # refant for selfcal
27:  set interval=25                  # interval for selfcal
28:  set scselect=                    # select for selfcal on phase cal
29:  set nxy=3,3                      # number of windows for plotting
30:  set flux=1.6
31:
32:  set ointerval=9999         # interval for offset cal selfcal
33:  set olinecal=chan,1,46,45,45
34:  set linenarrow=chan,1,109,63,63
35:  set axis=time,phase
36:
37:  set line=
38:
39:  set dev=1/xs              # plotting device
40:  set select=                # select for uvplt
41:  set options=              # options for uvplt
42:
43:  set cutoff=
44:
45:  foreach _arg ($*)
46:      set $_arg
47:  end
48:
49:  goto $task
50:
51:  logs:
52:  listobs vis=
53:
54:
55:  renewoffset:
56:
57:  # Secondary Calibrator split into files with with differen correlator configs:
58:  rm -rf "$offsetcal"N.red
59:  cp -r "$offsetcal"N.raw "$offsetcal"N.red
60:  rm -rf "$offsetcal"W.red
61:  cp -r "$offsetcal"W.raw "$offsetcal"W.red
62:
63:  flagoff:
64:  uvplt vis="$offsetcal"W.red device=$dev  \
```

```
65:      nxy=$nxy select=$select options=$options axis=$axis
66:
67:  uvplt vis="$offsetcal"N.red device=$dev \
68:      nxy=$nxy select=$select options=$options axis=$axis
69:
70:  exit 0
71:
72:  selfoff:
73:
74:  # if applying BP cal to 1751, also apply to sec phase, and to source, depending on
75:  #    window ...
76:  #
77:  # or just include USB (calibrating Wide band) :
78:
79:  selfcal vis="$offsetcal"W.red refant=$refant  \
80:      line=$olinecal options=noscale interval=$ointerval
81:
82:  gpplt vis="$offsetcal"W.red yaxis=phase yrange=-180,180 \
83:      device=$dev nxy=$nxy options=w
84:
85:  # Apply Wide Band calibration to Narrow Band:
86:  gpcopy vis="$offsetcal"W.red out="$offsetcal"N.red mode=copy options=nopass
87:
88:  rm -rf "$offsetcal"Nref.red
89:  uvcat vis="$offsetcal"N.red out="$offsetcal"Nref.red
90:
91:  # Another selfcal to get offset between Wide and Narrow bands:
92:  selfcal vis="$offsetcal"Nref.red refant=$refant select=$select    \
93:      line=$linenarrow options=phase interval=$ointerval
94:
95:  gpplt vis="$offsetcal"Nref.red yaxis=phase yrange=-180,180 \
96:      device=$dev nxy=$nxy options=w
97:
98:  exit 0
99:  renew:
100:  # CAREFUL: This will erase $cal.red to start from scratch
101:  # uvcat  vis=c0014.OphB11_C.1.miriad select=source(1625-254),-auto out=1625-254.raw
102:
103:  rm -rf $cal.red
104:  cp -r $cal.raw $cal.red
105:
106:  flag:
107:  uvplt vis=$cal.red device=$dev line= \
108:      nxy=$nxy select=$select options=$options
109:
110:  #---Flag data here---
111:  exit 0
112:
113:  selfcal:
114:
115:  #selfcal phase calibrator
116:
117:  selfcal vis="$cal".red flux=$flux refant=$refant select=$scselect options=amp,apriori,noscale interval=$interval lin
118:
119:  gpplt:
120:  gpplt vis="$cal".red yaxis=phase yrange=-180,180 \
121:      device=$dev nxy=$nxy options=w
122:
123:  sleep 2
124:
125:  rm -rf $cal.dm $cal.bm
126:  invert vis="$cal".red map=$cal.dm beam=$cal.bm \
127:      imsize=237,237 cell=1,1 line=$olinecal options=systemp select=$scselect
128:
129:  rm -rf $cal.cln
130:
131:  clean map=$cal.dm beam=$cal.bm niters=10000 out=$cal.cln cutoff=$cutoff
132:
133:  rm -rf $cal.restor
134:  restor model=$cal.cln map=$cal.dm beam=$cal.bm out=$cal.restor
```

```
135:
136:  cgdisp in=$cal.restor device=$dev
137:
138:  chmod -R 775 $cal.*
139:
140:  exit 0
141:
142:  renewsource:
143:  # CAREFUL: This will erase $source.red to start from scratch
144:  # uvcat  vis=c0014.OphB11_C.1.miriad select=source(OPHB11),-auto out=ophb11.raw
145:  rm -rf $source.red
146:  cp -r $source.raw $source.red
147:
148:  flagsource:
149:  uvplt vis=$source.red device=$dev line=$line \
150:      nxy=$nxy select=$select options=$options
151:
152:  #---Flag source data here---
153:  exit 0
154:
155:
156:  invert:
157:
158:  # Copy and apply gains table from phase calibrator
159:  gpcopy vis="$cal".red/ out=$source.red mode=copy
160:
161:  rm -rf "$source"GT.red
162:  uvcat vis=$source.red out="$source"GT.red
163:
164:  # Copy and apply gains table from offset calibrator (narrow band only)
165:  gpcopy vis="$offsetcal"Nref.red out="$source"GT.red mode=copy
166:
167:  rm -rf "$source"cal.red
168:  uvcat vis="$source"GT.red out="$source"cal.red select='window(5)'
169:
170:  rm -rf $source.dm $source.bm
171:  invert vis="$source"cal.red map=$source.dm beam=$source.bm imsize=237,237 \
172:      cell=1,1 options=systemp,double,mosaic line=$line sup=0 select=$select
173:
174:  #rm -rf $source.cln $source.restor
175:  #clean map=$source.dm beam=$source.bm out=$source.cln \
176:  #    niters=10000 cutoff=$cutoff
177:  #minmax in=$source.cln
178:  #restor map=$source.dm beam=$source.bm \
179:  #    model=$source.cln out=$source.restor
180:
181:  chmod -R 775 $source.*
182:
```

3.5.2 Simple Reduction - II

The example below has been supplied by Alberto, though some administrative details have been left out to make the example less cluttered. First we define some convenient variables, so we can re-use them in the script. The rule of thumb should be any number, or certainly multiply occuring text, should be used in a (shell) variable. That rules is not quite followed in the current example:

```
set FILE="c0001.n604_coC.1.miriad"
set SRC="NGC604"
set CAL1="0205+322"
set CAL2="0237+288"
set PBCAL="3C454.3"
set NOISE="NOISE"
set FLUX="URANUS"

set WIDE="channel,1,1,15,1"
set LINE="velocity,63,-317.521,2.54,2.54"
set CAL=$CAL2
set OCAL=$CAL1
set RESTFREQ="115.271202"
set REFA=9
```

Some of variables may be quite obvious, others less. For example, the setting for `LINE=` less came from gleaning the output of `uvlist`:

```
 % uvlist vis=$FILE options=spectra
rest frequency     : 115.27120 115.27120 115.27120 115.27120 115.27120 115.27120
starting channel   :         1        16        79       142       157       220
number of channels :        15        63        63        15        63        63
starting frequency : 111.47148 111.08239 111.05307 114.93370 115.32280 115.35211
frequency interval :  -0.03125  -0.00049  -0.00049   0.03125   0.00049   0.00049
starting velocity  :  9854.752 10866.791 10943.046   849.513  -162.527  -238.782
ending velocity    : 10992.691 10945.532 11021.788  -288.426  -241.268  -317.523
velocity interval  :    81.274     1.270     1.270   -81.274    -1.270    -1.270
```

First we note that the data set from CARMA is a single miriad dataset that contains all the sources. It is often , except in the most simple cases, much more convenient to keep track of things if the data is copied to single-source (or even single-setting) datasets:

```
foreach i ($SRC $CAL1 $CAL2 $PBCAL $NOISE $FLUX)
  uvcat vis=$FILE out=$i select="-auto,source("$i")"
end
```

It cannot be stressed enough to inspect the data data visually , in as many ways as you can imagine. Here are just a few examples:

```
# phase vs. time
  uvplt vis=$CAL device=/xs line=$WIDE axis=time,phase

# amplitude vs. time
  uvplt vis=$CAL device=/xs line=$WIDE axis=time,amp

# even for the source: it probably be random, unless there are false fringes
# or it is a very strong source
  uvplt vis=$SRC device=/xs line=$WIDE axis=time,phase

  uvspec  ...
```

As a result of this inspection perhaps we found some suspicious data, and it needs to be flagged. This could be in certain channels and/or time slots. Here is an example to flag a certain time-range for antenna 5:

```
uvflag vis=$CAL,$PBCAL,$SRC flagval=flag select="ant(5),time(21:30:00,22:15:00)"
```

First we proceed with (astronomical) passband calibration, to make sure the trends we saw in phase vs. time are not washed out by passband slopes. Notice we're compressing the whole time-ranges to get a single passband shape for all times:

```
mfcal vis=$PBCAL line="channel,282,1,1,1" interval=999 refant=$REFA
```

and inspect the result

```
uvspec vis=$PBCAL axis=chan,phase line="channel,282,1,1,1" device=/xs interval=999 yrange=-180,180
```

copy the passband from the PBCAL to the CAL

```
gpcopy vis=$PBCAL out=$CAL options=nocal
```

Create new dataset with calibration applied, otherwise linetype averaging does not work properly. Use all wideband channels.

```
uvcat vis=$CAL out=$CAL.pb select="window(1,4)"
```

Proceed with amplitude-phase calibration

```
gpcopy vis=$PBCAL out=$FLUX options=nocal
uvcat vis=$FLUX out=$FLUX.pb select="window(1,4)"

selfcal vis=$FLUX.pb options=apriori,amp,noscale interval=0.1 line="channel,1,1,30,1" refant=$REFA
bootflux vis=$FLUX.pb,$CAL.pb primary=$FLUX line="channel,1,1,30,1" taver=999
```

Self calibrate the phase calibrator, with passband calibration applied, and imposing the flux found by bootflux solution

```
selfcal vis=$CAL.pb line="channel,1,1,30,1" options=amp,noscale,apriori flux=1.2 interval=20 refant=$REFA
```

Inspect again. Now each channel should have a zero phase

```
uvspec vis=$CAL.pb axis=chan,phase line="channel,272,1,10,1" device=/xs interval=999 yrange=-180,180
```

Show time series of selfcalibrated wideband channels:

```
uvplt vis=$CAL.pb axis=time,phase device=/xs line="channel,1,16,15,1"
uvplt vis=$CAL.pb axis=time,amp device=/xs line="channel,1,16,15,1"
```

Or for a phase-only calibration, we self calibrate on the phase calibrator, with passband calibration applied. Most of the time the online amplitude calibration seems very good... Note that we are averaging over all wideband channels. Channel linetype averaging does not weigh by bandwidths and/or Tsys. This is why we split out only the continuum windows.

```
selfcal vis=$CAL.pb line="channel,1,1,30,1" interval=20 refant=$REFA
```

Inspect the results, again every channel should have zero phase:

```
uvspec vis=$CAL.pb axis=chan,phase line="channel,272,1,10,1" device=/xs interval=999 yrange=-180,180
```

Now show a time series of self calibrated wideband channels:

```
uvplt vis=$CAL.pb axis=time,phase device=$device line="channel,1,16,15,1"
```

Now that all calibration is done, it is a good idea to do some sanity checks. Looking at the gain amplitudes > 1 indicate that the phase calibrator was weaker than expected, perhaps due to pointing errors,

```
gpplt vis=$CAL.pb device=$device yaxis=amp
```

The phase gains should be smooth now:

```
gpplt vis=$CAL.pb device=$device yaxis=phase
```

Looking at the phase vs time after selfcal, they should be centered around zero:

```
uvplt vis=$CAL.pb device=$device axis=time,phase line=$WIDE
```

The phase vs baseline length plot should be inspected to assess atmospheric decorrelation, it should be flaring at the longer baselines but not overall decline:

```
uvplt vis=$CAL.pb device=$device line=$WIDE axis=uvdist,phase options=nobase
```

And finally amplitude vs. time: it should be about was it was set to in the selfcal if an amplitude selfcal was done:

```
uvplt vis=$CAL.pb device=$device line=$WIDE axis=time,amp
```

In the actual example script it now continues mapping the calibrator, and finally a number of the same set of observations for the source. It can be found in the examples directory as `example-blabla.csh`.

3.5.3 Hybrid Mode Calibration - III

This example[1] originates from Misty Lavigne and Stuart Vogel and uses a dataset taken in hybrid passband mode in order to calibrate the phase offsets between the different bands. This is often the case when a single "narrow band" is not able to catch the velocity range of the object of interest, in the current correlator galaxies appear to be the primary victim of this.

It assumes that a relatively bright quasar has been observed in the following modes:

1. Three 500/500/500 MHz wide bands, currently 15 channels each.
2. Three nb/nb/nb narrowband (BW depends on what you need for your object to fill the spectral range), currently 63 channels each.
3. Two bands in narrowband and the other in 500. Aka "hybrid" mode. The procedure below can be easily modified if one band is narrow, and the others 500.

Some further comments:

1. Uses the noise source for narrow-band channel to channel bandpass calibration. Since the astronomical data is in the USB in this example and the noise source is only in the LSB, it also conjugate LSB to USB.
2. Uses an astronomical source for wideband and low-order polynomical narrow- band passband calibration
3. Uses hybrid mode data for band-offset phase calibration
4. Generates temporal phase calibration from phase calibrator using super-wideband (average of all three bands from both sidebands)
5. Applies calibrations to each of the source data bands
6. Glues source bands back together
7. Flags bad channels in overlap region between bands.

There are various other assumptions in the procedure below that almost never apply exactly to your data. We also assume that the data have been properly flagged and that self cal solutions are good, and that the reference antenna is a proper choice. The script also assumes only one visibility calibrator.

The script first sets a few parameters, but note that some of these parameters (e.g. `superwidechan`, `narrowline`) depend on the specific correlator mode that was choosen.

```
set vis           = alldata.vis      # visibility file
set refant        = 10               # reference antenna
set cal           = 3C273            # passband calibrator
set viscal        = 1058+015         # visibility (phase) calibrator
set fluxcal       = 3C273            # flux calibrator
set source        = N3627            # source
set nb_array      = ( 4 5 6 )        # spectral line bands to calibrate
set wide_array    = ( 5 4 5  )       # hybrid band with wide setup
     # For each element in nb_array, the
     # corresponding element in wide_array
                                     # should be the hybrid band that is wideband
set superwidewin  = 4,5              # windows to use for super-wideband
set superwidechan = 1,1,30           # Channels for superwide
set bw            = 64               # Spectral Line bandwidth
```

[1] See also CARMA memo "CARMA Hybrid mode" (Lisa Wei, in prep)

```
set wideline      = 1,3,11,11        # line type for 500 MHz
set narrowline    = 1,3,58,58        # line type for narrow band
set sideband      = usb              # Sideband (used for noise conjugation)
set calint        = 0.2              # passband calibration interval (minutes)
set vcalint       = 30               # visibility calibrator cal interval
set fcalint       = 30               # flux calibrator interval
set ampcalint     = 30               # selfcal amplitude interval
set flux          = 18               # flux of flux calibrator (SMA or Woojin)
set visflux       = 5.1              # flux of vlisibility calibrator, calculated from fluxcal.csh
set order         = 1                # polynomial order for smamfcal fit
set edge          = 3                # of edge channels to discard in smamfcal
set badants       = 2,15             # bad antennas to flag
                                     # Do heavy uvflagging prior to script
set badchan1      = 6,61,1,1         # bad overlap channels between 1st 2 bands
set badchan2      = 6,124,1,1        # bad overlap channels between 2nd 2 bands
set restfreq      = 115.271203       # rest frequency of line

set gv=ghostview                     # your postscript viewer
```

Although you very most likely will have inspected the visibility data and perhaps had to flag bad data in time, frequency and/or baseline/antennae space, here is a simple example to flag two antennas:

```
uvflag vis=$vis select=ant'('$badants')' flagval=flag
```

Select the bands

```
# Select all-wideband and all-narrowband data
  rm -rf all.wide all.nb $cal.wide* $cal.nb* $cal.hyb*
bwsel vis=$vis bw=500,500,500 nspect=6 out=all.wide
bwsel vis=$vis bw=$bw,$bw,$bw nspect=6 out=all.nb

# First get super-wideband on passband calibrator and phase calibrator
  rm -rf $cal.wide $cal.wide.0 $viscal.v.wide $viscal.v.wide.0
uvcat vis=all.wide out=$cal.wide.0 \
        "select=-auto,source($cal),win($superwidewin)" options=nocal,nopass
uvcat vis=all.wide out=$cal.wide.1 \
        "select=-auto,source($cal)"                    options=nocal,nopass
uvcat vis=all.wide out=$viscal.v.wide.0 \
        "select=-auto,source($viscal)"                 options=nocal,nopass
```

Run `mfcal` on the superwideband (500/500/500) data. Don't bother using the noise source for the superwideband. Inspect the antenna based solutions in both frequency and time.

```
mfcal vis=$cal.wide.0 interval=$calint refant=$refant

# Inspect super-wideband passband
gpplt vis=$cal.wide.0 options=bandpass yaxis=phase nxy=4,4 yrange=-360,360 device=bp$cal.wide.0.ps/ps
$gv bp$cal.wide.0.ps

# Inspect temporal phase variation on superwideband
gpplt vis=$cal.wide.0 yaxis=phase yrange=-360,360 nxy=4,4 device=p$cal.wide.0.ps/ps
$gv p$cal.wide.0.ps

# Apply superwideband passband for later use in band offset cal
gpcopy vis=$cal.wide.0 out=$cal.wide.1
uvcat vis=$cal.wide.1 out=$cal.wide options=nocal

# Copy wideband passband to visibility calibrator
gpcopy vis=$cal.wide.0 out=$viscal.v.wide.0 options=nocal,nopol
uvcat vis=$viscal.v.wide.0 out=$viscal.v.wide options=nocal

# Determine phase gain variations on visibility calibrator using superwide
  rm -rf $viscal.v.wide.sw $viscal.v.wide.sw.test
```

```
uvcat vis=$viscal.v.wide out=$viscal.v.wide.sw select='win('$superwidewin')'
uvcat vis=$viscal.v.wide out=$viscal.v.wide.sw.test select='win('$superwidewin')'

selfcal vis=$viscal.v.wide.sw.test interval=0.1 refant=$refant
gpplt vis=$viscal.v.wide.sw.test yaxis=phase yrange=-360,360 device=testphase.ps/ps nxy=4,4
$gv testphase.ps


selfcal vis=$viscal.v.wide.sw line=channel,$superwidechan \
        interval=$vcalint options=phase refant=$refant
echo "**** Phases on the superwideband visibility calibrator $viscal.v.wide.sw"
gpplt vis=$viscal.v.wide.sw device=p$viscal.v.wide.sw.ps/ps yaxis=phase yrange=-360,360 nxy=4,4
$gv p$viscal.v.wide.sw.ps
```

Checking the phase calibrator:does it look like a nice point source. Notice we don't use mosaicing here, since it is a point source, though for extended sources you will want to use that option in `invert` when the source is mapped.

```
rm -rf $viscal.v.wide.sw.mp $viscal.v.wide.sw.bm $viscal.v.wide.sw.cl $viscal.v.wide.sw.r

invert vis=$viscal.v.wide.sw cell=0.5 imsize=257 line=chan,$superwidechan \
       map=$viscal.v.wide.sw.mp beam=$viscal.v.wide.sw.bm options=system,double sup=0

set rms = `histo in=$viscal.v.wide.sw.mp | grep Rms | awk '{print$4}'`

clean map=$viscal.v.wide.sw.mp beam=$viscal.v.wide.sw.bm out=$viscal.v.wide.sw.cl \
      niters=1000 cutoff=$rms

restor map=$viscal.v.wide.sw.mp beam=$viscal.v.wide.sw.bm model=$viscal.v.wide.sw.cl \
       out=$viscal.v.wide.sw.r

cgdisp in=$viscal.v.wide.sw.r,$viscal.v.wide.sw.r nxy=1,1 range=-.5,2,lin,3 \
       region=quart device=$viscal.ps/ps cols1=1 \
       type=grey,cont slev=p,5 levs1=50,15,10,5,-5 \
       options=full,noepoch,beambl csize=1,1 labtyp=arcsec
$gv $viscal.ps
```

For flux calibration, we offer two methods, depending if the flux calibrator is the same as the passband calibrator. At the end ask the user if the phasecal gains are acceptable and need to be applied later

```
if ($cal == $fluxcal) then
  rm -r $cal.wide.gain $viscal.v.wide.sw.gain $viscal.v.wide.sw.gain.applied
  # Calculating Gains on Fluxcal
  uvcat vis=$cal.wide.1 out=$cal.wide.gain options=nocal "select=win($superwidewin)"
  selfcal vis=$cal.wide.gain refant=$refant interval=$fcalint "select=source($fluxcal)" \
     options=noscale,amplitude flux=$flux
  gplist vis=$cal.wide.gain options=zeropha,amp > $fluxcal.gains
  less $fluxcal.gains

  # Calculating Gains on Phasecal
  uvcat vis=$viscal.v.wide out=$viscal.v.wide.sw.gain select='win('$superwidewin')'
  selfcal vis=$viscal.v.wide.sw.gain interval=$vcalint \
        refant=$refant options=noscale,amp flux=$visflux
  gplist vis=$viscal.v.wide.sw.gain options=zeropha,amp > $viscal.gains
  less $viscal.gains

else
  rm -r $viscal.v.wide.sw.gain $fluxcal.wide.0 $fluxcal.wide.gain $fluxcal.gains $viscal.gains
  # Fluxcal different from Pbcal"
  uvcat vis=all.wide out=$fluxcal.wide.0 \
        "select=-auto,source($fluxcal)" options=nocal,nopass
```

```
  # Passband correcting Fluxcal"
  gpcopy vis=$cal.wide.0 out=$fluxcal.wide.0 options=nocal,nopol
  uvcat vis=$fluxcal.wide.0 out=$fluxcal.wide.gain options=nocal "select=win($superwidewin)"
  # Calculating Gains on Fluxcal
  selfcal vis=$fluxcal.wide.gain refant=$refant interval=$fcalint "select=source($fluxcal)" \
    options=noscale,amplitude flux=$flux
  gplist vis=$fluxcal.wide.gain options=zeropha,amp > $fluxcal.gains
  less $fluxcal.gains

  # Calculating Gains on Phasecal
  uvcat vis=$viscal.v.wide out=$viscal.v.wide.sw.gain select='win('$superwidewin')'
  selfcal vis=$viscal.v.wide.sw.gain line=channel,$superwidechan \
     interval=$vcalint options=phase refant=$refant
  selfcal vis=$viscal.v.wide.sw.gain interval=$ampcalint \
        refant=$refant options=noscale,amp flux=$visflux
  gplist vis=$viscal.v.wide.sw.gain options=zeropha,amp > $viscal.gains
  less $viscal.gains

endif

# now ask the user if this should be applied later
echo -n "Apply Phasecal Gains to data? (y or n): " ;   set apply_gains=$<
```

Loop over each of the narrow bands and assemble the hybrid data

```
set nblength = $#nb_array
set list=(`awk "BEGIN{for(i=1;i<=$nblength;i++)print i}"`)

#  start nb loop
foreach i  ( $list )

set nb = $nb_array[$i]
set wide = $wide_array[$i]
rm -r all.hyb

# Select hybrid data
# NB:  assumes only 1 band is in wideband mode; if two bands are in wideband
#      mode, change hybrid selection to select on nb and modify bw=
if ( $wide == 1 || $wide == 4 ) then
    if ($nb == 2 || $nb == 5) then
      bwsel vis=$vis nspect=6 bw=500,$bw,0 out=all.hyb
    else
      bwsel vis=$vis nspect=6 bw=500,0,$bw out=all.hyb
    endif
endif
if ( $wide == 2 || $wide == 5 ) then
    if ($nb == 1 || $nb == 4) then
      bwsel vis=$vis nspect=6 bw=$bw,500,0 out=all.hyb
    else
      bwsel vis=$vis nspect=6 bw=0,500,$bw out=all.hyb
    endif
endif
if ( $wide == 3 || $wide == 6 ) then
    if ($nb == 1 || $nb == 4) then
      bwsel vis=$vis nspect=6 bw=$bw,0,500 out=all.hyb
    else
      bwsel vis=$vis nspect=6 bw=0,$bw,500 out=all.hyb
    endif
endif
```

Two sanity tests to make sure you have data and that the bands are present.

```
set test = `uvio vis=all.hyb | grep -i source | awk '{if (NR==1) print $4}'`
if ($test == "") then
  echo
  echo "FATAL!  There appears to be no valid data in all.hyb"
  echo "This is likely to be because wide_array[$i] = $wide is not valid"
  echo "(i.e. band $wide is not really wideband), or one of the other "
  echo "bands is not really narrowband.  Use uvindex to sort this out"
  exit 1
endif

uvlist vis=all.hyb options=spectra
```

Now we need to select single bands to process in this pass Select by source and band. First get the two bands in all-wideband mode Note that we use super-wideband calibrated file for the wide mode.

```
rm -rf $cal.win$nb* $cal.win$wide* $cal.wide.win$wide* $cal.wide.win$nb*
rm -rf $cal.hyb.win$nb* $cal.hyb.win$wide* noise.nb.win$nb*
uvcat vis=$cal.wide out=$cal.wide.win$wide "select=-auto,source($cal),win($wide)" \
options=nocal,nopass
uvcat vis=$cal.wide out=$cal.wide.win$nb "select=-auto,source($cal),win($nb)" \
options=nocal,nopass

# select hybrid wideband band
uvcat vis=all.hyb out=$cal.hyb.win$wide.0 "select=-auto,source($cal),win($wide)" \
options=nocal,nopass
# select the hybrid and all-narrowband narrow bands
# nb bands require extra step (applying noise source)
# we did not bother with noise source for wideband
uvcat vis=all.hyb out=$cal.hyb.win$nb.00 "select=-auto,source($cal),win($nb)" \
options=nocal,nopass
uvcat vis=all.nb out=$cal.nb.win$nb.00 "select=-auto,source($cal),win($nb)" \
options=nocal,nopass

# copy wideband passband determined from all-wideband mode to hybrid wideband
gpcopy vis=$cal.wide.0   out=$cal.hyb.win$wide.0 options=nocal,nopol
uvcat vis=$cal.hyb.win$wide.0  out=$cal.hyb.win$wide   options=nocal
```

Now get the noise source data. Use the noise source data obtained in all narrowband mode, and assume it also can be applied to hybrid narrowband.

```
if ($sideband == "USB" || $sideband == "usb" ) then
  rm -rf noise.lsb noise.usb
  @ lsbnb = $nb - 3
  uvcat vis=all.nb out=noise.lsb "select=-auto,source(NOISE),win($lsbnb)" \
        options=nocal,nopass
  uvcat vis=all.nb out=noise.usb "select=-auto,source(NOISE),win($nb)" \
        options=nocal,nopass
  set sdf = `uvio vis=noise.usb | grep sdf | grep DATA | awk '{print $5}'`
  set sfreq = `uvio vis=noise.usb | grep sfreq | grep DATA | awk '{if (NR==1) print $5}'`
  uvcal vis=noise.lsb out=noise.nb.win$nb.00 options=conjugate
  puthd in=noise.nb.win$nb.00/sfreq value=$sfreq type=d
  puthd in=noise.nb.win$nb.00/sdf value=$sdf type=d
else
  uvcat vis=all.nb out=noise.nb.win$nb.00 "select=-auto,source(NOISE),win($nb)" \
        options=nocal,nopass
endif
```

For the narrowband windows, first do a passband cal using the noise source

```
mfcal vis=noise.nb.win$nb.00 refant=$refant
```

```
# Passband cal using noise source
gpplt vis=noise.nb.win$nb.00 device=bpnoise.nb.win$nb.00.ps/ps options=bandpass yaxis=phase nxy=4,4 \
      yrange=-90,90
$gv bpnoise.nb.win$nb.00.ps

# Copy noise passband to astronomical all-narrowband and hybrid narrowbands, and apply
gpcopy vis=noise.nb.win$nb.00 out=$cal.nb.win$nb.00  options=nocal,nopol
gpcopy vis=noise.nb.win$nb.00 out=$cal.hyb.win$nb.00 options=nocal,nopol
uvcat vis=$cal.nb.win$nb.00 out=$cal.nb.win$nb.0 options=nocal
uvcat vis=$cal.hyb.win$nb.00 out=$cal.hyb.win$nb.0 options=nocal

# use smamfcal with 1st order polynomial to
# get passband on hybrid narrowband and copy to all narrowband
smamfcal vis=$cal.hyb.win$nb.0 line=chan,19,4,3 interval=1 refant=$refant edge=$edge options=opolyfit \
          polyfit=$order
gpplt vis=$cal.hyb.win$nb.0 options=bandpass yaxis=phase nxy=4,4 yrange=-90,90 \
    device=bp$cal.hyb.win$nb.0.ps/ps
$gv bp$cal.hyb.win$nb.0.ps

# Copy narrowband passband from hybrid to all-narrowband mode
gpcopy vis=$cal.hyb.win$nb.0    out=$cal.nb.win$nb.0 options=nocal,nopol

# check that all-narrowband passband is flat
rm -rf test.pass
uvcat vis=$cal.nb.win$nb.0 out=test.pass
mfcal vis=test.pass refant=$refant
gpplt vis=test.pass options=bandpass yaxis=phase nxy=4,4 yrange=-90,90 \
      device=bptest.ps/ps
$gv bptest.ps

# Apply astronomical narrowband passband to hybrid and all-narrowband
uvcat vis=$cal.hyb.win$nb.0    out=$cal.hyb.win$nb     options=nocal
uvcat vis=$cal.nb.win$nb.0    out=$cal.nb.win$nb     options=nocal

# Selfcal on hybrid wideband to remove temporal variations
# prior to band offset calibration
selfcal vis=$cal.hyb.win$wide line=channel,$wideline \
        interval=$calint options=phase refant=$refant

# Copy selfcal solution over to narrow hybrid band and apply
copyhd in=$cal.hyb.win$wide out=$cal.hyb.win$nb items=gains,ngains,nsols,interval
uvcat vis=$cal.hyb.win$nb out=$cal.hyb.win$nb.a

# Selfcal on narrow band of hybrid to determine band offset
selfcal vis=$cal.hyb.win$nb.a line=channel,$narrowline \
    interval=9999 options=phase refant=$refant
# Band offset between hybrid-narrowband $cal.hyb.win$nb.a
#      and hybrid-wideband $cal.hyb.win$nb
gplist vis=$cal.hyb.win$nb.a options=phase
# Also copy band offset to text file
gplist vis=$cal.hyb.win$nb.a options=phase >! mnband_offset.$cal.hybwin$nb.txt

# Test by applying to calibrator observed in all-narrowband mode
copyhd in=$cal.hyb.win$nb.a out=$cal.nb.win$nb items=gains,ngains,nsols,interval
uvcat vis=$cal.nb.win$nb out=$cal.nb.win$nb.a

# Remove antenna phase gain using super-wideband
rm -rf $cal.wide.sw
uvcat vis=$cal.wide out=$cal.wide.sw select='win('$superwidewin')'
selfcal vis=$cal.wide.sw line=channel,$superwidechan \
    interval=9999 options=phase refant=$refant
# Copy super-wideband gain to narrowband and apply
copyhd in=$cal.wide.sw out=$cal.nb.win$nb.a items=gains,ngains,nsols,interval
uvcat vis=$cal.nb.win$nb.a out=$cal.nb.win$nb.a.sc

# Selfcal to check that phases are roughly zero
# to within amount expected given temporal variations over interval
# between superwideband and all-narrowband observerations
selfcal vis=$cal.nb.win$nb.a.sc line=channel,$narrowline \
     interval=9999 options=phase refant=$refant
```

```
# List gains, which should be near zero except for temporal variations
# over interval between wideband and narrow band observations of cal
echo "**** Phase offset between super-wideband $cal.wide.sw "
echo "**** and all-narrow narrow band $cal.nb.win$nb.a.sc "
echo "**** Check that phases are near zero, limited by atmospheric flucatuations"
gplist  vis=$cal.nb.win$nb.a.sc options=phase

# Now apply calibrations to source data
rm -rf  $source.win$nb* $source.win$nb.bcal
# First select source data
uvcat vis=all.nb out=$source.win$nb.00 \
   "select=-auto,source($source),win($nb)"  options=nocal,nopass
# Copy and apply noise passband to source
gpcopy vis=noise.nb.win$nb.00 out=$source.win$nb.00 options=nocal,nopol
uvcat vis=$source.win$nb.00 out=$source.win$nb.0 options=nocal
# Copy and apply astronomical passband
gpcopy vis=$cal.hyb.win$nb.0 out=$source.win$nb.0 options=nocal,nopol
uvcat vis=$source.win$nb.0 out=$source.win$nb options=nocal
# Copy band offset to source
copyhd in=$cal.hyb.win$nb.a out=$source.win$nb items=gains,ngains,nsols,interval
rm -rf $source.win_$i
# Apply band offset using smachunkglue naming convention
uvcat vis=$source.win$nb out=$source.win_$i

# end nb loop
end
```

This end looping over the bands. The three bands can be glued back together, though the complexity below depends on how many files (bands) we had. It also flags the (bad) overlapping channels between bands that were glued together.

```
if ($nblength == 2) then
  set cfile=$source.$nb_array[1]$nb_array[2]
  rm -rf $cfile
  smachunkglue vis=$source.win nfiles=$nblength out=$cfile
  uvflag vis=$cfile line=channel,$badchan1 flagval=flag
else if ($nblength == 3) then
  set cfile=$source.$nb_array[1]$nb_array[2]$nb_array[3]
  rm -r $cfile
  smachunkglue vis=$source.win nfiles=$nblength out=$cfile
  # flag bad overlap channels
  uvflag vis=$cfile line=channel,$badchan1 flagval=flag
  uvflag vis=$cfile line=channel,$badchan2 flagval=flag
else
  set cfile=$source.$nb_array[1]
  rm -rf $cfile
  uvcat vis=$source.win_$nblength[1] out=$cfile
endif

# put in restfreq, using UV override principle
puthd in=$cfile/restfreq type=double value=$restfreq
```

To apply we are using a handy little c-shell alias:

```
rm -rf tmptmp.mir
alias apply 'uvcat vis=\!* out=tmptmp.mir; rm -rf \!*; mv tmptmp.mir \!*'
```

Phase and Amp calibration can now commence, if it was so selected earlier:

```
if ($apply_gains == 'y') then
  # copy super-wideband gains to source

  # Apply Phase Gains
  gpcopy vis=$viscal.v.wide.sw out=$cfile options=nopol,nopass
  #apply $cfile

  # Apply amplitude calibration

  rm -r medianflux
  #  Apply Amplitude Gains from Phasecal: $viscal
  #gpcopy vis=$viscal.v.wide.sw.gain out=$cfile options=nopol,nopass
  set medianflux = `grep Medians $viscal.gains | tr -d Medians:`
  echo $medianflux > medianflux
  gplist vis=$cfile options=replace jyperk=@medianflux
  apply $cfile
endif


if ($apply_gains == 'n' && $cal == $fluxcal) then
  # copy super-wideband gains to source

  # Apply Phase Gains
  gpcopy vis=$viscal.v.wide.sw out=$cfile options=nopol,nopass

  rm -r medianflux
  # Apply Amplitude Gains from Passband Cal: $cal
  set medianflux = `grep Medians $fluxcal.gains | tr -d Medians:`
  echo $medianflux > medianflux
  gplist vis=$cfile options=replace jyperk=@medianflux
  apply $cfile
endif


if ($apply_gains == 'n' && $cal != $fluxcal) then
  # copy super-wideband gains to source

  # Apply Phase Gains
  gpcopy vis=$viscal.v.wide.sw out=$cfile options=nopol,nopass

  rm -r medianflux
  # Apply Amplitude Gains from Fluxcal: $fluxcal
  set medianflux = `grep Medians $fluxcal.gains | tr -d Medians:`
  echo $medianflux > medianflux
  gplist vis=$cfile options=replace jyperk=@medianflux
  apply $cfile
endif
```

3.5.4 Calibration - IV

This script goes through a data reduction in MIRIAD of CARMA data. It can be modified in order to fit the specifics of various observations - depending on what needs to be flagged, which calibrator should be the passband and flux calibrator, etc.

```
# ------------------------------------------------------------
# [B]  Overview of Millimeter Wavelength radio data reduction
# ------------------------------------------------------------
# There are only a couple basic steps that must be done
# (0) baseline solutions - apply if online ones were wrong/not applied
#                          Needed if slope seen in baseline-baseline
#                          pairs.
# (1) bandpass/passband calibration
#                         -accomplish this with task mfcal
#                         -bootflux to scale the calibrator's amplitude
#                          or a more involved FLUX calibration here
# Next two steps create GAINS tables that can be applied to the source
# (2) phase calibration
# (3) amplitude calibration
#                         -self-cal on the calibrator
#
# (4) And Magic - the Fourier transform!
#                         - task "invert" to create the map
#
# The rest of the commands are simply to apply calibration
# to the desired source/calibrator, to look at the data,
# etc.
#
# (5) Flux calibration (see step 1)
```

Set variables for the script

```
set vis           = ct012.arp193.1.miriad
set refant        = 9
set source_name   = Arp193
set bandpass_name = 3C273
set phasecal_name = 3C273
set fluxcal_name  = 3C279
set outfile       = ct012_arp193_feb5
set antpos        = antpos.070115
```

```
# The following commands selects only the non-auto-correlation data
# and re-writes to the file data.mir

uvcat vis=$vis select=-auto out=data_auto.mir
```

Baseline Correction applied. In this example, data from June 9, 2007 were used and needd a baseline correction.

```
# See also: http://cedarflat.mmarray.org/observing/baseline/antpos.070509

if (0) then
```

```
  uvedit vis=data_auto.mir apfile=antpos.070115 out=baseline_data.mir
else
  cp -r data_auto.mir baseline_data.mir
endif




#-----------------------------------------------------------
# Rest Frequency Correction
#-----------------------------------------------------------
# Do a uvlist vis=xxx.mir options=spc to see the rest frequency
# and starting frequency of each channel.  To put in the
# proper rest frequency, do the following:
#
# uvputhd in=xxx.mir hdvar=restfreq varval=myrestfreq out=yyy.mir

uvputhd vis=baseline_data.mir hdvar=restfreq varval=225.282 out=data.mir




# ----------------------------------------------------------
# [E]  Preliminary Examination of Data
# ----------------------------------------------------------
# Here we look only at the calibrators to just check on weather,
# system temp

if ($4 == <2) then
   uvindex vis=data.mir log=uvindex.log   # scans uvdata file, keywords: vis, interval, refant, log, options
   listobs vis=data.mir log=listobs.log
   uvlist vis=data.mir options=spectra log=uvlist.log
   # What other variables are possible to plot?
   smavarplt vis=data.mir device=systemp.ps/cps nxy=2,3 yaxis=systemp options=compress  # removing compress prints all ba
   smavarplt vis=data.mir device=/xs nxy=2,3 yaxis=systemp options=compress yrange=0,1000
   uvlist vis=data.mir options=spec
   uvflag vis=data.mir options=noapply flagval=flag #how many flags applied?
   #uvplt axis variables: time, dtime, amplitude, real, imag, phase, uu, vv, uvdistance, uvangle, hangle, dhangle, parang
   smauvplt vis=data.mir device=/xs axis=time,phase select='-source(Arp193),-source(noise)' #line=wide,1,1
   smauvplt vis=data.mir device=/xs axis=time,amp select='-source(Arp193),-source(noise)' line=wide,1,1
   # closure vis=$vis line=wide,1,1,1  # to look at closure solutions
   # options=nowrap # to make plots not wrap - such as phase

   # Putting the correct rest frequency into the header
   #uvputhd vis=data.mir hdvar=restfreq type=d varval=226.434 out=data2.mir
endif




# ----------------------------------------------------------
# [F]  Flagging noticeably Bad Data
# ----------------------------------------------------------
# options=noapply # does not apply to data

# Determine what needs to be flagged by carefully examining the
# data.  Look at Tsys vs. time, Amplitudes and phases versus
# time, and the pointing.

# OVROs and C12 are out
uvflag vis=data.mir flagval=flag "select=ant(1,2,3,4,5,6,12)"


# This step does not seem necessary for this data set, flagging based
# on pointing, because examining data with smavarplt, yaxis=axisrms,
# I get that the absolute magnitude of variations never exceeds 1.
# BAZ - 7/13/2007
uvflag vis=data.mir flagval=flag "select=-pointing(0,3),-source(NOISE)"
```

```
# To flag antennas during a certain time range
#     Flagging all during these three minutes of bandpass observation
#      because there is a peak in amplitudes - triangular.
#uvflag vis=data.mir flagval=flag "select=time(07:15:00,07:18:00)"

# If I don't get rid of all OVROs, I have to do something special with the
# differing beam sizes if and only if I am doing a mosaic

# Unflag following to auto-flag bad antennas
#set badants = ""                    # bad antennas to flag
#uvflag vis=$vis select=anten'('$badants')' flagval=flag

# noticing in uvplt that baseline 12-14 and last scan is bad
#uvflag vis=data.mir flagval=flag "select=ant(12)(14)"
#uvflag vis=data.mir flagval=flag "select=time(11:00:00,11:15:00)"
#uvflag vis=data.mir flagval=flag "select=amp(30)"




#-----------------------------------------------------------------
# [G]  Data Handling
# ----------------------------------------------------------------
# Split up the data.  One can also leave it all together and use
# the appropriate select commands while executing various tasks.
# Or one can use the mega split program, ProjectExplode to separate
# by window and source.

uvcat vis=data.mir "select=source("$source_name")"  out=source.mir
uvcat vis=data.mir "select=source("$bandpass_name")"   out=bandpass.mir
uvcat vis=data.mir "select=source("$phasecal_name")"   out=phasecal.mir
uvcat vis=data.mir "select=source("$fluxcal_name")"  out=fluxcal.mir
# fluxcal testing testing
set fluxvis=bpcalib_fluxcal
set phasevis=fluxstep_phasecal
uvcat vis=data.mir "select=source("$fluxcal_name")" out=$fluxvis.mir
uvcat vis=data.mir "select=source("$phasecal_name")" out=$phasevis.mir



# ----------------------------------------------------------
# [H]  STEP 1 -> Bandpass/Passband calibration
# ----------------------------------------------------------

# super-wideband mfcal passband - 1 minute interval
# I choose a 1 minute interval here because I have observed my bandpass
# for a total of 10 minutes.  And why?
mfcal vis=bandpass.mir interval=1 refant=$refant

# look at the bandpass data
if ($1 < 2) then
   gpplt vis=bandpass.mir options=bandpass yaxis=phase nxy=3,2 yrange=-360,360 device=/xs
   gpplt vis=bandpass.mir options=bandpass yaxis=phase nxy=3,2 yrange=-360,360 device=bandpassSolution.ps/ps
   uvspec vis=bandpass.mir interval=15 options=nopass device=/xs nxy=3,3 axis=channel,amplitude
   uvspec vis=bandpass.mir interval=15 device=/xs nxy=3,3 axis=channel,amplitude
   uvspec vis=bandpass.mir interval=1000 device=/xs nxy=3,3 axis=channel,phase yrange=-180,180
endif

# copy and apply bandpass calibrator solution to phase calibrator
#  -no gain calibration
gpcopy vis=bandpass.mir out=phasecal.mir options=nocal
uvcat vis=phasecal.mir options=nocal out=phasecal_bp.mir

# copy and apply bandpass calibrator solution to the source, no cal
gpcopy vis=bandpass.mir out=source.mir options=nocal
uvcat vis=source.mir out=source_bp.mir

# copy and apply bandpass calibrator to the flux calibrator
```

```
gpcopy vis=bandpass.mir out=fluxcal.mir options=nocal
uvcat vis=fluxcal.mir out=fluxcal_bp.mir

# making second copy to have for the super calibrator sandwich try
uvcat vis=source.mir out=source2_bp.mir




#-----------------------------------------------------------------
# [H (continued)] STEP 1b -> Flux calibration
#-----------------------------------------------------------------
# Cleaning up before starting this section again
rm -rf *.gain *.gains *.sw *.wide *.flux *.medians

# Perhaps set these ahead of time
#set flux = 8.4
set flux = 12.03
set calint = 0.2
set vcalint = 18
set fcalint = 1
set superwidewin = "1,2,3,4,5,6"
set superwidechan = "1,1,90"
set lsbfluxchan = "1,1,45,45"
set usbfluxchan = "1,46,45,45"


# Following all done using test files.  No work done on the main files, yet.
mfcal vis=$fluxvis.mir interval=$calint refant=$refant

# Examining the data vs. freq and time
gpplt vis=$fluxvis.mir options=bandpass yaxis=phase nxy=3,2 yrange=-360,360 device=/xs
gpplt vis=$fluxvis.mir yaxis=phase nxy=3,2 yrange=-360,360 device=/xs

# Copying this derived bp solution from the flux calibrator to our phase calibrator
gpcopy vis=$fluxvis.mir out=$phasevis.mir options=nocal,nopol
uvcat vis=$phasevis.mir out=$phasevis.wide options=nocal

uvcat vis=$fluxvis.mir out=$fluxvis.gain options=nocal "select=win($superwidewin)"
selfcal vis=$fluxvis.gain refant=$refant interval=$fcalint "select=source($fluxcal_name)" options=noscale,amplitude,aprio
gplist vis=$fluxvis.gain options=zeropha,amp > $fluxvis.gains

# Unix pipe to get median values
grep Medians $fluxvis.gains | tr -d Medians: > $fluxvis.medians
cat $fluxvis.medians

# straightening out the phase of the phase calibrator - use a phase only selfcal with
# a fairly long integration time
uvcat vis=$phasevis.wide out=$phasevis.sw "select=win($superwidewin)"
selfcal vis=$phasevis.sw line=channel,$superwidechan interval=$vcalint options=phase refant=$refant

# Now the amplitude gains derived from the flux calibrator can be applied to the phase calibrator
# by replacing the amplitudes and keeping the phases determined from the selfcal solution

gplist vis=$phasevis.sw options=replace jyperk=@$fluxvis.medians

#The following special program, uvflux, can gather statistics on this phase calibrator
uvflux vis=$phasevis.sw options=nopol line=chan,$lsbfluxchan
uvflux vis=$phasevis.sw options=nopol line=chan,$usbfluxchan
uvflux vis=$phasevis.sw options=nopol > $phasevis.flux

# This value obtained by averaging the results form the LSB and USB above.
set visflux=11.26

echo "come back to this part"
# Finally, checking the time variance of the phase calibrator
echo "Checking the time variance of the phase calibrator"
uvcat vis=$phasevis.wide out=$phasevis.wide.gain
selfcal vis=$phasevis.wide.gain refant=$refant interval=$vcalint "select=source($phasecal_name)" \
        options=noscale,amplitude flux=$visflux
```

```
gplist vis=$phasevis.wide.gain options=zeropha,amp > $phasevis.gains




# ------------------------------------------------------------------
# [I]  STEPS 2 & 3 -> Amplitude & Phase Calibration
# ------------------------------------------------------------------
# Create the GAINS tables

# selfcal
# You want the interval to be about equal to source-calibrator cycle
# Use gplist to look at the time intervals (vis=file options=amp or phase)
selfcal vis=phasecal_bp.mir interval=13 refant=$refant
#selfcal vis=mastercalib_bp.mir interval=13 refant=13

gpplt vis=phasecal_bp.mir device=/xs yaxis=phase yrange=-720,720
gpplt vis=phasecal_bp.mir device=gainSolutions.ps/ps yrange=-180,180

# Copy self cal solution to
# SOURCE
# copy selfcal solution to source
gpcopy vis=phasecal_bp.mir out=source_bp.mir options=nopass mode=copy


echo "************************"
echo "*** Apply flux gains to phase cal***"
grep Medians $phasevis.gains | tr -d Medians: > $phasevis.medians
cat $phasevis.medians
gplist vis=phasecal_bp.mir options=replace jyperk=@$phasevis.medians


echo "************************"
echo "********Applying phase gains to source*********"
grep Medians $phasevis.gains | tr -d Medians: > $phasevis.medians
cat $phasevis.medians
gplist vis=source_bp.mir options=replace jyperk=@$phasevis.medians



# play with calibrator
# cleaning up
rm -rf phasecal.mp phasecal.bm phasecal.sl phasecal.cm phasecal.fits
#linetype,nchan,start,width,step  # step=width if you don't specify.
# Try the defaults   # options=systemp (will weight by the inverse of the system temp.)
invert vis=phasecal_bp.mir map=phasecal.mp beam=phasecal.bm cell=.33 imsize=512 line=channel,1,1,90
#invert vis=weakcal_bp.mir map=weakcal.mp beam=weakcal.bm cell=.33 imsize=512 line=channel,1,1,90
clean map=phasecal.mp beam=phasecal.bm out=phasecal.sl niters=1000
restor map=phasecal.mp beam=phasecal.bm model=phasecal.sl out=phasecal.cm

# look at some stats
imstat in=phasecal.cm region=quarter
imstat in=phasecal.cm region=box'(180,180,200,200)'
ellint in=phasecal.cm
# to look at the uv coverage

fits in=phasecal.cm op=xyout out=phasecal.fits




# ------------------------------------------------------------------
# [J]  FINAL step - invert
# ------------------------------------------------------------------
# (1)
# Check calibration on the weak calibrator
# copy bandpass solution to the weak calibrator
```

```
#gpcopy vis=bandpass.mir out=weakcal.mir options=nocal
#uvcat vis=weakcal.mir out=weakcal_bp.mir

# copy selfcal solution to weak calibrator
#gpcopy vis=phasecal_bp.mir out=weakcal_bp.mir options=nopass mode=copy

#rm -rf weakcal.mp weakcal.bm weakcal.sl weakcal.cm weakcal.fits
#invert vis=weakcal_bp.mir map=weakcal.mp beam=weakcal.bm cell=0.2 imsize=512
#clean map=weakcal.mp beam=weakcal.bm out=weakcal.sl niter=1000
#restor map=weakcal.mp beam=weakcal.bm model=weakcal.sl out=weakcal.cm

# (2)
# play with the source
rm -rf $source_name.mp $source_name.bm $source_name.sl $source_name.cm $source_name.fits $outfile.cm
#invert vis=source_bp.mir map=arp220.mp beam=arp220.bm cell=0.2 imsize=512 line=channel,1,1,90
invert vis=source_bp.mir map=Arp193.mp beam=Arp193.bm cell=0.2 imsize=512 line=channel,1,1,90 sup=0
#ine=velocity,30,-4.617,43.450
clean map=Arp193.mp beam=Arp193.bm out=Arp193.sl niters=10000
restor map=Arp193.mp beam=Arp193.bm model=Arp193.sl out=$outfile.cm

# Creating the cube
rm -rf $source_name.cube.mp $source_name.cube.bm $source_name.cube.sl $source_name.cube.cm $source_name.cube.fits $outfil
#invert vis=source_bp.mir map=arp220.mp beam=arp220.bm cell=0.2 imsize=512 line=channel,1,1,90
invert vis=source_bp.mir map=Arp193.cube.mp beam=Arp193.cube.bm cell=0.2 imsize=512 line=channel,90,1,1 sup=0
#ine=velocity,30,-4.617,43.450
clean map=Arp193.cube.mp beam=Arp193.cube.bm out=Arp193.cube.sl niters=10000
restor map=Arp193.cube.mp beam=Arp193.cube.bm model=Arp193.cube.sl out=$outfile.cube.cm


#fits in=M80.cm op=xyout out=M80.fits

#rm -rf arp193_2.mp arp193_2.bm arp193_2.sl arp193_2.cm arp193_2.fits
#invert vis=source2_bp.mir map=arp193_2.mp beam=arp193_2.bm cell=0.2 imsize=512
#clean map=arp193_2.mp beam=arp193_2.bm out=arp193_2.sl niter=1000
#restor map=arp193_2.mp beam=arp193_2.bm model=arp193_2.sl out=arp193_2.cm

#fits in=arp193_2.cm op=xyout out=arp193_2.fits

# Then look at the final image in ds9 or some other FITS format
# viwer.  Statistics can be examined, etc.

# ------------------------------------------------------------------
# [K]  DS9   Viewing Notes
# ------------------------------------------------------------------
# You can look at the *.*m maps in ds9 by doing a
# mirds9 <filename>
# once ds9 is open.
#      FRAME -> TILE to plot more than 1
#      FRAME -> BLINK to blink back and forth.




# ------------------------------------------------------------------
# [L] Misc. Notes for LATER
# ------------------------------------------------------------------
#
# For CARMA array:
# note if you have 3 beam sizes
# mospsf
# imfit
# so restor does not use first beam size and apply for all
# (this is not an issue for Arp 220 on 25 Apr 2007 because I only
#  had BIMA dishes)
```

3.6 Scripting

3.6.1 Interactive shells

Miriad vs. Unix. Miriad shell uses `save/load` and `tput/tget` commands, and if properly installed the *readline* library. Unix shells have different command history recall.

3.6.2 Programmable shells

Shell (csh/sh/bash) programming vs. python (pyramid)

3.6.3 Example: mosaic.py

Here is an example of a mosaic script, using `pyramid` procedures.

```
 1: #!/usr/bin/env python
 2: #
 3: #  History:
 4: #  june 02 mchw. ALMA script.
 5: #  15aug02 mchw. CARMA version edited from ALMA script.
 6: #  23aug02 mchw. calculate region from source size.
 7: #  20sep02 mchw. Re-import CARMA improvements for ALMA.
 8: #  25sep02 mchw. Re-import improvements from hex7.csh to hex19.csh
 9: #  26sep02 mchw. Increase imsize from 129 to 257.
10: #  12mar03 mchw. convert to PYTHON.
11: #  13mar03 pjt   more conversion to PYTHON, now at 200ft, renamed to mosaic.py
12:
13: import sys, os, time, string, math
14: from Miriad import *
15:
16: version='2003-03-14'
17:
18: print "   ---  ALMA Mosaicing (Cas A model)   ---   "
19:
20: # command line arguments that can be changed...
21: keyval = {
22:      "config"  : "config1",           # antenna config file (without the .ant extension)
23:      "dec"     : "-30",               # declination (can be a real number)
24:      "image"   : "casc.vla",          # image to test (nice Cas-A VLA image as default)
25:      "cell"    : "0.04",              # scale size (should be calculated from image)
26:      "nchan"   : "1",                 # number of channels
27:      "method"  : "mosmem",            # mosmem, joint, or default
28:      "flux"    : "732.063",           # expected flux in the image (for mosmem)
29:      "nring"   : "3",                 # number of rings in the mosaic
30:      "grid"    : "12.0",              # gridsize (in arcsec) for the mosaic
31:      "center"  : "",                  # optional center file that overrides (nring,grid)
32:      "VERSION" : "1.0 mchw"           # VERSION id for the user interface
33:      }
34:
35: help = """
36: The minimum amount of information you need to run this task is:
37:    a miriad image (image=) for the model.
38:    an antenna configuration file (<config>.ant) for uvgen
39: """
40:
41: keyini(keyval,help,0)
42: #                               report current defaults, exit if --help given
43: setlogger('mosaic.log')
44: #
45: # ------------------------------------------------------------------------
46:
47: #
```

```
48:  # define all variables, now in their proper type, for this script
49:  #
50:
51:  config  = keya('config')
52:  dec     = keyr('dec')
53:  cell    = keyr('cell')
54:  nchan   = keyi('nchan')
55:  method  = keya('method')
56:  center  = keya('center')
57:  flux    = keyr('flux')
58:  image   = keya('image')
59:  nring   = keyi('nring')
60:  grid    = keyr('grid')
61:
62:  harange = '-1,1,0.013'
63:  select  = '-shadow\(12\)'
64:  freq    = 230.0
65:  imsize  = 257                    # avoid 2**N, image size 2**N + 1 is good.  [or calculate from image]
66:
67:  mir = os.environ['MIR']
68:
69:  # ------------------------------------------------------------------------------
70:
71:  #   returns a list of strings that are the ascii centers as uvgen wants them
72:  #   (in a file) via the center= keyword
73:  def hex(nring,grid):
74:      center=""
75:      npoint=0
76:      for row in range(-nring+1,nring,1):
77:          y = 0.866025403 * grid * row
78:          lo = 2-2*nring+abs(row)
79:          hi = 2*nring-abs(row)-1
80:          for k in range(lo,hi,2):
81:              x = 0.5*grid*k
82:              npoint = npoint + 1
83:              if center=="":
84:                  center = center + "%.2f,%.2f" % (x,y)
85:              else:
86:                  center = center + ",%.2f,%.2f" % (x,y)
87:      return (npoint,center)
88:
89:  #   get the (as a string) value of an item in a dataset
90:  def itemize(data,item):
91:      log = 'tmp.log'
92:      cmd = [
93:          'itemize',
94:          'in=%s/%s' % (data, item),
95:          'log=%s' % log
96:          ]
97:      miriad(cmd)
98:      f = open(log,"r")
99:      v = string.split(f.readline())
100:     f.close()
101:     return v[2]
102:
103: #   copy a file from source (s) to destination (d)
104: def copy_data(s,d):
105:     zap(d)
106:     os.system("cp -r %s %s" % (s,d))
107:
108:
109: #  should this be
110: #    units=None
111: #    if units is None:
112: #        bla
113: #    else:
114: #        bla
115:
116: def puthd(map,item,value,units=0):
117:     cmd = [
```

```
118:          'puthd',
119:          'in=%s/%s' % (map,item),
120:          ]
121:      if (units == 0):
122:          cmd.append("value=%s" % value)
123:      else:
124:          cmd.append("value=%s,%s" % (value,units))
125:      return cmd
126:
127:  def demos(map,vis,out):
128:      cmd = [
129:          'demos',
130:          'map=%s' % map,
131:          'vis=%s' % vis,
132:          'out=%s' % out,
133:          ]
134:      zap_all(out+"*")
135:      return cmd;
136:
137:  def invert(vis,map,beam,imsize,select):
138:      cmd = [
139:       'invert',
140:       'vis=%s' % vis,
141:       'map=%s' % map,
142:       'beam=%s' % beam,
143:       'imsize=%d' % imsize,
144:       'select=%s' % select,
145:       'sup=0',
146:       'options=mosaic,double',
147:       ]
148:      zap(map)
149:      zap(beam)
150:      return cmd
151:
152:  def cgdisp(map):
153:      cmd = [
154:          'cgdisp',
155:          'in=%s' % map,
156:          'device=/xs',
157:          'labtyp=arcsec',
158:          'range=0,0,lin,8'
159:          ]
160:      return cmd;
161:
162:  def uvmodel(vis,model,out):
163:      cmd = [
164:          'uvmodel',
165:          'vis=%s' % vis,
166:          'model=%s' % model,
167:          'out=%s' % out,
168:          'options=add,selradec'
169:          ]
170:      zap(out)
171:      return cmd;
172:
173:  def implot(map,region='quarter'):
174:      cmd = [
175:          'implot',
176:          'in=' + map,
177:          'device=/xs',
178:          'units=s',
179:          'conflag=l',
180:          'conargs=1.4',
181:          'region=%s' % region
182:          ]
183:      return cmd
184:
185:  def imlist(map):
186:      cmd = [
187:          'imlist',
```

```
188:          'in=' + map,
189:          'options=mosaic'
190:          ]
191:      return cmd
192:
193:
194:  def imgen(map,out,pbfwhm):
195:      cmd = [
196:          'imgen',
197:          'in=%s' % map,
198:          'out=%s' % out,
199:          'object=gaussian',
200:          'factor=0',
201:          'spar=1,0,0,%g,%g' % (pbfwhm,pbfwhm)
202:          ]
203:      zap(out)
204:      return cmd
205:
206:
207:  def mosmem(map,beam,out,region,flux=0,default=0):
208:      cmd = [
209:          'mosmem',
210:          'map=%s' % map,
211:          'beam=%s' % beam,
212:          'out=%s' % out,
213:          'region=%s' % region,
214:          'rmsfac=200,1',
215:  #        'niters=200'
216:          'niters=2'
217:          ]
218:      if flux != 0:
219:          cmd.append('flux=%g' % flux)
220:      if default != 0:
221:          cmd.appenx('default=%s' % default)
222:      zap(out)
223:      return cmd
224:
225:  def restor(map,beam,model,out):
226:      cmd = [
227:          'restor',
228:          'model=%s' %  model,
229:          'map=%s' % map,
230:          'beam=%s' % beam,
231:          'out=%s' % out
232:          ]
233:      zap(out)
234:      return cmd
235:
236:  def regrid(map,tin,out):
237:      cmd = [
238:          'regrid',
239:          'in=%s' % map,
240:          'tin=%s' % tin,
241:          'out=%s' % out,
242:          'axes=1,2'
243:          ]
244:      zap(out)
245:      return cmd
246:
247:  def convol(map,out,b1,b2,pa):
248:      cmd = [
249:          'convol',
250:          'map=%s' % map,
251:          'out=%s' % out,
252:          'fwhm=%g,%g' % (b1,b2),
253:          'pa=%g' % pa
254:          ]
255:      zap(out)
256:      return cmd
257:
```

```
258:  def imframe(map,out):
259:      cmd = [
260:          'imframe',
261:          'in=%s' % map,
262:          'out=%s' % out,
263:          'frame=-1024,1024,-1024,1024'           # TODO:: the 1024 here depends on the input image size
264:          ]
265:      zap(out)
266:      return cmd
267:
268:  def uvgen(ant,dec,harange,freq,nchan,out,center):
269:      cmd = [
270:          'uvgen',
271:          'ant=%s' % ant,
272:          'baseunit=-3.33564',
273:          'radec=23:23:25.803,%g' % dec,
274:          'lat=-23.02',
275:          'harange=%s' % harange,
276:          'source=$MIRCAT/point.source',
277:          'telescop=alma',
278:          'systemp=40',
279:          #        'pnoise=30',
280:          'jyperk=40',
281:          'freq=%g' % freq,
282:          'corr=%d,1,0,8000' % nchan,
283:          'out=%s' % out,
284:          'center=%s' % center                     # notice we don't use a file, but a string of numbers
285:          ]
286:      zap(out)
287:      return cmd
288:
289:  def imdiff(in1,in2,resid):
290:      cmd = [
291:          'imdiff',
292:          'in1=%s' % in1,
293:          'in2=%s' % in2,
294:          'resid=%s' % resid,
295:          'options=nox,noy,noex'
296:          ]
297:      zap(resid)
298:      return cmd
299:
300:  def histo(map,region):
301:      cmd = [
302:          'histo',
303:          'in=%s' % map,
304:          'region=%s' % region
305:          ]
306:      return cmd
307:
308:  # ============================================================================
309:  #
310:  #   start of the actual script
311:  # ----------------------------------------------------------------------------
312:  # Nyquist sample rate for each pointing.
313:  # calc '6/(pi*250)*12'
314:  cells  = 500*cell
315:  region = "arcsec,box\(%.2f,-%.2f,-%.2f,%.2f\)" % (cells,cells,cells,cells)
316:
317:  ant    = config + '.ant'                  # antenna file for uvgen
318:  uv     = config + '.uv'                   # dataset for visibilities
319:
320:  demos1 = "%s.cas.%g.demos"  % (config,cell)
321:  base1  = "%s.%g.cas.%g"     % (config,dec,cell)
322:  base2  = "single.%g.cas.%g" % (dec,cell)
323:
324:  map1   = base1 + ".mp"
325:  beam1  = base1 + ".bm"
326:  map2   = base2 + ".map"
327:  beam2  = base2 + ".beam"
```

```
328:  mem    = base1 + ".mem"
329:  cm     = base1 + ".cm"
330:  mp     = base1 + '.mp'
331:  res    = base1 + '.resid'
332:  conv   = base1 + '.conv'
333:
334:  if center == "":
335:      (npoint,center) = hex(nring,grid)
336:      print "MOSAIC FIELD, using hexagonal field with nring=%d and grid=%g (%d pointings) " % (nring,grid,npoint)
337:  else:
338:      centerfile = center
339:      f = open(centerfile,"r")
340:      center=f.read()
341:      f.close()
342:      npoint = len(string.split(center,","))-1
343:      center=string.replace(center,'\n',',')
344:      print "MOSAIC FIELD, using center file %s (%d pointings) " % (centerfile,npoint)
345:
346:  print "   ---  ALMA Mosaicing (Cas A model)   ---   "
347:
348:  print " config  = %s" % config
349:  print " dec     = %g" % dec
350:  print " scale   = %g" % cell
351:  print " harange = %s  hours" % harange
352:  print " select  = %s" % select
353:  print " freq    = %g" % freq
354:  print " nchan   = %d" % nchan
355:  print " imsize  = %d" % imsize
356:  print " region  = %s" % region
357:  print " method  = %s" % method
358:  print " "
359:  print "   ---  TIMING   ---   "
360:
361:  if method == "mosmem":
362:      print "Generate mosaic grid"
363:      #  lambda/2*antdiam (arcsec)
364:      print 300/freq/2/12e3*2e5
365:
366:      print "Generate uv-data. Tsys=40K, bandwidth=8 GHz "
367:      miriad(uvgen(ant,dec,harange,freq,nchan,uv,center))
368:      os.system('uvindex vis=%s' % uv)
369:
370:      print "Scale model size from pixel 0.4 to %g arcsec" % cell
371:      # with 0.4 arcsec pixel size Cas A is about 320 arcsec diameter; image size 1024 == 409.6 arcsec
372:      # scale model size. eg. cell=0.1 arcsec -> 80 arcsec cell=.01 -> 8 arcsec diameter
373:
374:      copy_data(image,base2)
375:
376:      miriad(puthd(base2,'crval2',dec,units='dms'))
377:      miriad(puthd(base2,'crval3',freq))
378:      miriad(puthd(base2,'cdelt1',-cell,units='arcsec'))
379:      miriad(puthd(base2,'cdelt2',cell,units='arcsec'))
380:
381:      print "Make model images for each pointing center"
382:      miriad(demos(base2,uv,demos1))
383:
384:      print "Make model uv-data using VLA image of Cas A as a model (the model has the VLA primary beam)"
385:      for i in range(1,npoint+1):
386:          miriad(cgdisp(demos1+"%d"%i))
387:          vis_i = base1+".uv%d"%i
388:          demos_i = demos1+"%d"%i
389:          miriad(uvmodel(uv,demos_i,vis_i))
390:          if i==1:
391:              vis_all=vis_i
392:          else:
393:              vis_all=vis_all + ',' + vis_i
394:          print "UVMODEL: add the model to the noisy sampled uv-data"
395:
396:      miriad(invert(vis_all, base1+".mp", base1+".bm", imsize, select))
397:
```

```
398:      print "INVERT: "
399:
400:      miriad(implot(base1+'.mp',region=region))
401:      miriad(imlist(base1+'.mp'))
402:
403:      print "Make single dish image and beam"
404:
405:      pbfwhm = string.atof(grepcmd("pbplot telescop=alma freq=%g" % freq, "FWHM", 2)) * 60.0
406:
407:      print "Single dish FWHM = %g arcsec at %g GHz" % (pbfwhm,freq)
408:
409:      miriad(imframe(base2,base2+".bigger"))
410:      miriad(convol(base2+".bigger",base2+".bigger.map",pbfwhm,pbfwhm,0.0))
411:      miriad(regrid(base2+".bigger.map",base1+'.mp',base2+".map"))
412:      miriad(imgen(base2+".map",base2+".beam",pbfwhm))
413:      miriad(implot(base2+".map"))
414:      miriad(puthd(base2+".map",'rms','7.32'))            #  is that 1/100 of the flux ???
415:
416:  if method=='mosmem':
417:      print " MOSMEM Interferometer only"
418:      print " MOSMEM Interferometer only with niters=200 flux=%g rmsfac=200." % flux
419:      miriad(mosmem(map1,beam1,mem,region,flux=flux))
420:  elif method=='joint':
421:      print "Joint deconvolution of interferometer and single dish data"
422:      print "Joint deconvolution of interferometer and single dish data ; niters=200 rmsfac=200,1"
423:      miriad(mosmem(map1+','+map2,beam1+','+beam2,mem,region))
424:  elif method=='default':
425:      print "MOSMEM with default single dish image"
426:      print "MOSMEM with default single dish image; niters=200 rmsfac=200"
427:      miriad(mosmem(map1,beam1,mem,region))
428:  else:
429:      print "Unknown method " + method
430:
431:  miriad(restor(map1,beam1,mem,cm))
432:  miriad(implot(map1,region=region))
433:
434:  print "convolve the model by the beam and subtract from the deconvolved image"
435:  b1 = string.atof(grepcmd('prthd in=%s' % cm, 'Beam', 2))
436:  b2 = string.atof(grepcmd('prthd in=%s' % cm, 'Beam', 4))
437:  b3 = string.atof(grepcmd('prthd in=%s' % cm, 'Position', 2))
438:
439:  miriad(convol(base2,base1+'.conv',b1,b2,b3))
440:  miriad(implot(base1+'.conv',region=region))
441:
442:  print "regrid the convolved model to the deconvolved image template"
443:
444:  miriad(regrid(base1+".conv",base1+".cm",base1+'.regrid'))
445:  miriad(implot(base1+'.regrid',region=region))
446:
447:  #  skipping cgdisp /gif production
448:
449:  miriad(imdiff(base1+'.cm',base1+'.regrid',base1+'.resid'))
450:  miriad(implot(base1+'.resid',region=region))
451:  miriad(histo(base1+'.resid',region=region))
452:
453:  # ================================================================================
454:
455:  print "print out results - summarize rms and beam sidelobe levels"
456:  print "   ---  RESULTS   ---   "
457:
458:  #   extract information, the hard way
459:
460:  #   BUG: doesn't look like 'mp' has rms???
461:  #rms    = string.atof(itemize(mp,'rms')) * 1000
462:  rms = -1
463:  srms   = string.atof(grepcmd('histo in=%s' % res, 'Rms', 3))
464:  smax   = string.atof(grepcmd('histo in=%s' % res, 'Maximum', 2))
465:  smin   = string.atof(grepcmd('histo in=%s' % res, 'Minimum', 2))
466:  Model_Flux = string.atof(grepcmd('histo in=%s region=%s' % (conv,region),'Flux',5))
467:  Model_Peak = string.atof(grepcmd('histo in=%s region=%s' % (conv,region),'Maximum',2))
```

```
468:  Flux       = string.atof(grepcmd('histo in=%s region=%s' % (cm,region),'Flux',5))
469:  Peak       = string.atof(grepcmd('histo in=%s region=%s' % (cm,region),'Maximum',2))
470:  Fidelity   = Peak/srms
471:
472:  print " Config  DEC  HA[hrs]    Beam[arcsec] scale Model_Flux,Peak  Image_Flux,Peak Residual:Rms,Max,Min[Jy] Fidelit
473:  print " %s %g %s %.3f %g %g %g %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f" % (config,dec,harange,rms,b1,b2,
474:                                                                            cell,Model_Flux,Model_Peak,Flux,Peak,srms,smax,smin,Fi
475:
476:  #mv timing hex19.$config.$dec.$harange.$nchan.$imsize
477:  #cat $config.$dec.$harange.$nchan.$imsize
478:  #cat casa.results
479:  #enscript -r casa.results
480:
481:  #print "DEBUGGING"
482:  #string.atof(itemize(mp,'rms'))
```

3.7 Future

As we get to know CARMA and refine calibration strategies, a number of new techniques will undoubtedly will have to be addressed. We name a few that are appearing on the horizon that you can expect future versions of the cookbook to address:

- Polarization
- Blanking and Flagging: baseline and band dependant integration times. This will require some changes to the lower level Miriad code.
- More detailed primary beam models for OVRO and BIMA for improved mosaicing. Currently only simple gaussians are used. Again something needed in Miriad.
- Iterative Selfcal (cf. BIMA Song scripts)
- support for python (see e.g. AIPY)

Appendix A

Setting Up Your Account with Miriad

Setting up your account to use *MIRIAD* of course varies a little from system to system, mostly where the package was installed. If in doubt, ask a local Miriad user. We will assume you are using the `csh` shell. The environment variable `$SHELL` will display what login shell you are using.[1] For `bash`, just replace `.csh` with `.sh` in the examples below.

Typically you will need to know where MIRIAD is stored, and then

```
source /somewhere/miriad/miriad_start.csh
```

If you have installed a binary release, and have not edited the two `miriad_start.*` files, please do so. You may also want to check your version of Miriad:

```
cat $MIR/VERSION
```

it should be version 4.0.5 as of this writing (April 2007), *and 4.0.6 for December 2007.*

A.1 Site dependent setup

Each of the CARMA sites will have a maintained version of Miriad.

A.1.1 OVRO

There are two linux versions are maintained depending if you are down at OVRO in the Owens Valley

```
source /sw/miriad/cvs/miriad_start.csh
```

or up at CARMA on Cedar Flat:

```
source /array/miriad/cvs/miriad_start.csh
```

Local MIRIAD maintainer: Peter Teuben.

[1] For MacOSX an additional surprise will be that the two terminals, Terminal and xterm, have subtle differences how to set your default shell, and for Miriad you should almost always want to use the xterm

A.1.2 Berkeley

Only a linux version is maintained:

```
source /indirect/hp/wright/miriad/mir4/MIRRC.linux
```

Local MIRIAD maintainer: Mel Wright.

A.1.3 Caltech

Maintains Linux, Solaris and MacOSX ?

Local MIRIAD maintainer: ???

```
source ...
```

A.1.4 Illinois

```
source ...
```

Local MIRIAD maintainer: Douglas Friedel.

A.1.5 Maryland

Maryland uses mostly Mandrake Linux (10.1 as of this writing, but switching to Centos 5.1 as we speak) on IA-32 as well as IA-64 type machines. A few Solaris machines are still present, but Miriad is not actively maintained on them (though available upon request).

Maryland also uses `astromake`, which allows you to (interactively) load various packages in your shell. Although this comes with an obvious flexibility, the danger is that loading packages in a certain order could render your interactive shell useless, and loading multiple versions of miriad can make commands from the older version to peek through the new one and cause unexpected results. Use with caution. Example:

```
   % source /n/astromake/astromake_start
   % astroload ds9
   % astroload miriad
or:
   % astroload -v daily miriad
```

Local MIRIAD maintainer: Peter Teuben.

A.2 Installation

Both binary and source based installs are available for Miriad. For a binary release you will need to adjust the path to `MIR` in the two `miriad_start.*` files. There is a risk of shared library conflicts, in which case you will have to relink and/or recompile the code. The Miriad website provided more details and instructions how to do this.

A.2.1 Source Installation

Example of a two liner installation:

```
  1% curl ftp://ftp.astro.umd.edu/progs/carma/miriad.tar.gz | tar zxf -
or:
  1% wget -O - ftp://ftp.astro.umd.edu/progs/carma/miriad.tar.gz | tar zxf -

  2% miriad_cvs/install/install.miriad
```

and a few lines of usage to certify the installation was ok and you probably have a working version:

```
  3% source miriad_cvs/miriad_start.csh

  4% imgen out=map0
  5% itemize in=map0
  6% cgdisp in=map0 device=/xs
```

Note that this version of MIRIAD is a development version, and contain CVS administrative files to allow you to easily update and get the latest fixes directly via CVS. This is much preferred to downloading a tar file each time and install that.

A.2.2 Binary Installation

We expect to make available binary releases for Linux (32bit and 64bit), MacOSX (ppc and intel) and perhaps Solaris (sol10 w/ gcc, sol10 w/ sunstudio12?). Details will be on the Miriad website through the WIKI pages.

A.2.3 Keeping your version up to date

Various files in MIRIAD will be updated from time to time. Even if the source code does not change, there will be updated Flux Catalog and CARMA Baseline data. This is where CVS will come in very handy, so make sure this is installed on your computer. The very first time you want to use `cvs` you may not the "login" and store the anonymous password.

```
  1% cd $MIR
  2% cvs login
  Logging in to :pserver:anonymous@cvs.astro.umd.edu:2401/home/cvsroot
  CVS password:
  3% cvs -nq update
  ...
  M src/inc/maxdim.h
  M src/inc/maxdimc.h
  ...
  U src/subs/fitsio.for
  U src/prog/misc/itemize.for
  ...
```

Lines that start with ``U'' need to be updated:

```
  4% cvs update
```

after which subroutine can be added to the library, and programs can be re-installed:

```
  5% mirboss
  6% mir.subs fitsio
  7% mir.prog itemize
```

New and Old Build System

At the moment MIRIAD is undergoing a transition from an old build system (the `mir.subs` and the `mir.prog` are part of this) to a new "autoconf" based system that uses a `Makefile`. In the new build system **any** update **should** work as follows:

```
1% cd $MIR
2% cvs update
3% make install
```

Appendix B

Miriad cheatsheet

B.1 Reminders

- miriad-101:
  - the Miriad Package is a set of Unix commands, often called "tasks", with a set of *keyword=value* command line parameters to control the program. Typicallly you source a script (e.g. `miriad_start.csh`) to change your Unix environment to have this package included.
  - The Miriad Program (called `miriad`) is a special (miriad) unix program that acts like the AIPS shell and is an alternative method to invoke Miriad programs. Usseful for newbies, as a way of learning individual tasks.
  - Miriad data are directories, with items (normally files, but see below) inside.
- To get help on a task, `mirhelp <taskname>`, e.g. `mirhelp invert`
- source names are stored in UPPER case in visibility files, and are normally converted to upper case before any comparision. Hence the following two examples are synonymous:

```
select=source(mars)
select=source(MARS)
```

- Autocorrelations and a noise source are present in the data, so often you will wind up having to select them out, the minus sign creates an exclusion selection:

```
select=-auto,-source(NOISE)
```

  A notable exception where `select=-auto` does not work is `selfcal` and `mfcal`. This is a bug being worked on.

- When invoking a task from the Unix shell, use quotes for keywords that use unix meta characters, such as parenthesis. Example

```
% uvspec select='win(3)
```

B.2 Miriad DATASETS

Miriad datasets are implemented as a directory[1]. The data itself are organized in *items*, normally implemented as separate files, but small items (32 bytes or less) can be found together in a file called `header`.

[1] formally they can be a hierarchy of directories, but no practical use has been made of this

The Miriad program `itemize` will list the items in a dataset. Other programs that manipulate items are `puthd` (add of modify a simple item), `copyhd` (copy an item from one dataset to another), `delhd` (remove an item), `gethd` (show value of a simple item), `prthd` (show compound contents of a dataset), and `mathd` (perform a mathematical operation on an item).

Miriad currently knows about two types of data: visibility data and image cubes, described in a bit more detail below:

B.2.1 Visibility data

Bla bla, and See Appendix C for more information.

Apart from direct observatory data, you can create visibility data using `uvgen` or import them from other packages using the `fits` program.

Calibration Tables

Calibration programs such as `selfcal` and `mfcal` write gain and bandpass calibration tables inside a visibility dataset. Programs `gplist` and `bplist` will list them on the screen, and `gpplt options=gains` or `gpplt options=bandpass` will plot them. Programs such as `uvcat` and `uvcal` will selectively apply these complex gains as they copy the data.

B.2.2 Image data

Much like FITS images, miriad images... Although `invert` creates images, you can also create images from scratch with `imgen` and `maths`, and convert them from other packages using the `fits` program.

Mosaic Tables

Not unlike visibility data, image data can also contain ancillary tables to aid the organization of the image data. One example is mosaiced data, where a table of the pointing centers of a mosaiced field (`invert options=mosaic,....`) is contained. To get a listing of these centers, use `imlist options=mosaic`.

B.3 Common Miriad Keywords

A number of keywords are often used with the same meaning. You can use the `mirhelp` command on them to get current help, but here are some reminders to the most important ones:

B.3.1 device=

Graphics output is all done via PGPLOT, and the command line parameter `device=` is commonly used to select the device. Examples: `/ps, fig1.ps/vps, /xs, 2/xs, fig2.cps/vcps, plot1.gif/gif`. The `mirhelp device` command will also explain. If you use `device=?` PGPLOT will give you a list of the devices that were installed in your version of PGPLOT. Note that on some older `gfortran` based compilers the GIF device driver could not be compiled yet and will be absent.

B.3.2 select=

The `select=` keyword that many (but not all!) miriad programs use has a very rich set of commands to select from a visibility data stream. Detailed in the Users Guide, we merely provide a short cheat sheet here. The `mirhelp select` command also provides more details (look for select.kdoc)

```
time(t1,t2)                   in UT, accepts yymmmdd.fff or yymmmdd:hh:mm:ss.s format
ant(a1,a2,...)(b1,b2,..)      select baselines from the a's and b's . b's optional
uvrange(uvmin,uvmax)          (in kLambda)
uvnrange(uvmin,uvmax)         (in nanosecs)
vis(n1,n2)                    visibility number n1..n2 (inclusive)
increment(inc)                every inc'th visibility
ra(r1,r2)
dec(d1,d2)
ha(h1,h1)                     hour angle
lst(lst1,lst2)                LST range
elevation(el1,el2)
dra(p1,p2)
ddec(p1,p2)
dazim(p1,p2)
delev(p1,p2)
pointing(p1,p2)               uses MAX(az,el) error
pol(p1,p2,p3,...)             polarization (select from "i,q,u,v,xx,yy,xy,yx,rr,ll,rl,lr")
source(NAME1,NAME2,...)
purpose(LIST[,option])        Select on purpose (BFGPRSO). CARMA guarentees them to be alphabetical.
freq(f1,f2)                   sky freq must be in range f1..f2 (GHz)
amp(amplo,amphi)              one number means amp(amplo)
shadow(d)                     shadowing less than 'd' (meter)
bin(b1,b2)
on(n)                         select on (1) or off (0) for single dish observations
auto                          auto correlations
window(w1,w2,...)             spectral window number (1..maxspect)
seeing(s1,s2)                 select when rms path variations is between s1..s2 (microns)
```

These may be combined (logical AND) with comma separation, e.g. `select=ant(1),win(5)`.

B.3.3 line=

```
line=channel,NUMBER,START,WIDTH,STEP      (integers)
line=velocity,NUMBER,START,WIDTH,STEP     (km/s)
line=wide,NUMBER,START,WIDTH,STEP         (integers)
```

NUMBER = number of channels to output

START = starting channel number

WIDTH = number of channels to average together

STEP = channel increment

The `mirhelp line` command also provides more details (look for line.kdoc)

B.3.4 region=

Much like the `select=` for visibility data, this selects a portion from your miriad image data cube for further processing. Again, details are in the Users Guide,we merely provide this in brief form here. The `mirhelp region` command also provides more details.

```
images(z1,z2)
quarter(z1,z2)
boxes(xmin,ymin,xmax,ymax)(z1,z2)
polygon(x0,y0,x1,y1,x2,y2,...)(z1,z2)
mask(file)
abspixel
relpixel
relcenter
arcsec
kms
```

B.3.5 options=

This is a catch-all keyword many programs use to refine the operations of a program. They are normally used as a comma separated list of (minimum matched) options, e.g.

```
% uvplt vis=3c273 options=nocal,flagged,nobase,dots
```

Many programs share common options.

B.3.6 vis=, in=

Used for input for visibility data (vis=; some programs, such as `invert`, accept multiple files separated by a comma) and images (in=).

Appendix C

UV Variables

C.1 UV Dataset

A *MIRIAD* uv dataset is composed of a collection of items and '$u-v$ variables'. The variables are parameters that are known at the time of the observation, and include measured data, and the description of the observation set up (*e.g.* correlator set up and observing centers).

Table C.1 gives a list of the items that are used to build up a *MIRIAD* uv dataset.

The *Programmers Guide* contains more detailed information on how a visibility dataset is constructed, this Appendix only reports which variables can be found in the item `visdata`. The text item `vartable` contains an ordered (for quick indexing) list of all the variables which exist in the `visdata` item.

A list of all items in a visibility dataset is summarised in Table C.1 below. A list of all the uv variables can be obtained with the *MIRIAD* program `uvlist` or `uvio` for the brave of heart.

The storage **types** (2nd column) in the table below are:

```
A -- ascii (NULL terminated)
R -- real (32 bit IEEE)
D -- double (64 bit IEEE)
C -- complex (2 * 32 bit IEEE)
I -- integer (32 bit twos complement)
J -- short (16 bit twos complement)
K -- long (64 bit  twos complement) *** not currently used in visdata ***
```

They are the same as the data type in the first column of the `vartable` item in a *MIRIAD* uv dataset.

Variables with two dimensions have the first dimension varying fastest, the usual FORTRAN notation.

NB: The formal version of this document is recorded as "*January 3, 2008*".

Table C.1: *MIRIAD* items in a uv visibility dataset

Item name	Type	Description
obstype	ascii	value: '`cross`', '`auto`' or '`mixed`'
history	text	history text file (in principle editable)
vartable	text	lookup table for all uv variables (do not edit!)
visdata	mixed	data stream of uv variables
flags	integer	optional flags for narrowband data
wflags	integer	optional flags for wideband data
gains	mixed	antenna gain table (delhd this item to disable gain table)
nfeeds	integer	number of feeds on each antenna
ntau	integer	Number of delay/spectral index terms per antenna in 'gains'
nsols	integer	number of records in 'gains'
ngains	integer	number of antenna gains in each record of 'gains'
interval	double	gain interpolation time tolerance (days)
leakage	complex	polarization leakage parameters
freq0	double	reference frequency for delay terms
freqs	mixed	frequency set up description table for 'bandpass'
bandpass	complex	bandpass function gains (delhd this item to disable passband corrections)
nspect0	integer	number of windows in the bandpass function
nchan0	integer	total number of channels in the bandpass function

Name	Ty	Units	Comments
airtemp	R	centigr.	Air temperature at observatory
antaz(nants)	D	deg.	azimuth of antennas (BIMA was using 0=south CARMA will use 0=north)
antdiam	R	meters	Antenna diameter
antel(nants)	D	deg.	elevation of antennas
antpos(nants, 3)	D	nanosec	Antenna equatorial coordinates, with X along the local meridian (not Greenwich)
atten(nants)	I	dB	Attenuator setting (Hat Ck/CARMA) datatype R ???
axismax(2,nants)	R	arcsec	Maximum tracking error in a cycle. axismax(1,?) is azimuth error, axismax(2,?) is the elevation error.
axisoff(nants)	R	nanosec	Horizontal offset between azimuth and elevation axes (CARMA)
axisrms(2,nants)	R	arcsec	RMS tracking error. axisrms(1,?) is azimuth error, axisrms(2,?) is the elevation error.
baseline	R	-	The current antenna baseline Baseline is stored as $256 * ant1 + ant2$ or $2048 * ant1 + ant2 + 65536$ The uv coordinates are calculated as $uvw = xyz(ant2) - xyz(ant1)$. Note that this is different from the AIPS/FITS convention (where $uvw = xyz(ant1) - xyz(ant2)$). When writing this variable, software must ensure that $ant1 < ant2$. `baseline` is also known as `preamble(4)` or `preamble(5)` depending if you have uv or uvw data resp.
bin	I	-	Pulsar bin number.
cable(nants)	D	nanosec	measured length of IF cable (Hat Ck)
calcode	A	-	ATCA calcode flag
chi or chi(nants)	R	radians	Position angle of the X feed relative to the sky. This is the sum of the parallactic angle and the `evector` variable. If only one value is present, all antennas are assumed to have identical values.
chi2	R	radians	Second feed angle variation (SMA)

coord(*)	D	nanosec	uv(w) baseline coordinates ?? what epoch ?? `coord` is also known as `preamble(1:2)` or `preamble(1:3)` depending if you have uv or uvw data resp.
corbit	R	-	Number of correlator bits (Hat Ck)
corbw(2)	R	MHz	Correlator bandwidth setting (Hat Ck) Must take the values 1.25, 2.5, 5.0, 10.0, 20.0, 40.0 & 80.0 MHz.
corfin(4)	R	MHz	Correlator LO setting before Doppler tracking (Hat Ck) This is the LO frequency at zero telescope velocity Must be in the range 80 to 550 MHz.
cormode	I	-	Correlator mode (Hat Ck). Values are: 1 : 1 window /sideband by 256 channels 2 : 2 windows/sideband by 128 channels 3 : 4 windows/sideband by 64 channels, single sideband 4 : 4 windows/sideband by 64 channels, double sideband
coropt	I	-	Correlator option (Hat Ck) 0 means cross-correlation 1 means auto-correlation Same as the `obstype` item?
corr(nchan)	J or R	-	Correlation data `corr` is really a complex quantity, but the data stream variable can be stored otherwise for efficiency.
cortaper	R	-	On-line correlation taper (Hat Ck) This is the value at the edge of the window The value is from 0-1.
dazim(nants)	R	radians	Offset in Azimuth. (CARMA)
ddec	R	radians	Offset in declination from `dec` in `epoch` coordinates. The actual observed DEC is calculated as dec + ddec.
dec	R or D	radians	Declination of the phase center/tangent point. See `epoch` for coordinate definition. See also `obsdec`
delay(nants)	D	nanosec	delay setting at beginning of integration (Hat Ck)
delay0(nants)	R	nanosec	delay offset for antennas (Hat Ck)
deldec	R or D	radians	Declination of the delay tracking center. See `epoch` for coordinate definition.
delev(nants)	R	radians	Offset in Elevation (CARMA)
delra	R or D	radians	Right ascension of the delay tracking center. See `epoch` for coordinate definition.
dewpoint	R	centigr.	Dew point at weather station (Hat Ck)
dra	R	radians	Offset in right ascension from `ra` in `epoch` coords. The actual observed RA is calculated as ra + dra/cos(dec).
epoch	R	years	A badly named variable – this defines the mean equinox and equator for the equatorial coordinates `ra`, `dec`, `dra` and `ddec`. The epoch of the coordinates is actually the observing time. Values less than 1984.0 are Besselian with coordinates in the FK4 system. Values greater than 1984.0 are Julian with coordinates in the FK5 system. You will typically find 1950.0 or 2000.0 here.
evector or evector(nants)	R	radians	Position angle of the X feed, to the local vertical. If only one value is present, all antennas are assumed to be identical.
focus(nants)	R	volts	Focus setting (Hat Ck)
freq	D	GHz	Rest frequency of the primary line
freqif	D	GHz	? (Hat Ck only?)
inttime	R	seconds	Integration time (see also time)
ischan(nspect)	I	-	Starting channel of spectral window
ivalued(nants)	I	?	Delay step (Hat Ck) Used in an attempt to calibrate amp and phase vs. delay.
jyperk	R	Jy/K	The efficiency Jy/K,

			calculated during online calibration
jyperka(nants)	R	sqrt(Jy/K)	Antenna based Jy/K,
			calculated during online calibration (Hat Ck)
latitud	D	radians	Geodetic latitude of the observatory.
lo1	D	GHz	First local oscillator (Hat Ck/CARMA)
			`lo1` is in the range 70 GHz - 115 GHz for 3mm.
lo2	D	GHz	Second local oscillator (Hat Ck)
longitu	D	radians	Longitude of the observatory.
lst	D	radians	Local apparent sidereal time.
modedesc	A	-	Correlator mode description (CARMA only)
			Example: 500-32-8-X-X-X-X-X
mount	I	-	The type of antenna mounts.
or mount(nants)			If only one value is given, all antennas
			are assumed to be the same. Possible values are:
			0: Alt-az mount.
			1: Equatorial mount.
			2: X-Y.
			3: orbiting.
			4: bizarre.
name	A	-	ATCA raw RPFITS file name.
nants	I	-	The number of antennas
			Following variables use a dimension of `nants`:
			antpos(nants, 3)
			focus(nants)
			phaselo[1-2](nants)
			phasem1(nants)
			systemp(nants, nspect)
			wsystemp(nants, nwide)
			temp(nants, ntemp)
			tpower(nants, ntpower)
			axisrms(2,nants)
			dazim(nants)
			delev(nants)
			The antennas are always numbered starting at 1.
nbin	I	-	Total number of pulsar bins.
nchan	I	-	The total number of individual frequency channels
			The following variables have the dimension of nchan:
			corr(nchan)
npol	I	-	The number of simultaneous polarisations
nschan(nspect)	I	-	Number of channels in spectral window
nspect	I		Number of spectral windows
			Following variables use a dimension of `nspect`:
			ischan(nspect)
			nschan(nspect)
			restfreq(nspect)
			sdf(nspect)
			sfreq(nspect)
			systemp(nants, nspect)
ntemp	I	-	Number of antenna thermisters
			Following variables use a dimension of `ntemp`:
			temp(nants, ntemp)
ntpower	I	-	Number of total power measurements
			The following variable depends on `ntpower`:
			tpower(nants,ntpower)
			`ntpower` is currently 1, could be more later.
nwide	I	-	Number of wideband channels
			Variables which depend on `nwide` are:
			wfreq(nwide)
			wwidth(nwide)
			wcorr(nwide)

			wsystemp(nants,nwide)
obsdec	D	radians	Apparent declination of the phase centre/tangent point at time of observation. See also `dec`
observer(*)	A	-	The name of the observer
obsline(*)	A	-	The name of the primary spectral line of interest to the observer
obsra	D	radians	Apparent right ascension of the phase centre/tangent point at time of observation. See also `ra`
on	I	-	Either 1, 0 or -1, for on, off pointing, and Tsys spectrum resp. for auto-correlation data.
operator(*)	A	-	The name of the current operator
pbfwhm	R	arcsec	(Deprecated) Primary Beam at Full Width Half Maximum For Hat Ck, it is approximately 11040.0/`lo1`.
pbtype(*)	A	-	Primary beam type to be used in imaging.
phaselo1(nants)	R	radians	Antenna phase offset (Hat Ck/CARMA)
phaselo2(nants)	R	radians	Second LO phase offset (Hat Ck/CARMA)
phasem1(nants)	R	radians	IF cable phase (Hat Ck/CARMA)
plangle	R	degrees	Planet angle
plmaj	R	arcsec	Planet major axis (note units)
plmin	R	arcsec	Planet minor axis
pltb	R	Kelvin	Planet brightness
pntdec	R or D	radians	Declination of the pointing center. See `epoch` for coordinate definition.
pntra	R or D	radians	Right ascension of the pointing center. See `epoch` for coordinate definition.
pol	I	-	Polarization type of the correlation data. Values follow the AIPS/FITS convention, viz: 1: Stokes I 2: Stokes Q 3: Stokes U 4: Stokes V -1: Circular RR -2: Circular LL -3: Circular RL -4: Circular LR -5: Linear XX -6: Linear YY -7: Linear XY -8: Linear YX
precipmm	R	mm	Mm of precipitable water vapour in the atmosphere.
pressmb	R	millibar	atmospheric pressure.
project(*)	A	-	The name of the current project
purpose(*)	A	-	Scientific intent or purpose For CARMA: B=bandpass, F=flux, G=gain (phase/amp) P=polarization, R=radio pointing, S=science target, O=other
ra	R or D	radians	Right ascension of the phase center/tangent point. See `epoch` for the definition of the coordinate system. See also `obsra`
rain	R	mm	The current amount of water in the rain gauge. The rain gauge is emptied at 9:00 AEST (ATCA).
refpnt(2,nants)	R	arcsec	Reference pointing offsets. refpnt(1,?) is azimuth offset, refpnt(2,?) is the elevation offset.
relhumid	R	%	Relative Humidity at observatory
restfreq(nspect)	D	GHz	Rest frequency for each spectral window. This may be zero for continuum observations.
rmspath	R	microns	RMS path variation (CARMA, for HatCrk units were %) see also smonrms
sctype	A	-	Scan type (ATCA?)
sdf(nspect)	D	GHz	Change in frequency per channel
sfreq(nspect)	D	GHz	Sky frequency of (center of) first channel in window
smonrms	R	μm	ATCA seeing monitor rms value (see also rmspath)

source(*)	A	-	The name of the source
srv2k(nants)	R	?	??? (Hat Ck)
systemp	R	Kelvin	Antenna system temperatures
or systemp(nants)			
or systemp(nants,nspect)			
tau230	R	-	Optical depth at 230 GHz, as measured with the ... system (Hat Ck/CARMA)
tcorr	I	-	HasTsys correction has been applied (0:none, 1:applied) (CARMA, ATNF)

telescop(*)	A	-	The telescope name. Some standard values are:
			`'ATCA'`
			`'HATCREEK'`
			`'VLA'`
			`'WSRT'`

temp	R	centigr.	Antenna thermistor temperatures (Hat Ck)
(nants, ntemp)			
themt(nants)	R	Kelvin	temperature of the hemt amplifier (Hat Ck)
tif2(nants)	R	Kelvin	temperature of IF amplifier (Hat Ck)
time	D	days	The time (nominally UT1) stored as a Julian date. For example, noon on Jan 1, 1980 is 2,444,240.0! `time` is also known as `preamble(3)` or `preamble(4)` depending if you have uv or uvw data resp. time is the beginning of an integration with length inttime
tpower	R	volts	Total power measurements (Hat Ck)
(nants, ntpower)			
trans	R	K	CARMA
tscale	R	-	Optional correlation scale factor Used only when `corr` is stored as J (16 bits).
tsis(nants)	R	Kelvin	temperature of the SIS mixers (Hat Ck)
tsky	R	-	CARMA
ut	D	radians	The time since midnight Universal time (nominally UT1).
veldop	R	km s^{-1}	The sum of the radial velocity of the observatory (in the direction of the source, with respect to the rest frame) and the nominal systemic radial velocity of the source.
veltype(*)	A	-	Velocity rest frame. Possible values for `veltype` are: VELO-LSR: rest frame is the LSR VELO-HEL: rest frame is the barycentre VELO-OBS: rest frame is the observatory FELO-LSR: rest frame is the LSR (deprecated) FELO-HEL: rest frame is the barycentre (deprecated)
version(*)	A	-	The current hardware/software version Current options: oldhat, newhat For carma: x.y.z
vsource	R	km s^{-1}	Nominal radial systemic velocity of source. Positive velocity is away from observer.
wcorr(nwide)	C	-	Wideband correlations. The current ordering is: wcorr(1:2) are the digital LSB and USB. wcorr(3:4) are the analog LSB and USB.
wfreq(nwide)	R	GHz	Wideband correlation average frequencies (center?)
wind	R	km/h	Wind speed in km/h (ATCA)
winddir	R	deg	Wind direction (where the wind is blowing from) (note: originally encoded as 'N', 'SE', 'W', etc.)
windmph	R	mph	Wind speed - in imperial units!
wsystemp	R	K	System temperature for wide channels.
or wsystemp(nants)			
or wsystemp(nants,nwide)			
wwidth(nwide)	R	GHz	Wideband correlation bandwidths
xsampler	R	percent	X sampler statistics (ATCA).
(3,nants,nspect)			
xtsys(nants,nspect)	R	Kelvin	System temperature of the X feed (ATCA).
xtsysm(nants,nspect)	R	Kelvin	???
xyamp(nants,nspect)	R	Jy	On-line XY amplitude measurements (ATCA).

xyphase (nants,nspect)	R	radians	On-line XY phase measurements (ATCA).
ysampler (3,nants,nspect)	R	percent	Y sampler statistics (ATCA).
ytsys(nants,nspect)	R	Kelvin	System temperature of the Y feed (ATCA).
ytsysm(nants,nspect)	R	Kelvin	???

C.2 Telescope specific notes

A reminder on some telescope specific variables

C.2.1 ATCA

```
calcode
name
rain
sctype
smonrms
wind
xsampler(3,nants,nspect)
xtsys(nants,nspect)
xyamp(nants,nspect)
xyphase(nants,nspect)
ysampler(3,nants,nspect)
ytsys(nants,nspect)
```

C.2.2 CARMA

```
dazim(nants)
delev(nants)
modedesc
axisrms       "skyErr"  -- temporary sqrt(2) issue
axisoff
lo1 changes, phaselo1=0
lo2 still absent
purpose
```

C.2.3 SMA

```
chi2
```

C.2.4 BIMA/Hat Creek

Although the telescope name is for historic reasons called `HATCREEK`, they are really the 6m `BIMA` antennae, but while this array was operational at the Hat Creek site in Northern California. The following UV variables were specificially used for this array, although some of them moved to CARMA as well:

```
atten(nants)
cable(nants)
corbit
corbw(2)
corfin(4)
cormode
coropt
cortaper
```

```
delay(nants)                    carma
delay0(nants)
dewpoint
focus(nants)
freqif
ivalued(nants)
lo1                             carma
lo2
phaselo1(nants)                 carma
phaselo2(nants)                 carma
phasem1(nants)                  carma
rmspath                         carma
srv2k(nants)
tau230                          carma
temp(nants, ntemp
themt(nants)
tif2(nants)
tpower(nants, ntpower)
tsis(nants)
```

C.3 Examples

```
% ls -l 3c273/
total 1808
-rw-r--r--    1 teuben   teuben      49952 Oct 12  1998 flags
-rw-r--r--    1 teuben   teuben        136 Jul 24  1998 header
-rw-r--r--    1 teuben   teuben      48700 Oct 12  1998 history
-rw-r--r--    1 teuben   teuben        671 Jul 24  1998 vartable
-rw-r--r--    1 teuben   teuben    1725300 Jul 24  1998 visdata
-rw-r--r--    1 teuben   teuben       1760 Jul 30  1998 wflags

% itemize in=3c273
Itemize: Version 31-JUL-97
  nwcorr   = 13608
  ncorr    = 387072
  vislen   = 1725304
  obstype  = crosscorrelation
  history    (text data, 48704 elements)
  visdata    (binary data, 1725300 elements)
  vartable   (text data, 675 elements)
  flags      (integer data, 12487 elements)
  wflags     (integer data, 439 elements)

% uvio 3c273
uvio Version 16-jan-01 rjs
0x       0 FILE: 3c273
0x       0 SIZE: project   Count=12,Type=a
0x       8 DATA: project   n196d028.cal
0x      18 SIZE: source    Count=5,Type=a
0x      20 DATA: source    3C273
0x      30 SIZE: ra        Count=1,Type=d
0x      38 DATA: ra                3.26861624
0x      48 SIZE: dec       Count=1,Type=d
0x      50 DATA: dec              0.03582093395
0x      60 SIZE: vsource   Count=1,Type=r
0x      68 DATA: vsource                      0
0x      70 SIZE: plmaj     Count=1,Type=r
0x      78 DATA: plmaj                        0
0x      80 SIZE: plmin     Count=1,Type=r
.....
0x    12b0 DATA: tif2              34.60030746
0x    12e8 SIZE: pol       Count=1,Type=i
0x    12f0 DATA: pol       -6
0x    12f8 SIZE: wcorr     Count=18,Type=c
0x    1300 DATA: wcorr            0.1456700563        -0.1822117269
0x    1398 SIZE: tscale    Count=1,Type=r
```

```
0x    13a0 DATA: tscale         0.0009044817416
0x    13a8 SIZE: corr      Count=1024,Type=j
0x    13b0 DATA: corr      0
0x    1bb8 SIZE: coord     Count=2,Type=d
0x    1bc0 DATA: coord             -96.06886361
0x    1bd8 SIZE: time      Count=1,Type=d
0x    1be0 DATA: time               2450671.342   97AUG10:20:12:01.1
0x    1bf0 SIZE: baseline  Count=1,Type=r
0x    1bf8 DATA: baseline                   260
0x    1c00 ========== EOR (1) ========
0x    1c08 DATA: wcorr             0.1115201488         0.1867246479
0x    1ca0 DATA: tscale          0.001088747638
0x    1ca8 DATA: corr      0
0x    24b0 DATA: coord              19.81238265
0x    24c8 DATA: baseline                   516
0x    24d0 ========== EOR (2) ========
0x    24d8 DATA: wcorr              0.106826134        -0.0437762402
0x    2570 DATA: tscale         0.0009396394016
0x    2578 DATA: corr      0
0x    2d80 DATA: coord             -8.372860208
0x    2d98 DATA: baseline                   261
0x    2da0 ========== EOR (3) ========
...
0x   15068 DATA: baseline                  2314
0x   15070 ========== EOR (36) ========
0x   15078 DATA: obsra               3.268015211
0x   15088 DATA: obsdec           0.03609215511
0x   15098 DATA: chi              -0.7118714452
0x   150a0 DATA: tpower             19.17340851
0x   150d8 DATA: ut                5.289289984
0x   150e8 DATA: lst               2.459578844
0x   150f8 DATA: axisrms           0.2067008913
0x   15160 DATA: focus             7.899638176
0x   15198 DATA: delay             289.0346413
0x   15200 DATA: antaz             1.040362579
0x   15268 DATA: antel            0.5775128425
0x   152d0 DATA: themt                   475
0x   15308 DATA: tif2              34.53898239
0x   15340 DATA: wcorr            0.02284911089        -0.1492281109
0x   153d8 DATA: tscale         0.0009604850202
0x   153e0 DATA: corr      0
0x   15be8 DATA: coord             -95.98970399
0x   15c00 DATA: time               2450671.342   97AUG10:20:12:13.0
0x   15c10 DATA: baseline                  260
0x   15c18 ========== EOR (37) ========
0x   15c20 DATA: wcorr             0.2651385069        0.05663052946
0x   15cb8 DATA: tscale         0.0009315458592
0x   15cc0 DATA: corr      0
0x   164c8 DATA: coord             19.82889086
0x   164e0 DATA: baseline                  516
0x   164e8 ========== EOR (38) ========
0x   164f0 DATA: wcorr             0.2428172529         0.1108904481
.....
```

Appendix D

CARMA Data

As a reminder, here we summarize some of the peculiarities of CARMA MIRIAD visibility data if you have been used to BIMA, SMA, WSRT or ATNF data.

D.1 Oddities

1. All CARMA data have auto-correlations preceding the cross-correlations. Some calibration programs, most notably `selfcal` and `mfcal`, cannot handle this? File bugzilla ? `select=-auto` to filter them out.
2. Some of the correlator setting has half edge channels and should always be flagged.
3. Most CARMA data have a noise source added, which can be used for bandpass calibration. However, be sure not to apply linelength or baseline corrections to these data. Use `select=-source(noise)` to filter them out.

D.2 Data Versions

Sometimes it is useful to know at what stage your CARMA data has been taken, and at what stage the data was (re)filled by the Data Archive. A special uv variable `version` is used to label this formal data version:[1]

```
% uvlist vis=cx012.SS433.2006sep07.1.miriad options=var,full | grep version
UVLIST: version  4-may-06
version :0.1.2
```

D.3 version

Here is the log of data versions. Those annotated with **[refill]** should be refilled in order to see the corrected data. The various stages of baseline corrections are not maintained here, see Section D.4.6.

- 2006/02/01: (VERSION 0.1.2)
- 2006/12/01: noise source sufficiently amplified for narrow band passband calibration

[1] see also: carma/sdp/AstroHeaderWriter.cc: astroHdrMap_p.putString("version", "1.0.1", 1);

- 2006/12/xx: correlator now handling all windows on all baselines
- 2007/01/xx: auto-correllations added [**refill**]
- 2007/01/11: intent (uv variable `purpose` added), e.g. select=purpos(b)
- 2007/01/xx: fixed cross-talk other subarrays that stored some uv variables as 0 (bugzilla #376?)
- 2007/01/31: jyperk now correctly made baseline dependant for proper `invert` weighting (bugzilla #339) [**refill**]
- 2007/02/08: (VERSION 1.0.1) new convention of storing `skyErr` monitor point in `axisrms` [**refill**] (CVS 1.80)
- 2007/03/23: baselines updated. All data between Jan 8 and March 23 should be patched manually using `uvedit`.
- 2007/03/??: blanking activated
- 2007/05/24: line-length corrections activated
- 2007/11/26: amplitude decorrelation fixed; use uvdecor see D.4.5) to fix
- 2007/12/04: source name confusion (bugzilla #564 fixed (btw, we're at data version 1.0.3!!)
- 2008/01/23: fixed minor antenna pad correction rotation; data before this ALWAYS need baseline corrections applied
- 2008/03/31: frequent semi-automated updates of flux calibrator lists (FluxSource.cat in both CARMA and MIRIAD)
- 2008/04/20: better models for mars brightness temperature (see also miriad's `marstb` program)

D.4 Historic Data Correction

In past times certain data corrections were needed that have since then been moved into the data filler or at the telescope monitor point level. The latter type can normally not be solved by refilling the data.

D.4.1 Axis offset correction

An axis offset correction is normally never needed. Only early engineering data (before January 5, 2007) need this axis offset correction. Example of usage:

```
axcor vis=xxx.mir axoff=@axoff.comb.070101 out=yyy.mir
```

Also note the `axcor` program may not be installed with your version of Miriad.

D.4.2 jyperk (bugzilla 339)

Data before 'xxx' confused the scalar `jyperk` with the deprecated array `jyperka` antenna based array. In order to correct this data, such that programs like `invert` will correctly compute the noise characteristics of the resulting image, use the `jyperk` program:

```
jyperk vis=xxx.mir out=yyy.mir
```

One can optionally supply an array of Jy/K values for the 15 antennae, but the current values in the 65 and 145.3 for OVRO and BIMA antennas resp.

See also bugzilla bug # 339.

D.4.3 Flagging based on tracking errors (bugzilla 376)

The `axisrms` UV variable holds the tracking error (in arcsec, in Az and El) for each antenna in the array. It can be useful to automatically flag data when the tracking is above a certain error, or even antennae based (e.g. allow OVRO to have a smaller tolerance than the BIMA antennae). In older data the axisrms was not written properly, and could even be negative. It is currently written $\sqrt{2}$ times what it really should be. But check your plots!

```
% varplt vis=c0048.umon.1.miriad device=/xs yaxis=axisrms options=overlay yrange=-4,4

% uvflag vis=c0048.umon.1.miriad 'select=-pointing(0,4)' flagval=flag options=noapply
```

this last example shows the number of visibilities that would be flagged if their RMS pointing was off by more than 4 arcsec.

D.4.4 Incorrect source name in miriad file (bugzilla 564)

Should be obvious looking at the output of `listobs`. oct/nov 2007. still looking into this. Fixed Dec 3, 2007. Always been present, but never showed up until the last few months. Load on `acc` seems to have triggered this bug in MAW data avering. If your data is mislabeled, a careful uvcat, puthd the source name could fix it. But for mosaic'd observations the pointing center would probably be mis-averaged, and although the visibility data seem to be ok, the data processing would be affected. The advice is to flag the time range of the mislabeled source, since they are small portions of your track, but if you want to get the most out of the data, are careful uvcat/puthd combination may get you there.

D.4.5 Amplitude Decorrelation

All data taken before November 26, 2007, are subject to a small amount of amplitude decorrelation dependent on the difference in delay length between the two antennas in a baseline. The program `uvdecor` attempts to correct for this:

```
% uvdecor vis=xxx.mir out=yyy.mir delaymax=8550
```

Note that the integration times (now baseline based) are adjusted (decreased) to account for the increased noise on baselines with longer antenna delay differences. The value of `delaymax=8550` (nanoseconds) was emperically determined from good fringetest data in the 2007 B array, in which the amplitudes dropped linearly with delay differences. The `delaymax` value is where the amplitude would have dropped to 0!

Especially if your source is extended, it is highly recommended to play with this option for B-array data (with delays up to 6000 ns, decorrelation up to 70%) but even in C-array data (delays up to 2000 ns, decorrelations up to 25%) it should be considered.

D.4.6 Baseline Correction

All data prior to January 23, 2008, should have their baseline corrected. See also Section . Use `uvedit` and the appropriate baseline file

Index