Miriad

Multichannel Image Reconstruction,

Image Analysis and Display

CARMA Cookbook

Peter Teuben

18 Jun 2012 - 15:08

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 (Linux, Solaris, 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 Guides

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

1.1.1 Miriad Users Guide

Ideally the cookbook assumes familiarity with the 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 a description of the powerful `select=` keyword. Chapter 6 on image data is much shorter but also important to read. Chapter 10 on flagging is also important. Since this particular Users Guide was most recently overhauled by the ATNF team, some of the procedures described in there are somewhat ATNF specific. Hence our CARMA cookbook.

1.1.2 SMA Users Guide

The separately 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 D) 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. There are also some commands in MIRIAD that are CARMA specific, and might not run (well) on data from e.g. WSRT, ATNF or SMA.

1.2 Future

This cookbook is a live document, and you can expect changes to come with a new correlator, procedures such as PACS, and polarization. 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

1.3 STYLE

There are two possible styles for code examples. One is WYSIWYG as you type it on the Unix command line and in shell scripts:

```
 % uvlist vis=cx012.SS433.2006sep07.1.miriad options=spectra
```

and the other employs the `miriad` shell, an AIPS like program in which you set variables and run the task, one by one. The output also shows all program parameters, leaving them blank if the default is used. These examples are not cut-and-paste friendly, but perhaps read a little easier. This `miriad` shell will also remember the previous setting of your keywords, which can be a blessing as well as a curse.

```
Task:   uvlist
  vis      = cx012.SS433.2006sep07.1.miriad
  options  = spectra
  select   =
  line     =
  scale    =
  recnum   =
  log      =
```

Within the `miriad` shell, the parameters are global, when you switch a taskname, the same `vis=` would appear in this list if the new task also has this parameter. This concept doesn't quite apply to the Unix command line.

1.4 Links

- MIRIAD main webpage: http://carma.astro.umd.edu/miriad
- Wiki page for Miriad and CARMA Cookbook: http://carma.astro.umd.edu/wiki/index.php/Miriad

[1] subscription details on web....

- Miriad bugzilla (part of CARMA bugzilla) at http://www.mmarray.org/bugzilla
- Google mailing list for MIRIAD users, at http://groups.google.com/group/miriad?hl=en

1.5 Revision History

- 20-apr-2007: first draft
- 15-feb-2008: Draft for version 2 of this document
- 21-jun-2008: Summerschool 2008 version
- 12-jul-2009: Largely Revised Summerschool 2009 version
- jun-2012: Summerschool 2012 version (bfmask, polarization)

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 written. Not to mention all those convincing me to keep adding more.

Chapter 2

A Simple Example

Before we bury ourselves into the details of a more typical CARMA data reduction, let us look at a very simple continuum observation and go over the essentials from calibrating visibility data to making astronomical images. While this is also a silent introduction to MIRIAD, we will cover more what MIRIAD is all about in later chapters.

2.1 CARMA Continuum data

The dataset ftp://ftp.astro.umd.edu/pub/carma/data/SgrA.mir.tar.gz is used for this example[1]

The commands you see below are all Unix shell commands, most of these are in fact MIRIAD programs. This is one of at least three ways to reduce your data using MIRIAD. It is the most compact form and probably the one most people will be familiar with.

Later we will explain the ways you get your data from the archive, but for now, lets assume you have the tar file in your current (ideally clean) working directory, you first untar this (usually compressed) file using the Unix `tar` command:

```
tar -zxf SgrA.mir.tar.gz
```

and you will see your observation is a directory (a MIRIAD dataset) with files in them. You should never manipulate these file directly, unless you really know what you are doing. MIRIAD program generally deal with these files (in MIRIAD parlor,they are called items).

Probably the next thing you do is make an observing summary log where you can check the names, frequency setup, LST ranges of the various sources in your observation etc.. The MIRIAD task "`listobs`" is used for this:

```
listobs vis=SgrA.mir  log=listobs.log
```

Note that unlike the typical options in Unix commands, MIRIAD commands use a "*keyword=value*" command line syntax, but are otherwise implemented as Unix commands. In fact, they are simple compiled Fortran programs that call libraries. It is relatively straightforward to change and recompile these programs (MIRIAD comes with the source code).

Looking at the `listobs.log` file, and here is the example from SgrA:

[1] windows 1 and 4 from `c0319.1D_86GCRing.5.mir`, observed on March 20, 2009 by M. Pound and F. Yusef-Zadeh, in D array, with a spatial resolution of about 5 arcsec.

```
            Observed Sources Coordinates and Corr Freqs
Source         Purpose    RA          Decl         Vlsr            Corfs in MHz
NOISE              B      20 32 45.53 40 39 36.63     0.00E+00     0.0
MWC349             BF     20 32 45.53 40 39 36.63     0.00E+00     0.0
3C345              B      16 42 58.81 39 48 37.00     0.00E+00     0.0
1733-130           G      17 33 02.71 -13 04 49.54    0.00E+00     0.0
GCRING             S      17 45 40.03 -29 00 27.87    0.00E+00     0.0
```

the dataset has several sources: a flux calibrator (F), a phase (gain) calibrator (G), a passband calibrator (B) and the science source (S) of interest. Sometimes some of the calibrators are actually the same. In particular, the phase calibrator will normally be observed alternatingly with your source every 10-20 mins. The flux and/or passband calibrator are normally stronger and observed for a short time (5-15 mins) before or after your observation. Sometimes a planet, if available, will be observed as well, for a more reliable flux calibration. In this simple example we will just use a passband and phase (gain) calibrator, `3C345` and `1733-130` resp. The NOISE source is an artificial NOISE source, injected by the system which can optionally be used for passband calibration.

For convenience, we first extract all the cross-correlations of all astronomical sources from this dataset, leaving out the NOISE source and the auto-correllations (more about those later). This is done with the powerful `uvcat` command:

```
uvcat vis=SgrA.mir  select='-source(NOISE),-auto' out=all.vis
```

Now we are ready for a series of calibrations that can be applied to all the data. Line length and baseline calibration are the two we show first:

First we apply a line length calibration. This corrects for any instrumental phase drifts that are due to changes in the length of the fibers, which are notably temperature sensitive and will expand and contract during the day, and especially can be abrubt during sunrise and sunset. *Maybe give example with varplt how to look at these line length phases?*

```
linecal vis=all.vis
uvcat   vis=all.vis out=all_1.vis
```

Next we can apply a baseline correction. Often during the course of an array configuration, better baselines become available, and they should be retro-actively applied to data from that array configuration. An option in the general purpose `uvedit` program accomplishes this, by feeding is a new antenna position file:

```
uvedit vis=all_1.vis out=all_2.vis apfile=$MIRCAT/baselines/carma/antpos.090331
```

Here you will need to find out which `antpos` file is the appropriate one for your observation. Check the output of listobs with available `antpos` files. If you want to see how much the positions are different, use this command:

```
$MIRCAT/baselines/carma/xyzdiff.py listobs.log $MIRCAT/baselines/carma/antpos.090331
```

There are now two basic approaches to calibration. We either continue on the scheme described before: selecting a calibration type, and apply this to all the data, and continue this pipeline until we are ready to map. Although very elegant, some of the more complex calibration schemes are not easily implemented this way. Nonetheless, let us see how this scheme continues.

First we use a short 15 minute observation of a bright quasar (3C345 in this case) to calibrate the data such that the passband will then have a zero phase (on average) and constant amplitude. The task `mfcal` does this by computing an antenna based frequency dependent solution, which we then apply to the whole observation.

```
mfcal  vis=all_2.vis  refant=2  interval=999  select="source(3C345)"
uvcat  vis=all_2.vis  out=all_3.vis  options=nocal
```

We now calculate the gains as a function of time using the `selfcal` program[2]. This program can also lookup the last reported flux of the calibrator and thus give you properly calibrated fluxes in your source later on. After selfcal, just as before, we apply the gains and now obtaining a fully calibrated dataset!

```
mselfcal vis=all_3.vis refant=2 select="source(1733-130)" options=amp,apriori,noscale interval=5
puthd in=all_3.vis/interval value=0.1
uvcat vis=all_3.vis out=all_4.vis
```

After calibration, and before applying the gains, you will notice we have cheated and widened the validity interval (in units of days) to something big enough to safely interpolate into the source data. Use the output of listobs and your own judgement if this is a valid assumption.

In this particular example we have calibrated both amplitude and phase, another option would be to only calibrate the phases, and use a planet for amplitude scaling. This will be covered in later chapters.

Now we are ready for mapping and deconvolution. One of the standards tests is to map the calibrator and compare it to the beam:

```
invert vis=all_4.vis map=map0 beam=beam0 imsize=128 line=chan,1,1,2,2 select="source(1733-130)"
clean map=map0 beam=beam0 out=model0
restor map=map0 beam=beam0 model=model0 out=clean0
```

This quasare is a point source,so the program `clean` is appropriate to use. But CARMA is a heterogeous array, with different primary beams for different baselines, where we need to apply a mosaicing option to the mapping program, even if there is only one single pointing in the observation. Except when you want to detect or map a very small field of view, or a point source, this is essential. The current example observation of SgrA has 7 pointings, and because the source is extended, mosaicing is essential here:

```
invert vis=all_4.vis map=map1 beam=beam1 line=wide,1,1,2,2 "select=source(GCRING)" options=mosaic imsize=129
mossdi map=map1 beam=beam1 out=model1
restor map=map1 beam=beam1 model=model1 out=clean1

ds9 &
mirds9 clean1
```

A last note before you might get your fingers too wet: these commands are obviously tedious to type, and prone to typos. Of course in real life we use scripts (e.g. C-shell or python scripts), in such a way that you repeat your data reduction script, finetuning the various calibration schemes you apply.

2.1.1 CARMA test data and shell scripts

For this chapter there are 3 shell scripts available for study that with their comments go into more depth and show plotting example of a simple continuum observation. You will always be able to find these scripts, and their accompanying data, in ftp://ftp.astro.umd.edu/pub/carma/data

[2]mselfcal and selfcal are near-identical implementation of the same program

1. **simple0.csh**: A very short 1 minute observation called “fringetest”, used to show how the various calibration schemes can even dillute the signal of a strong quasar.

2. **simple1a.csh**: a simple calibration pipeline, as we discussed before. The advantage of keeping all sources together in a single file is fewer files (and thus I/O, saving time) is needed and it is easier to check how well the calibration worked on any source in your track.

```
#! /bin/csh -f
#
# extracted from c0319.1D_86GCRing.5.mir, originally observed at CARMA on 20 March 2009.
# PI: Marc Pound and Farhad Yusef-Zadeh.
#
#
# this script works mostly with one visibility file, and uses select= to work on selected data
#
# sources:
# --------
# MWC349     flux (ignored here)
# 3C345      passband
# 1733-130   phase
# GCRING     source


# administration: getting the data

set tar=../SgrA.mir.tar.gz
tar -zxf $tar
# set vis=$tar:t:r:r
set vis=SgrA.mir

# set an antpos file for baseline correction: two are available 090225 and 090331
set antpos=$MIRCAT/baselines/carma/antpos.090331

# source names (we use the PURPOSE code from listobs: S, B, G)
set sname=GCRING
set bname=3C345
set gname=1733-130

# reference ants
set bref=2
set pref=2

# work can start -------------------------------------------------------------------------

# get a listing of what happened
listobs vis=$vis log=listobs.log

# select out only the useful astronomical data
rm -rf all.vis
uvcat vis=$vis  select=’-source(NOISE),-auto’ out=all.vis

# notice ant 1 has some issue
# smauvplt vis=all.vis device=/xs axis=time,phase options=nocal,nopass
uvflag vis=all.vis flagval=flag select="ant(1)"

# baseline calibration
rm -rf all_1.vis
uvedit vis=all.vis    out=all_1.vis apfile=$antpos

# linelength calibration
rm -rf all_2.vis
linecal vis=all_1.vis
#gpplt vis=all_1.vis yaxis=phase nxy=5,3 device=/xs options=wrap
uvcat   vis=all_1.vis out=all_2.vis

# smauvplt vis=all_2.vis device=/xs axis=time,phase options=nocal,nopass

# bandpass calibration
rm -rf all_3.vis
```

```
mfcal vis=all_2.vis refant=$bref select="source($bname)" interval=999
#gpplt vis=all_2.vis yaxis=phase nxy=5,3 device=/xs options=bandpass
#gpplt vis=all_2.vis yaxis=amp   nxy=5,3 device=/xs options=bandpass
uvcat vis=all_2.vis out=all_3.vis options=nocal
# gain calibration, looking quite nice now
rm -rf all_4.vis
mselfcal vis=all_3.vis refant=$pref select="source($gname)" options=amp,apriori,noscale interval=5
#gpplt vis=all_3.vis yaxis=phase nxy=5,3 device=/xs options=gains
#gpplt vis=all_3.vis yaxis=amp   nxy=5,3 device=/xs options=gains
puthd in=all_3.vis/interval value=0.1
uvcat vis=all_3.vis out=all_4.vis
# should check flagged values in all_3 and all_4 and they should be the same
# if the new interval is long enough to catch all inter- and extra-polations
# map the calibrator
rm -rf beam0 map0  model0 clean0
invert vis=all_4.vis map=map0 beam=beam0 imsize=128 line=wide,1,1,2,2 select="source($gname)"
clean map=map0 beam=beam0 out=model0
restor map=map0 beam=beam0 model=model0 out=clean0
# map the source
rm -rf beam1 map1 model1 clean1
invert vis=all_4.vis map=map1 beam=beam1 line=wide,1,1,2,2 "select=source($sname)" options=mosaic,double,systemp imsize=129
mossdi map=map1 beam=beam1 out=model1
restor map=map1 beam=beam1 model=model1 out=clean1
```

3. **simple1b.csh**: breaking it up in different source files. This is the way the more complex calibration routines work. You will wind up with a lot more files this way.

```
#! /bin/csh -f
#
# extracted from c0319.1D_86GCRing.5.mir, originally observed at CARMA on 20 March 2009.
# PI: Marc Pound and Farhad Yusef-Zadeh.
#
#
# this script works by copying over gain tables and applying them, not the simplest
# of ways, but flexible in terms of the more complex calibration schemes that can occur.
#
# sources:
# --------
# MWC349     flux (ignored here)
# 3C345      passband
# 1733-130   phase
# GCRING     source
# administration: getting the data
set tar=../SgrA.mir.tar.gz
tar -zxf $tar
# set vis=$tar:t:r:r
set vis=SgrA.mir
# set an antpos file for baseline correction: two are available 090225 and 090331
set antpos=$MIRCAT/baselines/carma/antpos.090331
# source names (we use the PURPOSE code from listobs: S, B, G)
set sname=GCRING
set bname=3C345
set gname=1733-130
```

```
# reference ants
set bref=2
set pref=2

# work can start -------------------------------------------------------------------------

# get a listing of what happened
listobs vis=$vis log=listobs.log

# select out only the useful astronomical data
rm -rf all.vis
uvcat vis=$vis  select='-source(NOISE),-auto' out=all.vis

# notice ant 1 has some issue
# smauvplt vis=all.vis device=/xs axis=time,phase options=nocal,nopass
uvflag vis=all.vis flagval=flag select="ant(1)"

# baseline calibration
rm -rf all_1.vis
uvedit vis=all.vis    out=all_1.vis apfile=$antpos

# linelength calibration
rm -rf all_2.vis
linecal vis=all_1.vis
#gpplt vis=all_1.vis yaxis=phase nxy=5,3 device=/xs options=wrap
uvcat   vis=all_1.vis out=all_2.vis

# now we split the full source (B,G,S) in pieces
rm -rf b_1.vis g_1.vis s_1.vis
uvcat vis=all_2.vis out=b_1.vis select="source($bname)"
uvcat vis=all_2.vis out=g_1.vis select="source($gname)"
uvcat vis=all_2.vis out=s_1.vis select="source($sname)"

# bandpass calibration
mfcal vis=b_1.vis refant=$bref interval=999
#gpplt vis=b_1.vis yaxis=phase nxy=5,3 device=/xs options=bandpass
#gpplt vis=b_1.vis yaxis=amp   nxy=5,3 device=/xs options=bandpass

# and copy gains into G and S, and apply them
rm -rf g_2.vis
gpcopy vis=b_1.vis out=g_1.vis options=nocal,nopol
uvcat  vis=g_1.vis out=g_2.vis

rm -rf s_2.vis
gpcopy vis=b_1.vis out=s_1.vis options=nocal,nopol
uvcat  vis=s_1.vis out=s_2.vis


# gain calibration, looking quite nice now
mselfcal vis=g_2.vis refant=$pref options=amp,apriori,noscale interval=5

#gpplt vis=g_2.vis yaxis=phase nxy=5,3 device=/xs options=gains
#gpplt vis=g_2.vis yaxis=amp   nxy=5,3 device=/xs options=gains

puthd in=g_2.vis/interval value=0.1

rm -rf s_3.vis
gpcopy vis=g_2.vis out=s_2.vis options=nopass,nopol
uvcat  vis=s_2.vis out=s_3.vis

# should check flagged values in s_2 and s_3 and they should be the same
# if the new interval is long enough to catch all inter- and extra-polations

# map the calibrator

rm -rf beam0 map0  model0 clean0
invert vis=g_2.vis map=map0 beam=beam0 imsize=128 line=wide,1,1,2,2
clean map=map0 beam=beam0 out=model0
restor map=map0 beam=beam0 model=model0 out=clean0

```

```
# map the source

rm -rf beam1 map1 model1 clean1
invert vis=s_3.vis map=map1 beam=beam1 line=wide,1,1,2,2 options=mosaic,double,systemp imsize=129
mossdi map=map1 beam=beam1 out=model1
restor map=map1 beam=beam1 model=model1 out=clean1
```

2.2 A little more on MIRIAD

As you have seen above, MIRIAD programs are just Unix programs that you can run, normally from an interactive shell, or through a shell script. The other thing to re-emphasize is that most MIRIAD datasets are implemented as a set of items (usually files) organized in a directory. You refer to a MIRIAD dataset by its directory name, you can use the tools on your computer to copy, rename, remove, archive etc. them.

If you need help on a MIRIAD command, there are several ways. For example for `uvcat` you would type:

```
mirhelp uvcat
uvcat -k
uvcat -kw
```

Appendix B contains another quick summary of MIRIAD and some of the options you will frequently use. Appendix E lists the peculiarities of the MIRIAD data dialect that the CARMA telescopes write, which can be useful for those familiar with MIRIAD data from other telescopes.

Chapter 3

Description of necessary steps

a) Overview

- data inspection
- flag bad data (here and throughout data reduction)
- baseline corrections
- linecal
- passband calibration
- flux calibration
- gain calibration
- making images

b) Details of individual steps

- show plots!

Chapter 4

Common problems: how to spot them and what to do about it

a) phase jumps

b) poor system temperatures

c) lack of ”fringes”

d) bad correlator band

e) missing data for antenna

f) bad channels

g) bad pointing/tracking

h) passband calibrator is resolved

i) missing flux or passband observations

j) bright point source in the field

...

Chapter 5

Example scripts

a) ”continuum”

b) continuum + narrow spectral line

c) flux calibration

d) mosaic

e) hybrid mode

Chapter 6

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.

6.1 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.

6.1.1 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 before the list of datasets has been

[1] javaws is part of the Java Development Kit (JDK) and if not present, you may have to install it. javaws should be in the installation bin directory

displayed 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 gzip 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
```

this creates a directory `cx012.SS433.2006sep07.1.miriad` which we refer to as a miriad dataset. The MIRIAD program `itemize` lists the items in this dataset. It is also a good sanity check if your dataset appears intact.

```
  % 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 once you have placed your data in the cache, so you need to first logon and trigger the CARMA filler to create (or make visible) the dataset.

6.1.2 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. *NOTE: location is now outdated, needs updated*

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). The `/misc/sdp/archive_sciencedata/` are gzip compressed tar files again, but with the site specific `.mir` convention.

6.2 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`.

There is a package called MIS (can be found in `$MIR/borrow/mis`), that takes this one step further, and builds a whole suite of (pipeline-able) scripts around this concept.

6.3 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.

6.4 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]

6.4.1 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:

[2] See: `http://cedarflat.mmarray.org/observing/quality/`
[3] On occasion a glitch in the data transfer process has resulted in incomplete datasets

```
  % 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
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 6.5.2. 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.

6.4.2 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
```

[4] generic email: carma@ncsa.uiuc.edu

```
-----------------------------------------------

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
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
-----------------------------------------------
```

6.4.3 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 6.5.3 for a discussion on this. Note that `uvlist` only displays the first selected frequency setting. If your source and calibrator have a different setup, add `select=source()` to look at the appropriate setting for each object.

6.4.4 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. Since observers change twice a week, this is a good reason to check on your data quality within a few days.

6.4.5 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. The end-goal of calibration is to create flat phases and/or flat amplitudes for the calibrators as function function of frequency and/or time. Use `uvflag` and friends where needed to edit out discrepant data that could throw off 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 "`taskname -k`" will give you a full description of all the program parameters.

To inspect the amplitudes in a totally 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 (but busy) program for checking your data quality is the closure phase, which we expect to be zero on average:

```
% 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 D):

```
% 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

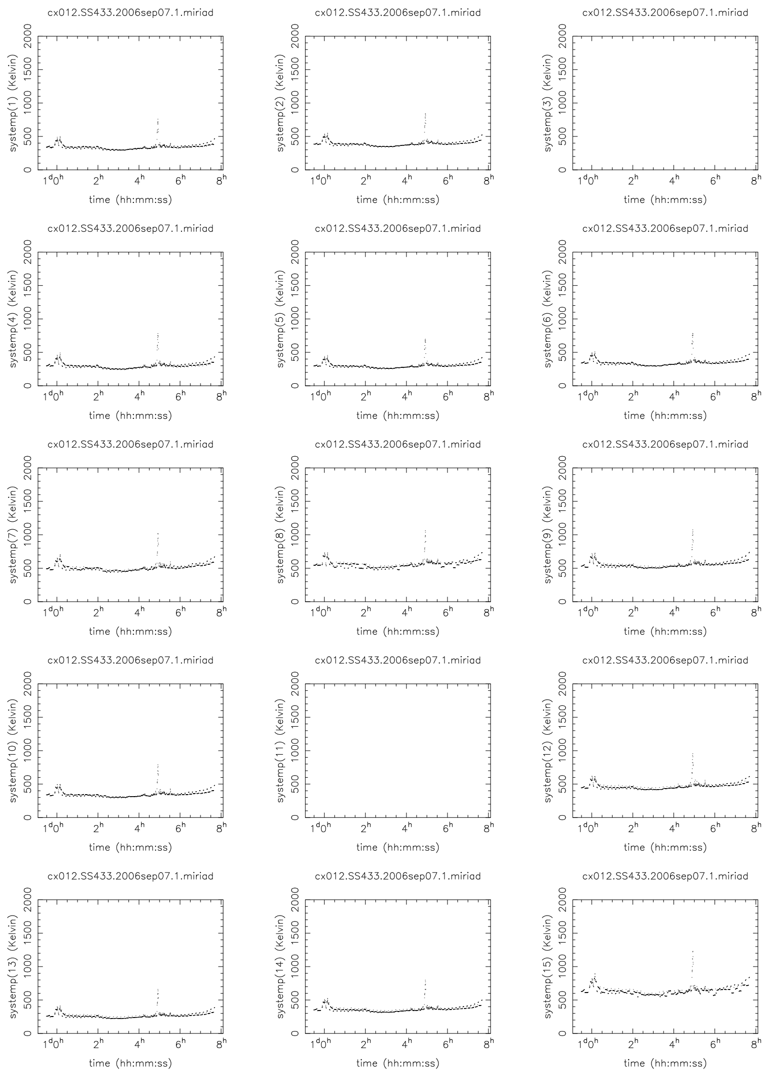

Figure 6.1: System temperature plot for 3C273 made with `varplt`, `cx012.SS433.2006sep07.1.miriad`

6.5 Initial Data Correction

6.5.1 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 thus 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 E 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.

6.5.2 Baseline correction

You should always check if you need to (re)apply a baseline correction[5]. Although your data may come with a pretty decent baseline solution, often a few weeks into a new array configuration improved baselines will become 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`[7] To apply a new baseline, use the program `uvedit` to your multisource data set.

```
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 accidentally apply the same baseline solution twice.

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.

You can also visually check how much the baselines changed from what is available in your visibility dataset, by comparing the X-Y-Z values (in nsec) in the `antpos` file with what `listobs` produced. The following command will compare two antpos files, or even the output of listobs and an antpos file:

```
$MIRCAT/baselines/carma/xyzdiff.py listobs.log $MIRCAT/baselines/carma/antpos.120410
```

6.5.3 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
```

[5] any data prior to 22-jan-2008 should always be corrected
[6] At CARMA, `/home/obs/web_pages/observing/baseline/` is the official repository
[7] use `cvs update` to get any updated baselines

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

```
% 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 losing any potential time variability as well as (in this case 6) dimensionality.

At this stage it is perhaps useful to remind you of the difference between a UV variable, which can be viewed as possibly time dependant variables, and a header item. Both are present in UV datasets. The program `uvputhd` can operate on UV variables. Technically they live in the `DATASET/visdata` item. On the other hand, scalar items can be manipulated with `puthd`. If you create an item with the same name as a UV variable, that UV variable has now been made time-independent and is hidden from view until you remove the item with `delhd`.

6.5.4 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 antenna structure (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 you are looking at.

TODO: careful with select=-source(noise)

TODO: say more about the difference between `linecal` and `uvcal options=linecal`

6.5.5 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 database. 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.

6.5.6 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`).

6.5.7 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 the wide band explicitly.

6.5.8 Flagging using tvflag and pgflag

The interactive flagging program `tvflag` must be run on an 8-bit (a.k.a. PseudoColor) display. Most modern desktops are so color rich, they cannot be effectively run in 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
```

An alternative program, using PGPLOT, and called `pgflag` is available.

6.5.9 Flagging based system temperatures

Although the mapping program `invert` has an option to weigh your data by the system termperatures (`options=systemp`, it may also be advantageous to throw away data with large system temperatures.

For example:

```
% uvflag vis=cx012.SS433.2006sep07.1.miriad tsys=1000  flagval=flag
```

would flag all records where the system temperature was above 1000K.

6.5.10 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.

6.5.11 bfmask based (un)flagging

In the spring 2012 CARMA added a new UV variable, `bfmask`. This is an integer, one for each spectral window, where each bit in this mask identifies any of 31 possible failure modes why the data was flagged

```
% uvlist ...  options=bfmask recnum=0 ...

 Date               Vis # Source     bfmask    A1 A2 Pol  Itime  Dtime     dra(")  ddec(") Flag

12MAY17:00:54:31.5    401 3C111    0x00000000  20 20 LL     9.7   10.0     0.00    0.00 T
12MAY17:00:54:31.5    402 3C111    0x20000F0F  21 21 LL     0.0   10.0     0.00    0.00 F
12MAY17:00:54:31.5    403 3C111    0x00000000  22 22 LL     9.7   10.0     0.00    0.00 T
12MAY17:00:54:31.5    404 3C111    0x000000C0  23 23 LL     9.7   10.0     0.00    0.00 F
12MAY17:00:54:31.5    405 3C111    0x10000000  16 17 LL     9.7   10.0     0.00    0.00 F
```

The task `uvbflag` can be used to unflag records where a certain mask was set, and you would like to take the risk and unflag the data, as the underlying correllations might still be good (this is for you to decide). For example:

```
% uvbflag vis=...   mask=3,4,11,12 flagval=unflag
```

The meanings of the masks is hidden in the help file for uvbflag, as well as described in the item `blfmask` (e.g. `more $VIS/blfmask`).

You could also try looking at all tracking related masked data, and unflag those:

```
% uvflag vis=... flagval=flag
% uvbflag vis=...   mask=3,4,11,12 flagval=unflag
```

would first flag all data (careful: no backup of the old flags is made!), and then unflag the data where originally the CARMA pipeline deemed the antenna was not long enough on target, and flagged the correllation. This would be a way to find out how good the removed correllations were that were not on target and compare them to the "best" deemed values.

6.6 Calibration

6.6.1 Passband Calibration

When a strong calibrator is available it can be used using `mfcal` to correct the passband of your other 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 inspect the solutions with `gpplt`:

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

% gpplt vis=pbcal.mir device=/xs options=bandpass  yaxis=amp    nxy=5,3
% gpplt vis=pbcal.mir device=/xs options=bandpass  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
% uvcat  vis=phasecal.mir out=phasecal_pb.mir
```

and then you can proceed calibrating the now passband corrected data for the phase calibrator. The same applies to any other (e.g. your science source) sources that need to be further calibrated.

6.6.2 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[8] for the calibrator and apply this to the source:

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

```
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
```

6.6.3 Autocorrelation

Auto-correlations are handled by the datafiller as of January 31, 2007 (see also Appendix E). 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
...
```

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

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
```

6.6.4 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, to apply 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
```

6.6.5 Phase Transfer

After the passband solution has been derived and copied to the other sources in your track (e.g. phase calibrator, flux calibrator, science targets), you will want to calibrate the phase (and possibly amplitude) as it changes with time over the course of your observations. This is accomplished by self-calibrating on your gain calibrator, which was observed periodically over the course of your observations, and then using this solution for your science target(s). The necessary assumption is that the atmospheric variations are the same for the science target as for the gain calibrator, which of course depends on the angular distance between the two.

Here is an example of how phase transfer is accomplished:

```
1: set VIS = c0001.example.passband.miriad
2: uvcat vis=$VIS select="source(3C273),win(1),-auto" out=3C273.w1.passband.miriad
3: uvcat vis=$VIS select="source(science),win(1),-auto" out=science.w1.passband.miriad
4: selfcal vis=3C273.w1.passband.miriad refant=8 interval=5 options=phase
5: gpcopy vis=3C273.w1.passband.miriad out=science.w1.passband.miriad
```

In line 1, the passband calibrated data is defined as "c0001.example.passband.miriad".

In line 2, UVCAT is used to select the first window (the lower sideband of band 1) of the gain calibrator (3C273) and write out just this data in a file called "3C273.w1.passband.miriad".

In line 3, UVCAT is used to select the first window (the lower sideband of band 1) of the science target (science) and write out just this data in a file called "science.w1.passband.miriad".

In line 4, SELFCAL is used to derive the phase solution as a function of time using the gain calibrator. The reference antenna is defined as C8 (refant=8), which was chosen because it was in the center of the array and didn't have to be flagged earlier on in the data reduction process. The interval was set to five minutes (interval=5) because that is the amount of time spent on the gain calibrator in each calibrator-science target cycle. Finally, SELFCAL was only used to find the variation of phase with time, and not amplitude with time, by setting options=phase. Amplitude calibration will be discussed in a later section.

In line 5, the phase/time solution derived in line 4 is copied from 3C273 data to the science target.

The above example only calibrates one window of the calibrator and copies that solution to the same window of the science target. Of course, you will want to calibrate every window. For continuum observations, each window has 500 MHz bandwidth, and the easiest way to calibrate the data will be to derive a solution for each window of the gain calibrator and copy the solution to the same window of the science target. For most line observations, usually one or two bands will have narrow bandwidth (62 MHz or less) and the remaining band(s) will have 500 MHz bandwidth. In this case, it is probably best to derive the SELFCAL solution on the gain calibrator for the 500 MHz window and copy that to the other windows in the same sideband.

Here is an example of how to derive the phase/time solution for a narrow band dataset in which band 1 has 500 MHz bandwidth and bands 2 and 3 have 8 MHz bandwidth:

```
set VIS = c0001.example.passband.miriad
uvcat vis=$VIS select="source(3C273),win(1),-auto" out=3C273.w1_500.passband.miriad
uvcat vis=$VIS select="source(3C273),win(2),-auto" out=3C273.w2_8.passband.miriad
uvcat vis=$VIS select="source(3C273),win(3),-auto" out=3C273.w3_8.passband.miriad
uvcat vis=$VIS select="source(3C273),win(4),-auto" out=3C273.w4_500.passband.miriad
uvcat vis=$VIS select="source(3C273),win(5),-auto" out=3C273.w5_8.passband.miriad
uvcat vis=$VIS select="source(3C273),win(6),-auto" out=3C273.w6_8.passband.miriad
uvcat vis=$VIS select="source(science),win(1),-auto" out=science.w1_500.passband.miriad
uvcat vis=$VIS select="source(science),win(2),-auto" out=science.w2_8.passband.miriad
uvcat vis=$VIS select="source(science),win(3),-auto" out=science.w3_8.passband.miriad
```

```
11: uvcat vis=$VIS select="source(science),win(4),-auto" out=science.w4_500.passband.miriad
12: uvcat vis=$VIS select="source(science),win(5),-auto" out=science.w5_8.passband.miriad
13: uvcat vis=$VIS select="source(science),win(6),-auto" out=science.w6_8.passband.miriad
14: selfcal vis=3C273.w1.passband.miriad refant=8 interval=5 options=phase
15: selfcal vis=3C273.w4.passband.miriad refant=8 interval=5 options=phase
16: gpcopy vis=3C273.w1.passband.miriad out=science.w1_500.passband.miriad
17: gpcopy vis=3C273.w1.passband.miriad out=science.w2_8.passband.miriad
18: gpcopy vis=3C273.w1.passband.miriad out=science.w3_8.passband.miriad
19: gpcopy vis=3C273.w4.passband.miriad out=science.w4_500.passband.miriad
20: gpcopy vis=3C273.w4.passband.miriad out=science.w5_8.passband.miriad
21: gpcopy vis=3C273.w4.passband.miriad out=science.w6_8.passband.miriad
22: gpcopy vis=3C273.w1.passband.miriad out=3C273.w2_8.passband.miriad
23: gpcopy vis=3C273.w1.passband.miriad out=3C273.w3_8.passband.miriad
24: gpcopy vis=3C273.w4.passband.miriad out=3C273.w5_8.passband.miriad
25: gpcopy vis=3C273.w4.passband.miriad out=3C273.w6_8.passband.miriad
```

In lines 2-13 we split the passband calibrated data into 12 files, one for each source (the gain calibrator and the science target) and window (windows 1-6). In lines 14+15 we derive the phase/time solution for the gain calibrator (3C273) in the two wide (500 MHz) windows. In lines 16-18 we copy the solution from window 1 of the gain calibrator to windows 1-3 (the lower sideband windows) of the science target. In lines 19-21 we copy the solution from window 4 of the gain calibrator to windows 4-6 (the upper sideband windows) of the science target. Finally, in lines 22-25 we copy the solution from the wide bands of the gain calibrator to the narrow bands of the gain calibrator.

6.6.6 PACS calibration

In PACS ("Paired Antenna Calibration Scheme") mode, a set of CARMA and SZA antennas (currently 8) are paired up (they are physically close to each other) in the A- or B-array. The SZA phases of a quasar at 1cm can be scaled up by the ratio of the observing frequencies to apply a correction to the 1mm or 3mm CARMA phases, to compensate for the rapid atmospheric phase fluctuations. While the SZA is continuesly observing the quasar to obtain these corrected phases, CARMA is switching between the quasar and science source in the usual source-cal cycle.

There are currently two methods to apply PACS calibration, **baseline** based and **antenna** based, described in the next two sections.

Antenna based: gpbuddy

In the antenna based approach, a selfcal solution is first obtained, which determines a time dependant gain (read: phase) for each antenna[9], $g_1(i_1, t_1)$ for the CARMA antennas (i_1), and $g_2(i_2, t_2)$ for the SZA antennas (i_2). The scaled SZA phases can now be applied to the CARMA phases to obtain the phase correction:

$$\phi^C_{i_1 j_1} = \phi^C_{i_1 j_1} - \alpha \phi^S_{i_2 j_2}$$

where α is the ratio of the observing frequencies at CARMA to SZA. This number is around 3 for 3mm and 7 for 1mm.

For various reasons this calibration is currently a two-step process MIRIAD. The program `gpbuddy` computes the antenna based SZA phase correction ($\alpha \phi^S_{i_2}$), and stores these into the `phaseatm` uv-variable of the output dataset. These phase corrections are then applied in a baseline based mode using `options=atmcal` in `uvcal`, including a small bandpass slope correction based on ν/ν_0, where ν_0 is normally LO_1.

The following example illustrates this style of phase calibration, skipping the usual passband calibration, but starting with the slow selfcal, the rapid selfcal, and finally transferring the phases from SZA to CARMA.

[9] this is actually done in two steps, a longer time interval based selfcal is first applied to remove any gross drifts

```
# We have the following sources:
# cal_c :  carma calibrator, roughly only every 20mins for 5 mins
# cal_s :  sza calibrator, continuesly observed
# src   :  carma source, roughly only every 20 mins for 15 mins
#
# to keep the example simple, some keyword (e.g. refant) are omitted

# step 1: slow and fast selfcal for the CARMA calibrator

selfcal vis=cal_c  interval=3
uvcat   vis=cal_c                out=cal_cs
selfcal vis=cal_cs interval=0.05                      #### gains actually unused


# step 2: slow and fast selfcal for the SZA calibrator

selfcal vis=cal_s  interval=3
uvcat   vis=cal_s                out=cal_ss
selfcal vis=cal_ss interval=0.05


# step 3: apply SZA calibrator phases to CARMA calibrator (so you can check improvement)

gpbuddy vis=cal_cs vis2=cal_ss    out=cal_ca
uvcal   vis=cal_ca                out=cal_atm   options=atmcal


# step 4: apply SZA calibrator phases to CARMA source, but first apply slow CARMA calibrator

gpcopy  vis=cal_c                 out=src
uvcat   vis=src                   out=src_s
gpbuddy vis=src_c  vis2=cal_ss    out=src_ca
uvcal   vis=src_ca                out=src_atm   options=atmcal
```

GPBUDDY is a MIRIAD task that copies the gain table of a selected number of antennas into a second set of antennas in another dataset. It is intended to be used in conjunction with UVCAL,to perform antenna-based atmospheric phase correction. All non-selected antennas have the option of having their gains interpolated from paired antennas using different methods.

GPBUDDY will take the gains corresponding to the antennas in list2 from the dataset specified by vis2, then set their amplitudes to one and multiply the phases by a given scale factor, then unwrap them and write them into the antenna-based phaseatm uv variable in the vis dataset for the antennas in list1. Antennas present in vis but not listed in list1 will get a phaseatm value that is obtained from the application of the specified method. Antennas for which the method produce no solution will be flagged during that time interval. `uvcal options=atmcal` will interpret these phaseatm tables at phases at the LO1 frequency and correctly compute and apply the atmospheric delays.

```
Program: GPBUDDY


Keyword: vis
        The input visibility file, containing the visibility data
        to be copied with an additional phaseatm table.
        No default.

Keyword: out
        Output file for vis, if selected. This file will contain the phaseatm
        variable derived from the gains of a buddy antenna.

Keyword: vis2
        The 2nd input visibility file, containing a selfcal gain table from
        which gains will be applied to antennas in the primary
```

```
        dataset (the gain table of the input visibility file).

        Default is to leave this blank, which will simply copy
        gains internally from the primary visibility dataset.

Keyword: show
        Show the East-North layout (in meters) in tabular format.
        LISTOBS will also print these out by default.
        Default: false

Keyword: list1
        The list of primary antennas to receive new gains.

Keyword: list2
        The 2nd list of paired antennas to apply gains to primary

Keyword: scale
        This is usually a number larger than 1 and can normally be
        computed from the ratio of the effective frequencies at which the two
        gain solutions were derived.
        Currently no default allowed, since we have not properly
        obtained these effective frequencies. The usual numbers are 3.09 for
        3mm and 7.4 for 1mm.
Keyword: options
        ** not used at the moment **

Keyword: mode
        gains or phaseatm.
        For gains the gains of the input file(s) are overwritten,
        For phaseatm you will need to supply (an) output file(s).
        DO NOT USE.
        Default: phaseatm

Keyword: nearest
        Use nearest neighbor for time interpolation. If not, linear
        interpolation is used.
        Default: true.

        Will become false, since nearest doesn't know how to flag
        when nothing in the interval.

Keyword: method
        Method by which antenna phased of non-paired antennas are
        deduced.
        Currently implemented are:
        power:     inverse power law weighted average on projected distance
        gaussian:  Gaussian weighted average on projected distance
        tophat:    equal weights within a projected radius
        parabol:   inverse projected distance square within a radius
        none:      none, the phase corrections for non-buddies are 0 (not implemented)
        Default: power

Keyword: param
        Parameter for the weighting function method.
        For power-law: negative of the power index
        For gaussian: Gaussian FWHM (in nanoseconds)
        For tophat: radius (in nanoseconds)
        For parabol: radius (in nanoseconds)
        Default: 2

Keyword: antipol
        Compute antenna phases for non-paired antennas by interpolating
        over paired antennas using a user-selectable weighting function
        specified by wscheme and param
        Default: true

Keyword: reset
        Normally for non-paired antennas the phaseatm are set to 0,
        to prevent any changes to those antennae. However, these baselines
        are not flagged. By setting reset=false you will then force these
```

```
        baselines to be flagged.
        Default: false

        **WARNING**    this keyword will disappear and absorbed into method=
```

Baseline based: bldelay

BLDELAY is a Miriad program that reads baseline phases and derives estimates for the atmospheric delay. The visibility data should have been calibrated to give zero phases in all windows and channels at the start of the data. There is no averaging done. If this is required it should be done in another Miriad task.

```
Program: BLDELAY

Keyword: vis
        The input visibility file. No default.

Keyword: out
        The file containing the derived delays.

Keyword: options
        The options are
            mean        (default) estimate the delay from the mean phase
                        across the band divided by the mean frequency,
                        and zero phase at zero frequency
            slope       estimate the delay from the slope across all the
                        bands or channels
            track       track phases in time to resolve wraps

Keyword: select
        This selects the data to be processed, using the standard uvselect
        format. Default is all data.

Keyword: line
        Specify whether to use the wideband ("wide") or channel
        ("channel") data to derive the delays.

Keyword: wraps
        Number of 2*pi steps to search for the best delay fit. The
        default is 3 (to search +/- 3*pi)
```

minipacs

A method was divised to quickly asses the amount of success PACS can have. This is dubbed *minipacs*, in which two nearby quasars are observed and the difference between regular and PACS calibration can be quickly assessed. It was first used in the winter 2009/2010 PACS campaign.

6.6.7 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. [10]

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

During commissioning CARMA maintained a list of 12 "secondary flux calibrators"[11] and publishes their fluxes as function of time. The current list is 4 (3C84, 3C273, 3C345, and MWC349) for which - based on 1mm and 3mm - also a spectral index is recorded if a reliable planet is available. Fluxes are maintained in MIRIAD in a database that you can consult using the `calflux` program:

```
  % calflux source=3c84
  ...
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 consists of the older BIMA flux calibrator list (1985-2002), appended with new CARMA values. There is a gap between November 2002 and December 2006, which includes the time the BIMA dishes were moved from Hat Creek to Cedar Flat for the CARMA merge.

Another source of information is the flux data maintained by ATCA[12] and SMA[13].

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

`bootflux`... example

6.6.8 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:[14]

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
```

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):

[11] 0319+415 (3C84), 0530+135, 0854+201, 0927+390, 1229+020 (3C273), 1256-057 (3C279), 1642+398, 1751+096, 2148+069, 2225-049 (3C446), 2251+158 (3C454.3), and MWC349

[12] `http://www.narrabri.atnf.csiro.au/calibrators/`

[13] `http://sma1.sma.hawaii.edu/callist/callist.html`

[14] The 2012 version of this program has new options to use two other tables with better models. CARMA Memo #59 discusses this in more detail.

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

This is actually an example where `puthd` should work just as well:

```
% puthd in=ct007.2008mar02_3mm.mars.1.mir/pltb value=187.675 type=real
```

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).

6.6.9 Polarization

CARMA is able to observe in either dual polarization (LL and RR) as well as full stokes (LL, RR, LR and RL).

An example:[15] This assumes FULLPOL correlator mode with 8 (500 MHz) wideband windows, for CARMA this will be in the odd windows 1,3,5,7,9,11,13,15 (the even windows have 0 channels).

step 0: setup parameters

```
set RAW = ...              # name of miriad dataset
set REFANT = 1             # MUST be a 10-m telescope!
set SRC = ...              # source name
set PASSCAL = ...          # passband calibrator name
set GAINCAL = ...          # gain calibrator name
set GAINFLUX = ...         # flux density of gain calibrator, if you know it
set LEAKFILE = "None"      # visibility data file with leak corrections, if available
set MAP = ...              # continuum map name
set I_RMS = ...            # noise in total intensity map
set QU_RMS = ...           # measured noise in Stokes Q,U,V maps
set REGION = 'arcsec,box(12,-12,-12,12)' # region to plot
```

step 1: 'xyphase' calibration

The 'xyphase' is the LCP-RCP phase difference. A direct calibration of the xyphase is possible only on the 10-m telescopes by observing a polarized noise source. This calibration is performed automatically every 45 minutes by the standard observing script. Xyphase calibration data are labeled 'purpose(P)' in the Miriad dataset.

```
xyauto vis=$RAW select='purpose(P)'
    # ... fits purpose(P) data, stores phase correction as a bandpass

smagpplt vis=$RAW options=bandpass,nofit,wrap device=/xs yrange=-180,180 xaxis=chan yaxis=phase
    # ... examine result; all phases will be zero except for C1-C6 LCP

uvcat vis=$RAW out=wide.xy options=nopol select='-source(noise),-auto,-purpose(P)'
    # ... rewrite data to apply the correction
```

step 2: passband correction

At this point phase(LCP) = phase(RCP) on the 10-m, but not the 6-m, telescopes. It is essential to choose a 10-m telescope as the REFANT when fitting the passband in order to transfer the xyphase calibration to the 6-m telescopes.

[15] See also `pol_basic.csh`

```
mfcal vis=wide.xy select=source($PASSCAL) interval=0.1 refant=$REFANT
smagpplt vis=wide.xy options=bandpass,nofit,wrap device=/xs yrange=-180,180 xaxis=chan yaxis=phase
    # ... examine the result

uvcat vis=wide.xy out=wide.pb options=nocal,nopol
    # ... rewrite data to apply the correction

uvaver vis=wide.pb out=wide.av line=chan,8,1,47,47 options=nocal
    # ... condense to just 8 channels to speed further analysis
```

step 3: leakage calibration

Leakage corrections compensate for cross-coupling between the LCP and RCP channels. Leakages depend on the observing frequency and (unfortunately) the observing date (because of receiver swaps). If feasible, use standard leakage tables. For now, obtain these by emailing chat@astro.berkeley.edu. Or, if the gain calibrator was observed over a sufficiently wide range of parallactic angle (>90 degrees, say), fit the leakages to these data, as shown below. Delete flux=$GAINFLUX if flux of gain calibrator is unknown.

```
uvplt vis=wide.av select='source('$GAINCAL'),pol(LL)' axis=time,parang device=/xs options=nobase,nopol
    # ... plot parallactic angle coverage of gain calibrator

if $LEAKFILE == "None" then
  gpcal vis=wide.av options=circular,qusolve,noxy,nopass flux=$GAINFLUX interval=0.5 \
    refant=$REFANT select='source('$GAINCAL')'
      # ... fit leakages (and gains) to gain calibrator in this data set
else
  gpcopy vis=$LEAKFILE out=wide.av options=nocal,nopass
  mfcal vis=wide.av select='source('$GAINCAL')' interval=0.5 refant=$REFANT flux=$GAINFLUX
      # ... copy leakages from $LEAKFILE, fit gains separately
endif

gpaver vis=wide.av options=scalar interval=15
      # smooth gains to 15-minute time resolution
puthd in=wide.av/senmodel value='GSV' type=ascii
# tells Miriad to include gain in variance calculation inside invert
```

step 4: generate maps, measure noise levels

At this point the data are fully calibrated. The xyphase and passband corrections were applied in writing wide.av; the leakage and gains items are present in the dataset;

```
invert vis=wide.av line=chan,8,1,1 \
  map=$MAP.I.mp,$MAP.Q.mp,$MAP.U.mp,$MAP.V.mp beam=$MAP.bm stokes=I,Q,U,V sup=0 \
  'select=source('$SRC')' options=mfs,systemp cell=0.25 imsize=512

rm noiseList
foreach MP ($MAP.I $MAP.Q $MAP.U $MAP.V)
  clean map=$MP.mp beam=$MAP.bm out=$MP.sl niters=3000
  restor map=$MP.mp beam=$MAP.bm model=$MP.sl out=$MP.cm
  cgdisp in=$MP.cm device=/xs region='arcsec,box(20,-20,-20,20)' labtyp=arcsec
  echo " " >> noiseList
  echo $MP".cm" >> noiseList
  imlist options=stat in=$MP.cm region='arcsec,box(20,-20,-20,-5)' | tail -2 >> noiseList
    # ... measure actual noise in a box offset from the center; change region if
    # ... source extends into it
end

tail -20 noiseList
    # ... rms for I probably will be greater than for Q,U,V
```

step 5: plot the polarization

as wel as enter noise levels I_RMS and QU_RMS before proceeding.

```
    cgdisp in=$MAP.I.cm,$MAP.Q.cm,$MAP.U.cm type=pixel,contour,contour options=full \
      region=$REGION labtyp=hms,dms cols1=2 cols2=4 slev=a,$QU_RMS,a,$QU_RMS \
      levs1=-15,-14,-13,-12,-11,-10,-9,-8,-7,-6,-5,-4,-3,3,4,5,6,7,8,9,10,11,12,13,14,15 \
      levs2=-15,-14,-13,-12,-11,-10,-9,-8,-7,-6,-5,-4,-3,3,4,5,6,7,8,9,10,11,12,13,14,15 \
      line=1,3,3 device=/xs
        # ... >3 sigma detection of Stokes Q and/or U needed for a polarization detection
        # ... remember that Q and U can be positive or negative

    rm -r $MAP.poli.cm $MAP.polm.cm $MAP.pa.cm
    impol poli=$MAP.poli.cm polm=$MAP.polm.cm pa=$MAP.pa.cm sigma=$QU_RMS,$I_RMS \
      in=$MAP.Q.cm,$MAP.U.cm,$MAP.I.cm sncut=3,2
        # ... derive pol intensity and PA maps from Stokes parameters

    cgdisp in=$MAP.I.cm,$MAP.poli.cm,$MAP.pa.cm type=contour,amp,angle \
      region=$REGION options=full labtyp=hms,dms vecfac=1.2,4,4 beamtyp=b,l,4 \
      lines=1,1,10 cols1=1 slev=a,$I_RMS \
      levs1=-6,-5,-4,-3,3,4,5,6,8,10,15,20,25,30,35,40,45,50,55 \
      device=/xs
        # ... plot polarization vectors on total intensity map
```

6.7 Mapping and Deconvolution

CARMA is a heterogeneous array, currently with 3 different types of antennae (10.4m, 6.1m and 3.5m), and as such will contribute 6 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) [16].

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!

6.7.1 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
  % mospsf in=xxx.bm ...
  % 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 mostl likely the OVRO-OVRO beam, followed by the OVRO-BIMA beam, and finally the BIMA=BIMA beam.

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

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

6.7.2 Weights in mapping

There are several options in `invert` to weigh your visibilities to optimize resolution vs. sensitivity:

- **sup=** sidelob suprression area (arcsec); 0 means natural, the default is uniform if not supplied.
- **fwhm=** gaussian taper in image domain, given in arcsec. This is effectively a smoothing in the image plane. In AIPS this is normally given in the UV plane, in $k\lambda$:

$$\theta_{xy} = \frac{182}{\theta_{uv}}$$

- **robust=** Brigg's weighting scheme (-big means none, +2 means natural)

In addition some of the options also control the weighting, in particular **options=systemp**. The current version of MIRIAD can only include the system temperatures into the weighting scheme, but for antennas that have bad gains (e.g. pointing errors) it could be useful to fold in the antenna gain.

One scheme to include the gains in the systemp weighting using this command:

```
% puthd in=$phcalfile/senmodel value=GSV type=ascii
```

This variable is normally set by MFCAL or GPCAL[17] but these are not commonly used for gain calibration at CARMA, and in any case `defsmodl` only sets gain-scaled variance for ATCA data. Setting `GSV` causes `uvdat` to include the gains in the variance calculation. `gpcopy` then copies the `senmodel` parameter to the source data.

In order for this to work, one should NOT apply the gains before running invert (though one can and should apply the passband).

It would also be possible to fold in the gains into the systemps, or modify jyperk, since the variance is calculated as follows:

$$\frac{(Jy/K)^2 T_1 T_2}{2\Delta\nu\Delta T}$$

where

$$\frac{Jy}{K} = \frac{2k}{\eta A}$$

for aperture efficiency η and aperture A. There is currently no option in MIRIAD programs to include the gains directly.

6.7.3 Channel averaging in invert

A perhaps not so well known shortcoming of mapping with `invert` is that MIRIAD tasks that depend on noise variances computes that variance based on the first channel that is retrieved. When in early 2010 band 4 was introduced with a much larger number of channels, instead of the COBRA 15 channel 500 MHz windows, this caused the naive approach to produce maps with inferior noise characteristics.

A work-around is to pre-process the data using `uvcal options=avechan` at the time of applying the bandpass calibration. This also has the added advantage that flagged channel data are re-averaged in

[17] see $MIRSUBS/defsmodl.for

new wide-band data. The phase calibration and mapping will then take place using the wide-band data. Mapping can optionally take place via `options=mfs` and properly combine the LSB and USB to account for the different beamshapes at the different frequencies.

```
% mfcal vis=bp_mir
% gpcopy bp_mir -> gc_mir
% uvcal  gc_mir -> gcb_mir options=avechan,nocal
```

6.8 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.

Chapter 7

Recipes

7.1 Calibration

7.1.1 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
...
```

7.1.2 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
```

7.1.3 Calibration-3

Switch correlator setup, with phase transfer.

7.1.4 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
```

7.2 Bandpass calibration

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

```
#! /bin/csh -f
#
# 1. Uses noise source for narrow-band channel to channel bandpass calibration
#    Conjugate LSB for USB
# 2. Uses 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.

# Assumes that a relatively bright quasar has been observed in the following
# modes:
# 1. 500/500/500
# 2. nb / nb/ nb   nb=narrowband
# 3. With 2 bands in narrowband and the other in 500.  aka "hybrid" mode
#     Note - easy mod to script to use 1 band in nb, others in hybrid.
#
# SNV 2/18/2007

# Assumes data properly flagged so that self cal solutions are good!!
# Make sure refant is a good choice!

# To-do list:
# 1. this script assumes just one visibility calibrator

# User parameters

set vis = c0064.jk_m51co_c.4.miriad  # visibility file
set refant=9                         # reference antenna
set cal = 3C279                      # passband calibrator
set viscal = 1153+495                # visibility calibrator
set source = M51MOS                  # source
set nb_array = ( 4 5 6 )             # spectral line bands to calibrate
set wide_array = ( 5 6 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 = "2,3,5,6"         # windows to use for super-wideband
set superwidechan = "1,1,60"         # Channels for superwide
set bw = 64                          # Spectral Line bandwidth
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 = 42                     # visibility calibrator cal interval
set order = 1                        # polynomial order for smamfcal fit
set edge = 3                         # # of edge channels to discard in smamfcal
```

```
set badants = "2,3,5"                    # 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

# End user parameters

uvflag vis=$vis select=anten'('$badants')' flagval=flag

rm -rf all.wide all.nb
rm -rf $cal.wide* $cal.nb* $cal.hyb*

# Select all-wideband and all-narrowband data
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 -r $cal.wide $cal.wide.0 $viscal.v.wide $viscal.v.wide.0
uvcat vis=all.wide out=$cal.wide.0 \
        "select=-auto,source($cal)" options=nocal,nopass
uvcat vis=all.wide out=$viscal.v.wide.0 \
        "select=-auto,source($viscal)" options=nocal,nopass

# mfcal passband on superwideband
# Don't bother using noise source for superwideband
mfcal vis=$cal.wide.0 interval=$calint refant=$refant
echo "**** Plot super-wideband passband on $cal.wide.0 "
gpplt vis=$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
echo "**** Check temporal phase variations on superwideband $cal.wide.0 "
gpplt vis=$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
uvcat vis=$cal.wide.0 out=$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 -r $viscal.v.wide.sw
uvcat vis=$viscal.v.wide out=$viscal.v.wide.sw select='win('$superwidewin')'
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

# LOOP OVER EACH NARROW BAND

set nblength = $#nb_array
if $nblength == 1 set list = 1
if $nblength == 2 set list = ( 1 2 )
if $nblength == 3 set list = ( 1 2 3 )

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
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
endif

echo "**** Be sure that bands are found by inspecting uvlist output!"
echo "**** If no frequency info is found, that bwsel parameters are wrong"
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

# Now select hybrid wideband band
uvcat vis=all.hyb out=$cal.hyb.win$wide.0 "select=-auto,source($cal),win($wide)" \
options=nocal,nopass
# Now 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

# 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
  echo " **** PROCESSING USB"
  rm -r 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 narrowband windows, first do a passband cal using noise source
mfcal vis=noise.nb.win$nb.00 refant=$refant
echo "**** 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 interval=$calint refant=$refant edge=$edge options=opolyfit \
          polyfit=$order
echo "**** Hybrid narrowband passband on $cal.hyb.win$nb.0 "
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 -r test.pass
uvcat vis=$cal.nb.win$nb.0 out=test.pass
mfcal vis=test.pass refant=$refant
echo "**** Narrowband passband (should be flat!)  on $cal.hyb.win$nb.0 "
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
echo "**** Band offset between hybrid-narrowband $cal.hyb.win$nb.a"
echo "****  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 -r $cal.wide.sw
uvcat vis=$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 -r  $source.win$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 -r $source.win_$i
# Apply band offset using smachunkglue naming convention
uvcat vis=$source.win$nb out=$source.win_$i

# end nb loop
end

# glue together 3 bands
set nblength = $#nb_array


if ($nblength == 2) then
  set cfile=$source.$nb_array[1]$nb_array[2]
  rm -r $cfile
  smachunkglue vis=$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 -r $cfile
  uvcat vis=$source.win out=$cfile
endif

# put in restfreq
puthd in=$cfile/restfreq type=double value=$restfreq

# copy super-wideband gains to source
copyhd in=$viscal.v.wide.sw out=$cfile items=gains,ngains,nsols,interval
```

```

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

7.3 Flux Calibration

7.3.1 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 ± 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
------------------------------------------------------------------------
```

7.4 Mosaiced Mapping and Deconvolution

Just make sure you have a good set of CPUs!

7.5 Simple Reduction

7.5.1 Simple Reduction - I

```
#! /bin/csh -f
#
# 3/7/07
#
# Script to reduce CARMA OPHB11 data
#   -- using 2 calibrators:
#           offsetcal to calibrate offset in phase b/w wide and narrow band
#                 cal to calibrate phases
#
# pre-work:
# set vis=c0014.OphB11_C.1.miriad
# carmadata  $vis.tar.gz
# puthd in=$vis/restfreq value=115.271204 type=double
# uvcat vis=$vis out=1751+096W.raw 'select=source(1751+096),-auto,time(11:29:00,11:38:20)'
# uvcat vis=$vis out=1751+096N.raw 'select=source(1751+096),-auto,time(11:08:00,11:30:00)'
# uvcat vis=$vis out=1625-254.raw  'select=source(1625-254),-auto'
# uvcat vis=$vis out=ophb11.raw    'select=source(ophb11),-auto'

umask 002

set task=flag                     # task to perform
set vis==c0014.OphB11_C.1.miriad  # base project vis file
set offsetcal=1751+096            # calibrator for phase offset b/w Wide and Narr. bands
set cal=1625-254                  # primary phase cal name
set source=ophb11                 # source name
set refant=8                      # refant for selfcal
set interval=25                   # interval for selfcal
set scselect=                     # select for selfcal on phase cal
set nxy=3,3                       # number of windows for plotting
set flux=1.6

set ointerval=9999         # interval for offset cal selfcal
set olinecal=chan,1,46,45,45
set linenarrow=chan,1,109,63,63
set axis=time,phase

set line=

set dev=1/xs              # plotting device
set select=                # select for uvplt
set options=              # options for uvplt

set cutoff=

foreach _arg ($*)
    set $_arg
end

goto $task

logs:
listobs vis=


renewoffset:

# Secondary Calibrator split into files with with differen correlator configs:
rm -rf "$offsetcal"N.red
cp -r "$offsetcal"N.raw "$offsetcal"N.red
rm -rf "$offsetcal"W.red
cp -r "$offsetcal"W.raw "$offsetcal"W.red

flagoff:
uvplt vis="$offsetcal"W.red device=$dev  \
```

```
    nxy=$nxy select=$select options=$options axis=$axis

uvplt vis="$offsetcal"N.red device=$dev \
    nxy=$nxy select=$select options=$options axis=$axis

exit 0

selfoff:

# if applying BP cal to 1751, also apply to sec phase, and to source, depending on
#    window ...
#
# or just include USB (calibrating Wide band) :

selfcal vis="$offsetcal"W.red refant=$refant  \
    line=$olinecal options=noscale interval=$ointerval

gpplt vis="$offsetcal"W.red yaxis=phase yrange=-180,180 \
    device=$dev nxy=$nxy options=w

# Apply Wide Band calibration to Narrow Band:
gpcopy vis="$offsetcal"W.red out="$offsetcal"N.red mode=copy options=nopass

rm -rf "$offsetcal"Nref.red
uvcat vis="$offsetcal"N.red out="$offsetcal"Nref.red

# Another selfcal to get offset between Wide and Narrow bands:
selfcal vis="$offsetcal"Nref.red refant=$refant select=$select    \
    line=$linenarrow options=phase interval=$ointerval

gpplt vis="$offsetcal"Nref.red yaxis=phase yrange=-180,180 \
    device=$dev nxy=$nxy options=w

exit 0
renew:
# CAREFUL: This will erase $cal.red to start from scratch
# uvcat  vis=c0014.OphB11_C.1.miriad select=source(1625-254),-auto out=1625-254.raw

rm -rf $cal.red
cp -r $cal.raw $cal.red

flag:
uvplt vis=$cal.red device=$dev line= \
    nxy=$nxy select=$select options=$options

#---Flag data here---
exit 0

selfcal:

#selfcal phase calibrator

selfcal vis="$cal".red flux=$flux refant=$refant select=$scselect options=amp,apriori,noscale interval=$interval line=$olinecal

gpplt:
gpplt vis="$cal".red yaxis=phase yrange=-180,180 \
    device=$dev nxy=$nxy options=w

sleep 2

rm -rf $cal.dm $cal.bm
invert vis="$cal".red map=$cal.dm beam=$cal.bm \
    imsize=237,237 cell=1,1 line=$olinecal options=systemp select=$scselect

rm -rf $cal.cln

clean map=$cal.dm beam=$cal.bm niters=10000 out=$cal.cln cutoff=$cutoff

rm -rf $cal.restor
restor model=$cal.cln map=$cal.dm beam=$cal.bm out=$cal.restor
```

```

cgdisp in=$cal.restor device=$dev

chmod -R 775 $cal.*

exit 0

renewsource:
# CAREFUL: This will erase $source.red to start from scratch
# uvcat  vis=c0014.OphB11_C.1.miriad select=source(OPHB11),-auto out=ophb11.raw
rm -rf $source.red
cp -r $source.raw $source.red

flagsource:
uvplt vis=$source.red device=$dev line=$line \
    nxy=$nxy select=$select options=$options

#---Flag source data here---
exit 0


invert:

# Copy and apply gains table from phase calibrator
gpcopy vis="$cal".red/ out=$source.red mode=copy

rm -rf "$source"GT.red
uvcat vis=$source.red out="$source"GT.red

# Copy and apply gains table from offset calibrator (narrow band only)
gpcopy vis="$offsetcal"Nref.red out="$source"GT.red mode=copy

rm -rf "$source"cal.red
uvcat vis="$source"GT.red out="$source"cal.red select='window(5)'

rm -rf $source.dm $source.bm
invert vis="$source"cal.red map=$source.dm beam=$source.bm imsize=237,237 \
    cell=1,1 options=systemp,double,mosaic line=$line sup=0 select=$select

#rm -rf $source.cln $source.restor
#clean map=$source.dm beam=$source.bm out=$source.cln \
#    niters=10000 cutoff=$cutoff
#minmax in=$source.cln
#restor map=$source.dm beam=$source.bm \
#    model=$source.cln out=$source.restor

chmod -R 775 $source.*

```

7.5.2 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`.

7.5.3 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
```

7.5.4 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 ($4 == <2) then
   uvindex vis=data.mir log=uvindex.log   # scans uvdata file, keywords: vis, interval, refant, log, options
   listobs vis=data.mir log=listobs.log
   uvlist vis=data.mir options=spectra log=uvlist.log
   # What other variables are possible to plot?
   smavarplt vis=data.mir device=systemp.ps/cps nxy=2,3 yaxis=systemp options=compress  # removing compress prints all baselines
   smavarplt vis=data.mir device=/xs nxy=2,3 yaxis=systemp options=compress yrange=0,1000
   uvlist vis=data.mir options=spec
   uvflag vis=data.mir options=noapply flagval=flag #how many flags applied?
   #uvplt axis variables: time, dtime, amplitude, real, imag, phase, uu, vv, uvdistance, uvangle, hangle, dhangle, parang
   smauvplt vis=data.mir device=/xs axis=time,phase select='-source(Arp193),-source(noise)' #line=wide,1,1
   smauvplt vis=data.mir device=/xs axis=time,amp select='-source(Arp193),-source(noise)' line=wide,1,1
   # closure vis=$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($fluxcal_name)" options=noscale,amplitude,apriori flux=$flux
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 $outfile.cube.cm
#invert vis=source_bp.mir map=arp220.mp beam=arp220.bm cell=0.2 imsize=512 line=channel,1,1,90
invert vis=source_bp.mir map=Arp193.cube.mp beam=Arp193.cube.bm cell=0.2 imsize=512 line=channel,90,1,1 sup=0
#ine=velocity,30,-4.617,43.450
clean map=Arp193.cube.mp beam=Arp193.cube.bm out=Arp193.cube.sl niters=10000
restor map=Arp193.cube.mp beam=Arp193.cube.bm model=Arp193.cube.sl out=$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)
```

Chapter 8

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: dual-polarization (LL and RR) are available, as well as full stokes.
- Blanking and Flagging: baseline and band dependant integration times. This will require some changes to the lower level Miriad code. A special uv-variable `bfmask(nspect)` is being prepared, with a task `uvbflag` that will transfer selected mask-bits to the normal miriad visbility flags.
- 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)
- on the fly (vs. continuous) observing

Appendix A

Installing Miriad

A.1 Setting up your account

Setting up your account to use MIRIAD of course varies a little from system to system, mostly in which directory 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 was installed, and source a file to set up your MIRIAD environment:

```
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 4.3.4 as of this writing (June 2012).

A.2 Site dependent setup

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

A.2.1 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/default/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 given its X11 output.

A.2.2 Berkeley

Only a linux version is maintained:

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

Local MIRIAD maintainer: Mel Wright.

A.2.3 Caltech

Maintains Linux, Solaris and MacOSX ?

Local MIRIAD maintainer: ???

```
source ...
```

A.2.4 Illinois

```
source ...
```

Local MIRIAD maintainer: Douglas Friedel.

A.2.5 Maryland

Maryland uses mostly Scienticic Linux (6.2) on IA-32 as well as IA-64 type machines, but Miriad is only actively maintained on 64-bit machines.

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 /astromake/astromake_start
   % astroload ds9
   % astroload miriad
or:
   % astroload -v daily miriad
   % astroload -v school11 miriad
```

Local MIRIAD maintainer: Peter Teuben.

A.3 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.3.1 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.3.2 Binary Installation

We expect to make available binary releases for Linux (32bit and 64bit), MacOSX (intel only) Details will be on the Miriad website through the WIKI pages.

A.3.3 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 note the "login" login prompt and store the anonymous password. Simply hit the RETURN key, the anonymous account has no 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
```

If you feel confident the compile flags work for you, any next updates can be riguriously done via the simple shortcut

```
8% mirupdate
```

Check your logfiles in `$MIR/tmp` though.

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:
  - for users 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 program that acts like the AIPS shell and is an alternative method to invoke Miriad programs. Useful for first time users, 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`. Another quick way to get help is `invert -k` or `invert -kw`.
- 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 CARMA 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)'
  % uvspec select="win(3)"
  ```

  If you are in the MIRIAD shell, these quotes are not needed. Although they seem identical,there is a subtle difference: the single quote will prevent even shell variables from being interpreted, so in scripts you often use double quotes , e.g.

  ```
  % uvspect select="win($win)"
  ```

B.2 Miriad DATASETS

Miriad datasets are implemented as a directory[1]. The data itself are organized in *items*, normally implemented as separate files inside this directory, but small items (32 bytes or less) can be found together in a file called `header`. 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:[2] visibility data and image cubes, described in a bit more detail below:

B.2.1 Visibility data

See Appendix D 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, though both of those have limitations to the number of UV variables it can produce (and thus limit the functionality of some MIRIAD programs).

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, in essence they have calibrated 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:

[1] formally they can be a hierarchy of directories, but no practical use has been made of this
[2] a third type of data, tables, are now also available but not widely used

B.3.1 vis=

B.3.2 in=

B.3.3 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.3.4 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 [t1,t2)
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.3.5 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.3.6 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
```

Note that unlike UV selection, the `region=` selections do not allow a minus sign to negate the region selection. One way around this is to use the `immask` task to shape the image mask, wihch all programs should properly adhere to.

B.3.7 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.3.8 vis=, in=

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

B.4 Gridding time/frequency/Keeping track of time in MIRIAD

Observations such as the one in CARMA are tagging by time, and map portions of the sky at a number of frequencies. As such the concept of understanding gridding coordinates becomes important how "pixels" are labeled. Are they using the center, or are they using the start of a interval/pixel/voxel... This section will clarify the conventions in MIRIAD. In FITS, an astronomy standard to exchange data, the convention is always the center of the pixel/interval, as long as the FITS keyswords `NAXISn, CRVALn, CDELTn, CRPIXn, CTYPEn` are involved.

B.4.1 time: uvgen, CARMA, uvaver

Time (UV variable `time`, in JD) is recorded at the start of an integration interval (UV variable `inttime`, in sec). For example, for `uvgen time=09jul01:10:00:00 harange=0,1,0.1` you will see times:[3]

```
09JUL01:10:00:00.0
09JUL01:10:05:59.0
09JUL01:10:11:58.0
09JUL01:10:17:57.1
09JUL01:10:23:56.1
09JUL01:10:29:55.1
09JUL01:10:35:54.1
09JUL01:10:41:53.1
09JUL01:10:47:52.1
09JUL01:10:53:51.2
```

Programs such `selfcal` and `mselfcal` extract and average times from these observations and again record the nominal starting time of their interval in the gain tables. `uvaver` can also average in time.

NOTE: CARMA data record their time at the midpoint of the integration. With variable integration times,and the blanking enabled, this makes the definiton of "time" a little fuzzy on the timescale of "inttime".

B.4.2 frequency: uvaver

Programs dealing with frequencies and velocities in multi-channel data follow the normal FITS convention where the center of the interval/pixel

B.4.3 position: invert

Maps produced by `invert` produce sky coordinates that comply with the FITS convention that the center of the pixel refers to the coordinate.

B.5 Programming in MIRIAD

MIRIAD was written with the intent that "anybody" should be able to re-program it. Most MIRIAD programs are implemented as a Fortran PROGRAM, calling a suite of subroutines from our MIRIAD library. Graphics is all done via the publicly available PGPLOT library, which comes with MIRIAD. A small part of the library is actually written in C, which implements all the lower level routines that read and write MIRIAD files. However, most of the "astronomy" is implemented in Fortran.

[3]see also explanation in the uvgen::time= keyword on timecentering

The most common scenario is when a program needs a small modification, for example, a write statement needs more digits, or an extra column.

B.5.1 Old-style build

The old-style build uses a set of shell scripts, whereas a configuration file contains the compiler flags and such. You can recognize the old-style build because `$MIR/bin` and `$MIR/lib` exist and has a few subdirectories. The new-style build works the other way around and places those normally inside a single `$MIR/build` directory.

```
% edit $MIRPROG/calib/mfcal.for
% mirboss
% mir.prog mfcal
```

If you want to change a subroutine, it will depend if your Unix version uses shared libraries if you need to recompile the relevant programs as well. Currently the answer is : Linux uses a shared library (`$MIRLIB/libmir.so`), whereas Mac does not (yet).

```
% edit $MIRSUBS/uvdat.for
% mirboss
% mir.subs uvdat
```

Another common scenario is that code has been updated by the MIRIAD developers. In that case the procedure would be the conservative approach and recompile everything:

```
% cd $MIR
% cvs update
% mirboss
% mir.install subs prog
```

Check the recent logfiles in `$MIR/tmp` for compilation or linking errors.

B.5.2 New-Style build

The new style build uses the well known `configure` script (via the `autoconf` procedure) a `Makefile`. You can recognize it because there will be a single `$MIR/build` directory within which we keep the usual `bin, lib, etc...` directories.

Although this method aims to be more portable (the old method often required editing compiler options in files that were hard to figure out for a newcomer), the drawback compared to the old-style is that you need to be in the MIRIAD home directory, and the chain of dependancies could mean a long (several minutes) build for what takes 0.5 sec in the old-style build. Example:

```
% edit $MIRPROG/calib/mfcal.for
% cd $MIR
% make install
% cd -
```

Unlike the previous examples in the old-style build, this recipe should work for any style of change to the MIRIAD source code .

Appendix C

Scripting

C.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.

C.2 Programmable shells

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

Useful to import, like MIRIAD programs, to allow a *keyword=value* commandline syntax. Easy to fake in csh.

C.3 Example: mosaic.py

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

```
#!/usr/bin/env python
#
#  History:
#  june 02 mchw. ALMA script.
#  15aug02 mchw. CARMA version edited from ALMA script.
#  23aug02 mchw. calculate region from source size.
#  20sep02 mchw. Re-import CARMA improvements for ALMA.
#  25sep02 mchw. Re-import improvements from hex7.csh to hex19.csh
#  26sep02 mchw. Increase imsize from 129 to 257.
#  12mar03 mchw. convert to PYTHON.
#  13mar03 pjt   more conversion to PYTHON, now at 200ft, renamed to mosaic.py

import sys, os, time, string, math
from Miriad import *

version='2003-03-14'

print "   ---  ALMA Mosaicing (Cas A model)   ---   "

# command line arguments that can be changed...
keyval = {
    "config"  : "config1",           # antenna config file (without the .ant extension)
    "dec"     : "-30",               # declination (can be a real number)
    "image"   : "casc.vla",          # image to test (nice Cas-A VLA image as default)
```

```
    "cell"    : "0.04",              # scale size (should be calculated from image)
    "nchan"   : "1",                 # number of channels
    "method"  : "mosmem",            # mosmem, joint, or default
    "flux"    : "732.063",           # expected flux in the image (for mosmem)
    "nring"   : "3",                 # number of rings in the mosaic
    "grid"    : "12.0",              # gridsize (in arcsec) for the mosaic
    "center"  : "",                  # optional center file that overrides (nring,grid)
    "VERSION" : "1.0 mchw"           # VERSION id for the user interface
    }

help = """
The minimum amount of information you need to run this task is:
   a miriad image (image=) for the model.
   an antenna configuration file (<config>.ant) for uvgen
"""

keyini(keyval,help,0)
#                                report current defaults, exit if --help given
setlogger(’mosaic.log’)
#
# ----------------------------------------------------------------------------

#
# define all variables, now in their proper type, for this script
#

config  = keya(’config’)
dec     = keyr(’dec’)
cell    = keyr(’cell’)
nchan   = keyi(’nchan’)
method  = keya(’method’)
center  = keya(’center’)
flux    = keyr(’flux’)
image   = keya(’image’)
nring   = keyi(’nring’)
grid    = keyr(’grid’)

harange = ’-1,1,0.013’
select  = ’-shadow\(12\)’
freq    = 230.0
imsize  = 257                    # avoid 2**N, image size 2**N + 1 is good.  [or calculate from image]

mir = os.environ[’MIR’]

# ----------------------------------------------------------------------------

#   returns a list of strings that are the ascii centers as uvgen wants them
#   (in a file) via the center= keyword
def hex(nring,grid):
    center=""
    npoint=0
    for row in range(-nring+1,nring,1):
        y = 0.866025403 * grid * row
        lo = 2-2*nring+abs(row)
        hi = 2*nring-abs(row)-1
        for k in range(lo,hi,2):
            x = 0.5*grid*k
            npoint = npoint + 1
            if center=="":
                center = center + "%.2f,%.2f" % (x,y)
            else:
                center = center + ",%.2f,%.2f" % (x,y)
    return (npoint,center)

#   get the (as a string) value of an item in a dataset
def itemize(data,item):
    log = ’tmp.log’
    cmd = [
        ’itemize’,
        ’in=%s/%s’ % (data, item),
```

```
        'log=%s' % log
        ]
    miriad(cmd)
    f = open(log,"r")
    v = string.split(f.readline())
    f.close()
    return v[2]

#   copy a file from source (s) to destination (d)
def copy_data(s,d):
    zap(d)
    os.system("cp -r %s %s" % (s,d))


#  should this be
#    units=None
#    if units is None:
#        bla
#    else:
#        bla

def puthd(map,item,value,units=0):
    cmd = [
        'puthd',
        'in=%s/%s' % (map,item),
        ]
    if (units == 0):
        cmd.append("value=%s" % value)
    else:
        cmd.append("value=%s,%s" % (value,units))
    return cmd

def demos(map,vis,out):
    cmd = [
        'demos',
        'map=%s' % map,
        'vis=%s' % vis,
        'out=%s' % out,
        ]
    zap_all(out+"*")
    return cmd;

def invert(vis,map,beam,imsize,select):
    cmd = [
     'invert',
     'vis=%s' % vis,
     'map=%s' % map,
     'beam=%s' % beam,
     'imsize=%d' % imsize,
     'select=%s' % select,
     'sup=0',
     'options=mosaic,double',
     ]
    zap(map)
    zap(beam)
    return cmd

def cgdisp(map):
    cmd = [
        'cgdisp',
        'in=%s' % map,
        'device=/xs',
        'labtyp=arcsec',
        'range=0,0,lin,8'
        ]
    return cmd;

def uvmodel(vis,model,out):
    cmd = [
        'uvmodel',
```

```
        'vis=%s' % vis,
        'model=%s' % model,
        'out=%s' % out,
        'options=add,selradec'
        ]
    zap(out)
    return cmd;

def implot(map,region='quarter'):
    cmd = [
        'implot',
        'in=' + map,
        'device=/xs',
        'units=s',
        'conflag=l',
        'conargs=1.4',
        'region=%s' % region
        ]
    return cmd

def imlist(map):
    cmd = [
        'imlist',
        'in=' + map,
        'options=mosaic'
        ]
    return cmd


def imgen(map,out,pbfwhm):
    cmd = [
        'imgen',
        'in=%s' % map,
        'out=%s' % out,
        'object=gaussian',
        'factor=0',
        'spar=1,0,0,%g,%g' % (pbfwhm,pbfwhm)
        ]
    zap(out)
    return cmd


def mosmem(map,beam,out,region,flux=0,default=0):
    cmd = [
        'mosmem',
        'map=%s' % map,
        'beam=%s' % beam,
        'out=%s' % out,
        'region=%s' % region,
        'rmsfac=200,1',
#        'niters=200'
        'niters=2'
        ]
    if flux != 0:
        cmd.append('flux=%g' % flux)
    if default != 0:
        cmd.appenx('default=%s' % default)
    zap(out)
    return cmd

def restor(map,beam,model,out):
    cmd = [
        'restor',
        'model=%s' %  model,
        'map=%s' % map,
        'beam=%s' % beam,
        'out=%s' % out
        ]
    zap(out)
    return cmd
```

```

def regrid(map,tin,out):
    cmd = [
        'regrid',
        'in=%s' % map,
        'tin=%s' % tin,
        'out=%s' % out,
        'axes=1,2'
        ]
    zap(out)
    return cmd

def convol(map,out,b1,b2,pa):
    cmd = [
        'convol',
        'map=%s' % map,
        'out=%s' % out,
        'fwhm=%g,%g' % (b1,b2),
        'pa=%g' % pa
        ]
    zap(out)
    return cmd

def imframe(map,out):
    cmd = [
        'imframe',
        'in=%s' % map,
        'out=%s' % out,
        'frame=-1024,1024,-1024,1024'           # TODO:: the 1024 here depends on the input image size
        ]
    zap(out)
    return cmd

def uvgen(ant,dec,harange,freq,nchan,out,center):
    cmd = [
        'uvgen',
        'ant=%s' % ant,
        'baseunit=-3.33564',
        'radec=23:23:25.803,%g' % dec,
        'lat=-23.02',
        'harange=%s' % harange,
        'source=$MIRCAT/point.source',
        'telescop=alma',
        'systemp=40',
        #        'pnoise=30',
        'jyperk=40',
        'freq=%g' % freq,
        'corr=%d,1,0,8000' % nchan,
        'out=%s' % out,
        'center=%s' % center                      # notice we don't use a file, but a string of numbers
        ]
    zap(out)
    return cmd

def imdiff(in1,in2,resid):
    cmd = [
        'imdiff',
        'in1=%s' % in1,
        'in2=%s' % in2,
        'resid=%s' % resid,
        'options=nox,noy,noex'
        ]
    zap(resid)
    return cmd

def histo(map,region):
    cmd = [
        'histo',
        'in=%s' % map,
        'region=%s' % region
```

```
        ]
    return cmd

# ============================================================================
#
#   start of the actual script
# -------------------------------------------------------------------------
# Nyquist sample rate for each pointing.
# calc '6/(pi*250)*12'
cells  = 500*cell
region = "arcsec,box\(%.2f,-%.2f,-%.2f,%.2f\)" % (cells,cells,cells,cells)

ant    = config + '.ant'                   # antenna file for uvgen
uv     = config + '.uv'                    # dataset for visibilities

demos1 = "%s.cas.%g.demos"  % (config,cell)
base1  = "%s.%g.cas.%g"     % (config,dec,cell)
base2  = "single.%g.cas.%g" % (dec,cell)

map1   = base1 + ".mp"
beam1  = base1 + ".bm"
map2   = base2 + ".map"
beam2  = base2 + ".beam"
mem    = base1 + ".mem"
cm     = base1 + ".cm"
mp     = base1 + '.mp'
res    = base1 + '.resid'
conv   = base1 + '.conv'

if center == "":
    (npoint,center) = hex(nring,grid)
    print "MOSAIC FIELD, using hexagonal field with nring=%d and grid=%g (%d pointings) " % (nring,grid,npoint)
else:
    centerfile = center
    f = open(centerfile,"r")
    center=f.read()
    f.close()
    npoint = len(string.split(center,","))-1
    center=string.replace(center,'\n',',')
    print "MOSAIC FIELD, using center file %s (%d pointings) " % (centerfile,npoint)

print "   ---  ALMA Mosaicing (Cas A model)   ---   "

print " config  = %s" % config
print " dec     = %g" % dec
print " scale   = %g" % cell
print " harange = %s  hours" % harange
print " select  = %s" % select
print " freq    = %g" % freq
print " nchan   = %d" % nchan
print " imsize  = %d" % imsize
print " region  = %s" % region
print " method  = %s" % method
print " "
print "   ---  TIMING   ---   "

if method == "mosmem":
    print "Generate mosaic grid"
    #  lambda/2*antdiam (arcsec)
    print 300/freq/2/12e3*2e5

    print "Generate uv-data. Tsys=40K, bandwidth=8 GHz "
    miriad(uvgen(ant,dec,harange,freq,nchan,uv,center))
    os.system('uvindex vis=%s' % uv)

    print "Scale model size from pixel 0.4 to %g arcsec" % cell
    # with 0.4 arcsec pixel size Cas A is about 320 arcsec diameter; image size 1024 == 409.6 arcsec
    # scale model size. eg. cell=0.1 arcsec -> 80 arcsec cell=.01 -> 8 arcsec diameter

    copy_data(image,base2)
```

```
    miriad(puthd(base2,'crval2',dec,units='dms'))
    miriad(puthd(base2,'crval3',freq))
    miriad(puthd(base2,'cdelt1',-cell,units='arcsec'))
    miriad(puthd(base2,'cdelt2',cell,units='arcsec'))

    print "Make model images for each pointing center"
    miriad(demos(base2,uv,demos1))

    print "Make model uv-data using VLA image of Cas A as a model (the model has the VLA primary beam)"
    for i in range(1,npoint+1):
        miriad(cgdisp(demos1+"%d"%i))
        vis_i = base1+".uv%d"%i
        demos_i = demos1+"%d"%i
        miriad(uvmodel(uv,demos_i,vis_i))
        if i==1:
            vis_all=vis_i
        else:
            vis_all=vis_all + ',' + vis_i
        print "UVMODEL: add the model to the noisy sampled uv-data"

    miriad(invert(vis_all, base1+".mp", base1+".bm", imsize, select))

    print "INVERT: "

    miriad(implot(base1+'.mp',region=region))
    miriad(imlist(base1+'.mp'))

    print "Make single dish image and beam"

    pbfwhm = string.atof(grepcmd("pbplot telescop=alma freq=%g" % freq, "FWHM", 2)) * 60.0

    print "Single dish FWHM = %g arcsec at %g GHz" % (pbfwhm,freq)

    miriad(imframe(base2,base2+".bigger"))
    miriad(convol(base2+".bigger",base2+".bigger.map",pbfwhm,pbfwhm,0.0))
    miriad(regrid(base2+".bigger.map",base1+'.mp',base2+".map"))
    miriad(imgen(base2+".map",base2+".beam",pbfwhm))
    miriad(implot(base2+".map"))
    miriad(puthd(base2+".map",'rms','7.32'))            #  is that 1/100 of the flux ???

if method=='mosmem':
    print " MOSMEM Interferometer only"
    print " MOSMEM Interferometer only with niters=200 flux=%g rmsfac=200." % flux
    miriad(mosmem(map1,beam1,mem,region,flux=flux))
elif method=='joint':
    print "Joint deconvolution of interferometer and single dish data"
    print "Joint deconvolution of interferometer and single dish data ; niters=200 rmsfac=200,1"
    miriad(mosmem(map1+','+map2,beam1+','+beam2,mem,region))
elif method=='default':
    print "MOSMEM with default single dish image"
    print "MOSMEM with default single dish image; niters=200 rmsfac=200"
    miriad(mosmem(map1,beam1,mem,region))
else:
    print "Unknown method " + method

miriad(restor(map1,beam1,mem,cm))
miriad(implot(map1,region=region))

print "convolve the model by the beam and subtract from the deconvolved image"
b1 = string.atof(grepcmd('prthd in=%s' % cm, 'Beam', 2))
b2 = string.atof(grepcmd('prthd in=%s' % cm, 'Beam', 4))
b3 = string.atof(grepcmd('prthd in=%s' % cm, 'Position', 2))

miriad(convol(base2,base1+'.conv',b1,b2,b3))
miriad(implot(base1+'.conv',region=region))

print "regrid the convolved model to the deconvolved image template"

miriad(regrid(base1+".conv",base1+".cm",base1+'.regrid'))
```

```
miriad(implot(base1+'.regrid',region=region))

#  skipping cgdisp /gif production

miriad(imdiff(base1+'.cm',base1+'.regrid',base1+'.resid'))
miriad(implot(base1+'.resid',region=region))
miriad(histo(base1+'.resid',region=region))

# ===============================================================================

print "print out results - summarize rms and beam sidelobe levels"
print "   ---  RESULTS   ---   "

#   extract information, the hard way

#   BUG: doesn't look like 'mp' has rms???
#rms    = string.atof(itemize(mp,'rms')) * 1000
rms = -1
srms   = string.atof(grepcmd('histo in=%s' % res, 'Rms', 3))
smax   = string.atof(grepcmd('histo in=%s' % res, 'Maximum', 2))
smin   = string.atof(grepcmd('histo in=%s' % res, 'Minimum', 2))
Model_Flux = string.atof(grepcmd('histo in=%s region=%s' % (conv,region),'Flux',5))
Model_Peak = string.atof(grepcmd('histo in=%s region=%s' % (conv,region),'Maximum',2))
Flux       = string.atof(grepcmd('histo in=%s region=%s' % (cm,region),'Flux',5))
Peak       = string.atof(grepcmd('histo in=%s region=%s' % (cm,region),'Maximum',2))
Fidelity   = Peak/srms

print " Config  DEC  HA[hrs]    Beam[arcsec] scale Model_Flux,Peak  Image_Flux,Peak Residual:Rms,Max,Min[Jy] Fidelity"
print " %s %g %s %.3f %g %g %g %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f" % (config,dec,harange,rms,b1,b2,
                                                              cell,Model_Flux,Model_Peak,Flux,Peak,srms,smax,smin,Fidelity)

#mv timing hex19.$config.$dec.$harange.$nchan.$imsize
#cat $config.$dec.$harange.$nchan.$imsize
#cat casa.results
#enscript -r casa.results

#print "DEBUGGING"
#string.atof(itemize(mp,'rms'))
```

Appendix D

UV Variables

D.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 D.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 D.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. UV-Variables, just like items in a MIRIAD dataset, should be limited to 8 characters.

NB: The formal version of this document is recorded as "*June 15, 2012*".

Table D.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 (never edit!)
blfmask	text	lookup table for bfmask
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** (or **rm**) this item to disable gain table
		table: (t,gains(ngains))(nsols)
bandpass	complex	bandpass function gains; **delhd** (or **rm**) this item to disable passband corrections
		table: (nchan0,ngains)
freqs	mixed	frequency set up description table for 'bandpass'
		table: (nschan,ndum,sfreq,sdf)(nspect0)
nspect0	integer	number of windows in the bandpass function
nchan0	integer	total number of channels in the bandpass function
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)
freq0	double	reference frequency for delay terms
leakage	complex	polarization leakage parameters
cgains	complex	baseline based channel gains
ncgains	integer	
ncbase	integer	
wgains	complex	baseline based wideband gains
nwgains	integer	
nwbase	integer	

Name	Ty	Units	Comments
airtemp	R	centigr.	Air temperature at observatory
antaz(nants)	D	deg.	azimuth of antennas (CW; BIMA used 0=south CARMA uses 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.
bfmask(nspect)	I	-	Blanking/Flagging mask - see task UVBFLAG for meaning
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(nspect)	I	-	Number of correlator bits (CARMA: 2,3,4)
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/CARMA ¡2011)
delayry(nants)	D	nanosec	delay setting RY at beginning of integration (CARMA 2011)
delaylx(nants)	D	nanosec	delay setting LX at beginning of integration (CARMA 2011)
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?)
ifchain(nspect)	?	?	ATNF specific
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 or mount(nants)	I	-	The type of antenna mounts. 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 for linelength measurements (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 Hat Ck 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 or systemp(nants) or systemp(nants,nspect)	R	Kelvin	Antenna system temperatures
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 (nants, ntemp)	R	centigr.	Antenna thermistor temperatures (Hat Ck)
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 midpoint of an integration with length inttime NOTE ut is at the starting time of an integration

tpower (nants, ntpower)	R	volts	Total power measurements (Hat Ck)
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 or wsystemp(nants) or wsystemp(nants,nwide)	R	K	System temperature for wide channels.
wwidth(nwide)	R	GHz	Wideband correlation bandwidths
xsampler (3,nants,nspect)	R	percent	X sampler statistics (ATCA).
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	???

D.2 Telescope specific notes

A reminder on some telescope specific variables

D.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)
```

D.2.2 CARMA

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

D.2.3 SZA

```
ambpsys(nants,nspect)         ambient Psys
pamatten(nants)               PAM attenuation
psys(nants,nspect)            Psys
psysattn(nants,nspect)        Psys attenuation
```

D.2.4 SMA

```
chi2
```

D.2.5 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)
```

D.3 Examples

Quite technical, but if you ever need to go into the guts of a MIRIAD dataset, these references on our standard CARMA fringe dataset may be helpful:

```
% ls -l fringe.3C273.2008jun18.4.miriad
total 568
-rw-r--r-- 1 teuben teuben  12688 2008-06-19 11:27 flags
-rw-r--r-- 1 teuben teuben    132 2008-06-19 11:27 header
-rw-r--r-- 1 teuben teuben      0 2008-06-19 11:27 history
-rw-r--r-- 1 teuben teuben    646 2008-06-19 11:27 vartable
-rw-r--r-- 1 teuben teuben 545228 2008-06-19 11:27 visdata
-rw-r--r-- 1 teuben teuben    852 2008-06-19 11:27 wflags

% itemize in=fringe.3C273.2008jun18.4.miriad

itemize: Version 1.4, 2008/02/19 20:06:29 UTC

  obstype  = mixed-auto-cross
  nwcorr   = 6552
  ncorr    = 98280
  vislen   = 545232
  visdata    (binary data, 545228 elements)
  history    (unknown data, 0 elements)
  wflags     (integer data, 212 elements)
  vartable   (text data, 650 elements)
  flags      (integer data, 3171 elements)

% uvio fringe.3C273.2008jun18.4.miriad

uvio Version 14-jan-09 pjt
0x00000000 FILE: (null)
0x00000000 SIZE: airtemp   Count=1,Type=r
0x00000008 DATA: airtemp            25.66670036
0x00000010 SIZE: antaz     Count=15,Type=d
0x00000018 DATA: antaz              125.0117534
0x00000098 SIZE: antel     Count=15,Type=d
0x000000a0 DATA: antel              39.66323054
0x00000120 SIZE: antpos    Count=45,Type=d
0x00000128 DATA: antpos             17.69928734
0x00000298 SIZE: axisoff   Count=15,Type=r
0x000002a0 DATA: axisoff         0.002540719928
0x000002e0 SIZE: axisrms   Count=30,Type=r
0x000002e8 DATA: axisrms            1.447669983
0x00000368 SIZE: cable     Count=15,Type=d
0x00000370 DATA: cable              1555.279097
...
0x000010b8 SIZE: time      Count=1,Type=d
0x000010c0 DATA: time               2454636.497   08JUN18:23:55:16.0
0x000010d0 SIZE: baseline  Count=1,Type=r
0x000010d8 DATA: baseline                   257
0x000010e0 ========== EOR (1) ========
0x000010e8 DATA: wcorr              1.041832447                     0
0x00001120 DATA: tscale         4.079189966e-05
0x00001128 DATA: corr      13059
0x00001298 DATA: baseline                   514
0x000012a0 ========== EOR (2) ========
```

```
0x000012a8 DATA: wcorr             1.045531988                    0
...
0x00084fe0 DATA: baseline                 3343
0x00084fe8 ========== EOR (1091) ========
0x00084ff0 DATA: wcorr             12.95456219          12.70519161
0x00085028 DATA: tscale         0.0007510706782
0x00085030 DATA: corr      9063
0x000851a0 DATA: coord             46.79749408
0x000851c0 DATA: baseline                 3599
0x000851c8 ========== EOR (1092) ========
```

(7114 lines)

Appendix E

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.

E.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 ? Use the `select=-auto` keyword 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.

E.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
```

E.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 6.5.2.

- 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 E.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)
- 2008/11/xx: PACS (SZA) data now routinely available. Usually place a 'z' in front of your ct or cx file but these are not integrated into the Illinois archive.
- 2008/12/22: planet data properly tagged in the first 10-20 sec after a switch from NOISE source (bugzilla #695).
- 2008/12/31: all data on this day have a leapsecond error: `uvedit time=-1` to fix.
- 2008/02/01: PACS (SZA) data now available via NCSA archive
- 2009/03/xx: tau230 was changed from CARMA's Weather.tau225 to OpacityMonitor.tau225

E.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.

E.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.

E.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.

E.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.

E.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.

E.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.

E.4.6 Baseline Correction

All data prior to January 23, 2008, should have their baseline corrected. See also Section 6.5.2. Use `uvedit` and the appropriate baseline file

Appendix F

Formularium

In this Appendix we summarize various formulae and list constants used in MIRIAD and interferometry.

F.1 Interferometry

F.1.1 Antennae positions

The `uvgen/ant=/baseunit=` keywords control the antennae positions of an array. These can be specified in either an equatorial system or a local (topocentric) system. The conversion between the two is controlled as followed. A miriad dataset records the antennae positions in the `antpos` UV variable, which records them in nanoseconds in a local equatorial coordinate system with the X axis pointing outwards from the local meridian. The conversion of these to a local topocentric system is:

$$E, N, U = \cos lat, \sin lat X, Y, Z$$

and the relationship betwen UVW and XYZ is, in a right-handed coordinate system (XY in the equatorial plane, X meridian, Y to the east, Z towards the north pole)

$$u = sinHcosH0Xv = -sindcosHsindsinHcosdYw = cosdcosH - cosdsinHsindZ$$

where H is the hour angle, δ the declination.

F.2 Constants

In `$MIRINC/mirconst.h` various constants are listed that the code uses. For some a double as well as single precision value is present.

```
PI     = 3.14159265358979323846   pi
CMKS   = 299792458.0              Speed of light (meters/second)
KMKS   = 1.380658E-23             Boltzmann constant (Joules/Kelvin)
HMKS   = 6.6260755E-34            Planck constant (Joules-second)
HOVERK = 0.04799216               Planck constant divided by Boltzmann constant (Kelvin/GHz)
```

F.3 Calibration

F.3.1 Line Length Calibration

Line length calibration works by utilizing the LO1 frequency, ν_0, and sending a signal along the fiber to each antannea, and accurately measuring the round trip time. Half this time is stored in the `delay` UV variable (in ns) in a MIRIAD dataset. At the LO1 reference frequency this would amount to a slowly drifting phase correction, computed to be between $-\pi$ and π, and stored in the `phasem1` UV variable (in radians):

$$\phi_i = 2\pi(\nu_0 L_i \bmod 1) - \pi$$

The `linecal` program then stores these antenna base phases from `phasem1` as time dependant complex gains in a standard MIRIAD gaintable

$$g_i = e^{i\phi_i}$$

and can be applied via commands such as `uvcat`. Note that this procedure does not apply any bandpass slope.

Recall that the LO1 frequency, ν_0, is generally doppler tracked, so some of the variations you will see with programs such as `varplt` will be due to a doppler shift component due to earth rotation. The maximum peak-to-peak variation in the earth rotation is 0.46 km/s, or 0.15 MHz at 100 GHz. For a typical 100m (fiberlength difference) baseline this would be about 15^0.

An alternative is `uvcal options=linecal,avechan`, which takes into account that the measurements in `phasem1` were taken at LO1 and takes out an additional frequency dependant phase slope based on $\nu/\nu_0(\phi_i - \phi_j)$. Accross a typical 500 MHz band at 3mm this would amount to about 2^0.

F.3.2 Passband Calibration

`mfcal` solves for antennae based complex gains $b_i(\nu)$, and corrects the visibities as follows:

$$V_{ij}^{'} = b_i b_j^* V_{ij}$$

F.3.3 Gain Calibration

`selfcal` solves for antennae based complex gains, $g_i(t)$, and corrects the visibities as follows:

$$V_{ij}^{'} = g_i g_j^* V_{ij}$$

Note that a gain larger than 1 means a loss of signal (e.g. due to bad pointing) where the observed visibilities needed to be multiplied by larger numbers to get back to the expected flux.

F.3.4 Buddy Phase Calibration

There are two implementatons of phase calibrating using a nearby buddy (SZA) antenna operating at around 30 GHz (1cm).

F.3.5 Interpolation

The time interpolation formulae[1] that computes a baseline based gain at time t, from antenna based gains g_A and g_B that had been computed at two surrounding times, $t_1 < t < t_2$, is given by the following

[1] See also `$MIRSUBS/uvgn.for::uvGnFac()`

expression:

$$G = g_{A,2}(1 + (\left|\frac{g_{A,1}}{g_{A,2}}\right| - 1)\epsilon)\left(\frac{g_{A,1}}{g_{A,2}}\right)^{\epsilon} g^{*}_{B,2}(1 + (\left|\frac{g_{B,1}}{g_{B,2}}\right| - 1)\epsilon)\left(\frac{g_{B,1}}{g_{B,2}}\right)^{-\epsilon}$$

and for a phase-only selfcal solution, where $|g| = 1$, this simplifies to:

$$G = g_{A,2}g^{*}_{B,2}\left(\frac{g_{A,1}}{g_{A,2}}\right)^{\epsilon}\left(\frac{g_{B,1}}{g_{B,2}}\right)^{-\epsilon}$$

where ϵ in [0,1]

$$\epsilon = \frac{t_2 - t}{t_2 - t_1}$$

F.4 Doppler Shift

CARMA spectral line data are normally doppler tracked. In addition channels are linear in frequency space, so you will find that we employ the (non-relativistic) radio definition of the doppler shift:

$$V_r = \frac{(\nu_o - \nu)}{\nu_o}c = \frac{(\lambda - \lambda_0)}{\lambda}c$$

instead of the optical definition:

$$V_o = \frac{(\lambda - \lambda_0)}{\lambda_0}c = \frac{(\nu_o - \nu)}{\nu}c$$

then, VELO is used to label an axis in the radio convention, and FELO when the axis is regular in frequency, but expressed in the optical convention.

In MIRIAD a few UV variables are used to describe this:

- restfreq
- sfreq
- sdf
- vsource Nominal radial ; a constant normally (but not for planets)
- veldop Sum of radial vel of observatory and that of source

Thus for a doppler tracked frequency, ν, the corrected frequency would be

$$\nu' = \nu$$

The relativistic convention for radial velocities is not implemented at CARMA yet.

Index