CARMA Memorandum Series #59

Monitoring of Secondary Calibrator Fluxes at CARMA

Amber Bauermeister, Statia Cook, Chat Hull

University of California, Berkeley

Woojin Kwon

University of Illinois at Urbana-Champaign

Tom Plagge

University of Chicago

June 14, 2012

ABSTRACT

A record of historical secondary calibrator fluxes is maintained at CARMA through two types of monitoring: through dedicated flux calibration (fluxcal) tracks and by extracting calibrator fluxes from science tracks. In this memo, we describe how each of these is carried out, emphasizing the changes implemented in flux monitoring using science tracks in May 2012. We briefly review the causes of variation in the measurements of secondary calibrator fluxes: both the intrinsic variability in the flux of calibration sources (the motivation for flux monitoring) as well as system-induced variations. We also present an overview of how the historical fluxes may be accessed by a CARMA user and how this record of fluxes should be used in the calibration of a science dataset.

Change Record

<table>
<tr><th rowspan="2">Revision</th><th>Date</th><th>Author</th><th>Sections/Pages Affected</th></tr>
<tr><th colspan="3">Remarks</th></tr>
<tr><td rowspan="2">1.0</td><td>2012-Jun-14</td><td>A. Bauermeister</td><td>1-14</td></tr>
<tr><td colspan="3">Initial version.</td></tr>
</table>

Contents

1. Introduction

1.1. Overview

Proper calibration of a dataset involves setting the flux scale using observations of a calibrator with a known flux. Typically, CARMA observations use three calibrators: a primary calibrator (ie., a planet), a bright bandpass calibrator and a less bright, but close-to-the-source, gain or phase calibrator to monitor the antenna gains over the course of the track. With perfect data, one can determine the flux of the bandpass and phase calibrators using the primary calibrator observation (using a program like **bootflux**). However, data are not always perfect and it is therefore always advised to check the derived calibrator fluxes against historical values. If the primary calibrator observation is of poor quality or even missing, it may be necessary to use a calibrator flux taken from historical values.

There is an ongoing effort at CARMA to monitor the fluxes of secondary calibrators over time in order to maintain this historical record of fluxes. This record is used for the calibration of datasets as described above as well as in planning observations, allowing CARMA users to make informed decisions on which calibrators to observe. Calibrator fluxes are tracked in the three bands available (1cm, 3mm and 1mm) through two separate avenues: dedicated observations (Dedicated Fluxcal Tracks, §3) and extraction of fluxes from science tracks (Fluxcal on Science Tracks, §4). In both cases, the fluxes of the secondary calibrators are calibrated using a primary calibrator: typically Mars, Neptune, or Uranus. It is important to note that the planet models used are only known to $\sim 20\%$ and therefore the fluxes we derive are similarly uncertain at this level. Prior to May 2012, secondary calibrator fluxes from science tracks were assigned an error of 15% to reflect this model uncertainty. From May 2012 forward however, fluxes from science tracks are reported with measured uncertainties when possible (see Section 4.2.1, step 11), which reflect the data quality but do not account for planet model uncertainties. The planet models and associated errors are discussed in detail in a separate memo on primary calibrators, Kwon et al., in preparation.

The purpose of this memo is to clarify how the fluxes of these secondary calibrators are calculated and to describe various issues regarding flux variability that may be relevant to a user of CARMA data. A brief description of how one might use the flux monitoring data is given below.

1.2. How to Make Use of Flux Monitoring Data

All the secondary calibrator fluxes extracted from both the dedicated fluxcal tracks and science tracks are stored in the MIRIAD catalog file `FluxSource.cat`. This can be found in each installation of MIRIAD, but will only be as up-to-date as the MIRIAD install. The current version of FluxSource.cat for CARMA can be found on this internal site:
`http://cedarflat.mmarray.org/information/carmaFluxSource.cat`
The user may read the calibrator fluxes directly from the catalog file or use one of following inter-

active tools:

- the new website interface written by Ted Yu:
  http://carma.astro.umd.edu/cgi-bin/calfind.cgi
- the MIRIAD program **calflux**. Running `cvs update` in $MIRCAT will update to the latest version of the flux catalog.
- `xplore`, the CARMA calibrator locator tool by Ted Yu, which can be found at this url:
  http://cedarflat.mmarray.org/observing/tools/xplore.html
  Use File→Update to get the latest version of the flux catalog.

If you have a dataset with observations of a primary flux calibrator, a standard strategy for flux calibration is to first flag and bandpass calibrate the data as you would normally then perform a phase-only **selfcal**. Running **bootflux** on the phase-only **selfcal**'d file will automatically calculate the true flux of the secondary calibrator over the course of the track (an averaging interval of 5 to 10 minutes is appropriate) using the planet brightness temperature written in the MIRIAD dataset by the CARMA system (see Kwon et al. for details). If desired, the brightness temperature may be specified in the **bootflux** call by `primary=PlanetName,Tb`. It is wise at this point to compare the fluxes you derive to the historical fluxes recorded in `FluxSource.cat`, being sure to compare values taken at a similar date and frequency. Fluxes derived from science tracks from May 2012 onward also report the spectral index of the calibrator when it can be derived (see Section 4.2.1). This can be used to extrapolate a flux to a different frequency in the band if desired, but should be used with caution. It is up to the discretion of the user to determine if the derived flux is appropriate or if some average from historical data should be used.

When using **bootflux**, it is important to keep the following points in mind. First, calibrator polarization, elevation-dependent gains and other issues can cause variation in the fluxes over the course of the track. See §2 for details. Second, the planet brightness temperature in the MIRIAD dataset is the planet model brightness temperature evaluated at the frequency of the LO. Therefore, if you are determining the flux in a window many GHz away from the LO, you may want to use a brightness temperature appropriate for the window via the `primary=PlanetName,Tb` keyword of **bootflux**.

The flux scale in your data can be set during the amplitude and phase gain calibration using the phase calibrator. For this discussion, we use an amplitude and phase **selfcal** for this step. If `options=apriori` is specified, the **selfcal** program will automatically read the `FluxSource.cat` file in your MIRIAD repository and select a flux appropriate for the date and frequency of your observation. This functionality should be used with caution: while some interpolation in time is performed, the frequency is mostly disregarded (the program only notes which of the three bands (1cm, 3mm, 1mm) the data is relevant for). Alternatively, the user can force a certain flux value in one of two ways: use the flux keyword of **selfcal** (this only works if `options=apriori` is NOT specified) or make a copy of `FluxSource.cat` in the working directory which contains the desired flux. The **selfcal** program will default to reading the local copy. A different flux catalog file can also be pointed to with the $MIRFLUXTAB variable.

Summary:

- Flag and bandpass calibrate (ie. **mfcal**) normally
- Phase-only **selfcal** with a short interval on the gain, passband and flux calibrators. Use `options=apriori` to treat the planet appropriately (as a disk).
- Run **bootflux**: ie. for the flux of 2232+117 from Uranus in the first wideband window
  `bootflux vis=A1.phsc.uv select=source(2232+117,URANUS),-auto taver=10 line=chan,1,1,39,39 primary=URANUS`
- Compare the flux calculated by **bootflux** to historical fluxes from `FluxSource.cat` and decide on an appropriate flux for your calibrator. If desired, write the selected flux for your frequency and date into a local copy of `FluxSource.cat`.
- Perform an amplitude **selfcal**, setting the desired flux either in a local copy of `FluxSource.cat` or with the flux keyword of **selfcal** (and without `options=apriori`).

A full example script for the flux calibration of a typical dataset is provided in Appendix A.

2. Observed Flux Variation: Intrinsic and System-Induced

2.1. Intrinsic Variability

Monitoring the fluxes of secondary calibrators is important as these objects are in no way steady. For an extreme example of intrinsic variation, see Figure 1, which shows the flux of 3C454.3 in each CARMA band as a function of time for the past four years (fluxes taken from `FluxSource.cat`).

2.2. Polarized Calibrators: 3 mm Perceived Variability

The 3mm feeds at CARMA are linearly polarized. The state of the system in 2011/2012 is such that the evector is defined at 90° so that the polarization of the data appears to be YY. Note that currently (Feb 2012), the data are incorrectly labelled as LL. For stokes I intensity I, linear polarization fraction p_{qu}, and polarization angle PA, the observed YY intensity as a function of parallactic angle χ is described by

$$YY = I\{1 - p_{qu}\cos[2(PA - \chi)]\}$$

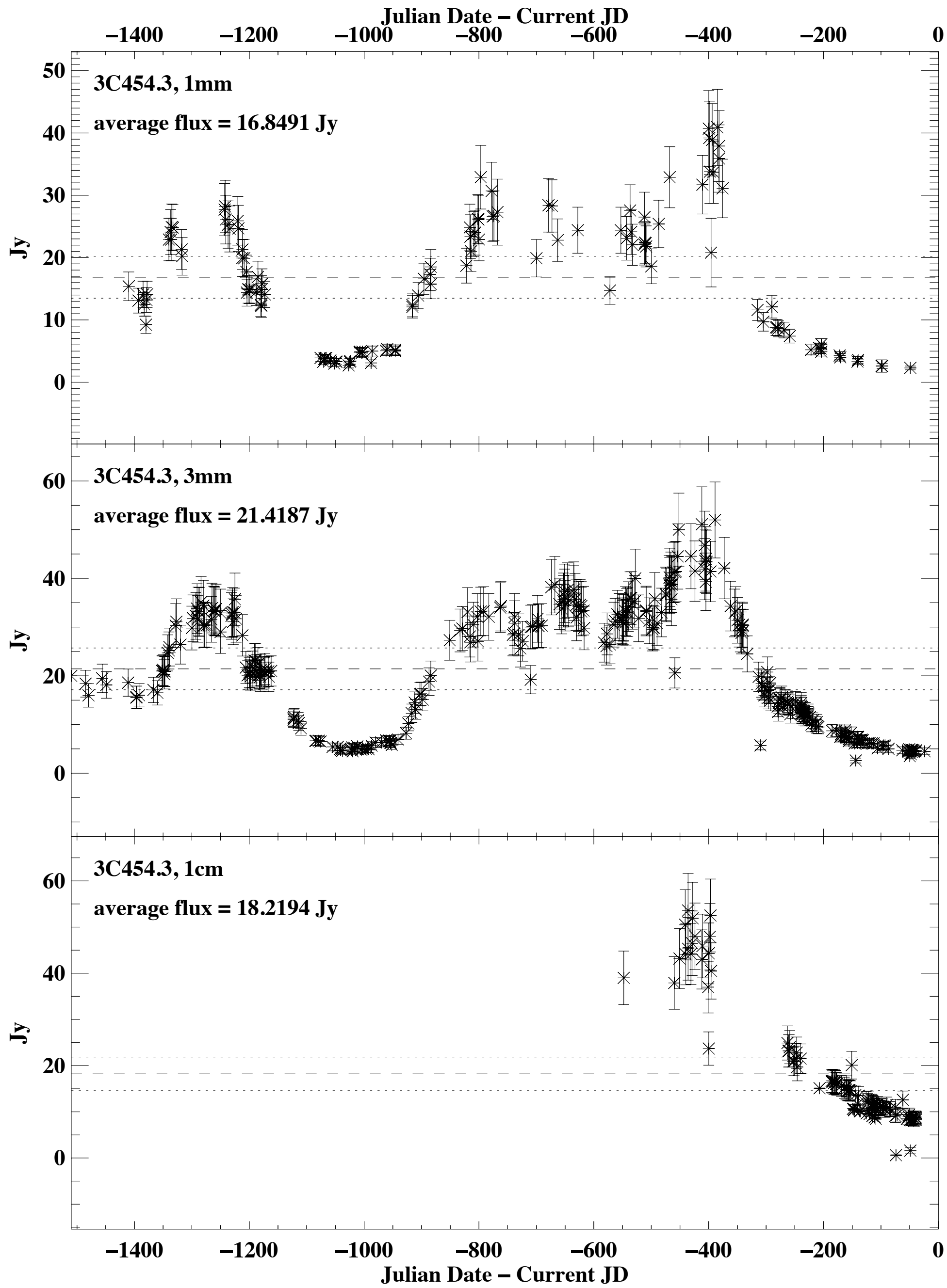

Fig. 1.— Historical fluxes of the standard calibrator 3C454.3 (taken from `FluxSource.cat`) in each of the three CARMA bands. Time is plotted as 'Julian Date - Current JD' where the 'Current JD' is February 8, 2012. The dashed line indicates the average flux, with dotted lines showing plus and minus 20%.

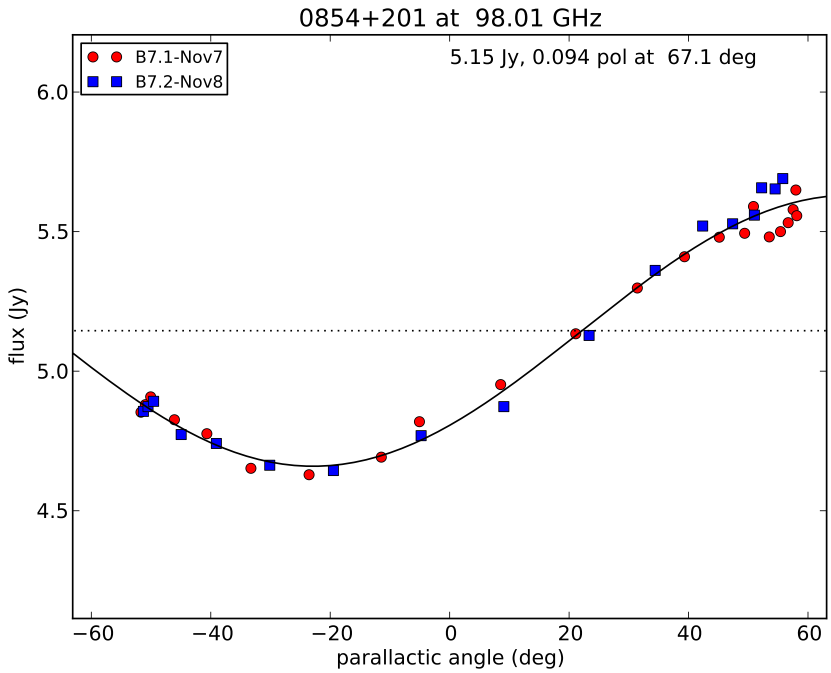

Fig. 2.— Flux of 0854+201 versus parallactic angle at 98 GHz on November 7-8 2011 from datasets c0834.7D_98B7.1 and c0834.7D_98B7.2.

This means that the 3mm flux of a polarized calibrator will appear to vary over the course of a track if the parallactic angle coverage is significant. This is illustrated in Figure 2, which shows the flux versus parallactic angle of 0854+201 at 98 GHz from two datasets in November 2011.

2.3. Elevation Dependence

We have noticed in the past that some datasets show evidence for an elevation dependence in the derived flux, presumably arising from elevation-dependent gains. We merely note this here but do not investigate this effect.

3. Dedicated Fluxcal Tracks

3.1. Sci1 (Woojin Kwon)

CARMA has been making efforts to take flux calibration data since its commissioning phase, regularly and when gaps in the science track schedule occur. The sci1 fluxcal script taking these dedicated flux calibration data is called `fluxcal.obs` and is located in the directory

∼obs/scripts/arrayHealth/ on the Cedar Flat computers. The script uses three specialized programs in the same directory: fluxcal_obsdef3.py, fluxcalinput3.py, and fluxcalutil.py. The first one is a variant of the CARMA program obsdef3.py, which provides desired command-line input parameters. In addition, initialization and pointing modules of the program are used in fluxcal.obs. The purpose of fluxcalinput3.py is to set the observation parameters to be the same as a normal CARMA observation script for a science project (such as elevation limits, pointing scheme, record time, etc.). The fluxcalutil.py script has utility modules for tuning, choosing primary, and sorting calibrators. The fluxcal script can be run anytime, as it figures out which primary and secondary flux calibrators are up in the sky when it is run. In order to maximize the number of calibrators observed in a given time, it sorts calibrators and first observes those setting in the western sky.

When running fluxcal.obs, observers can control the script behavior by several command-line input parameters[1] such as list=(weekly[2]/monthly/full/abstest), band=(both/1mm/3mm), fullstartover=(True/False), startprimary=(True/False), pnt=(True/False), alarm=(True/False), and endtrack. The weekly and monthly lists are named after the recommended running intervals and include four major secondary calibrators (MWC349, 3C84, 3C345, and 3C273) and a few tens of bright calibrators, respectively. These lists are good to be used to fill a gap of about an hour. On the other hand, the full list that includes all the calibrators in FluxSource.cat (212 as of June 2012) may be considered when relatively large gaps are expected or during array configuration changes, when many antennas are unavailable. The abstest list is for obtaining data designed for testing the mutual consistency of planet models like Mars, Uranus, and Neptune models. The 1mm and 3mm band parameters set the first LO to 230 and 106 GHz (as of June 2012) respectively, and the spectral windows (wide bands) are spread over the IF bandwidth. In the case of the both option selected, each calibrator is observed at 3 mm first. Note that the both option is not allowed for full list runs. The fullstartover parameter is valid only when the list is set to full. During a full list track running, calibrators are read from the fluxcalname1mm/3mm.list files and are written in the fluxcaldone1mm/3mm.list when they have been observed. In subsequent full list runs the done list calibrators are skipped until the fullstartover is set to True. Sometimes no primary flux calibrator is up in the sky at the beginning of a track. In such cases, the script ends immediately, with the suggestion that the observer use startprimary=False. When a primary becomes available in the middle of track, it is observed. Observers can also turn off the alarm notifying the end of track and specify the endtrack time in LST. Pointing can be turned off as well by pnt, but it is not recommended.

Calibrator fluxes are estimated through typical data reduction procedures. First, data sets are separated according to frequency (1 and 3 mm) and primary calibrator in case multiple primaries

[1]CARMA Wiki page: http://cedarflat.mmarray.org/twiki/index.php/Array_Health_Tasks#FluxCal. The original webpage can be found at: http://cedarflat.mmarray.org/fluxcal/howtofluxcal/

[2]Underlined options indicate the defaults.

have been observed: **uvcat**. Before or after the data split, bad data points are flagged. Generally, antennas without converged pointings need to be considered for flagging. In addition, it is helpful to limit *uv* distance if weather was bad. Then, each of the data subsets is bandpass-calibrated using the brightest calibrator: **mfcal**. After applying the bandpass solution to data (e.g., **uvcat**), calibrators are self-calibrated in phase only with a small interval (e.g., 0.5 minutes): **mselfcal** or **selfcal**. Finally, calibrator fluxes are estimated by **bootflux** in individual spectral windows. The uncertainties reported by **bootflux** are statistical errors of all the baseline values, which do not take into account weather changes between observations toward primary and secondary calibrators and are typically an order of magnitude smaller than practical uncertainties. Therefore, based on the minimum variation of calibrator fluxes monitored during the CARMA commissioning period (Kwon et al. 2006), a standard 15% uncertainty has been given all flux values so far. Note that this memo suggests a new scheme for a more reasonable error estimate (see Section 4.2.1, Step 11), but it cannot be applied for the dedicated fluxcal tracks as calibrators are observed only once. While all the calibrator fluxes of dedicated fluxcal tracks have been added into the CARMA catalog `FluxSource.cat`, those from weekly tracks have also been posted at a separate web page[3].

3.2. Sci2 (Tom Plagge)

The sci2 fluxcal script is called `fluxcal_sci2.py`, and is similar in functionality to its sci1 counterpart. By default, it runs at 1cm (35.938 GHz), and it requires sources to be above 30 degrees in elevation and 20 degrees or more away from the sun. The list of sources can be found in *scripts/arrayHealth/source.py*; the bright source list is used unless `list=full` is specified.

The script first observes all available 'abstest' targets: Mars, Uranus, Neptune, Jupiter, MWC349, and 3C84. Pointing is done first, and then each source is observed for `nrepSrc` integrations of `tintSrc` seconds (5 minutes total by default). After that, any other primary calibrators not part of the abstest list are observed in the same manner. Then the bright or full source list is observed until the specified endtrack. If the list of sources is complete and the endtrack LST has not yet been reached, the abstest sources and primary calibrators are observed again at the end of the script.

The reduction procedure for the sci2 fluxcal data is also very similar to sci1. The noise source, the inactive sideband, and the wideband visibilities (as opposed to the spectral channel visibilities) are first stripped from the MIRIAD file using **uvcat**. Flagging is then done based on shadowing, system temperatures, antenna-based gains (anything < 0.75 or > 1.5), and known hardware problems. The data are bandpass calibrated using a bright source (3C84 if available), and the MIRIAD file is then separated into sources. Each individual source is **selfcal**'ed, and **bootflux** is then run on the sources plus the primary calibrator, averaging over all windows. Mars is used as the primary

---

[3] http://cedarflat.mmarray.org/fluxcal/

calibrator if available, followed in order of precedence by Uranus, Neptune, and Jupiter. If there is no planet in the dataset, it is discarded. At this time, Brian Butler's model is used for Mars and the **smaflux** models are used for Uranus, Neptune, and Jupiter (since the MIRIAD models are known to be inaccurate at 1cm). At the time of this writing (June 2012), the CARMA fluxcal group is investigating better planet models, so this is likely to change when a new standard set of models is settled upon.

4. Fluxcal on Science Tracks

Prior to May 2012, the fluxes of secondary calibrators were extracted from science tracks with the interactive Python script `GetNewCalFlux.py`, written by Scott Schnee. Berkeley CARMA postdoc Frank Bigiel ran fluxcal on science tracks prior to February 2011, and Berkeley graduate students Amber Bauermeister, Statia Cook, Chat Hull and Katey Alatalo ran fluxcal on science tracks using this script until May 2012. This old script, `GetNewCalFlux.py`, was time-intensive to run and out-of-date, so the Berkeley graduate student team has developed a new fluxcal on science tracks reduction script, based on John Carpenter's flux extraction script, `fluxes.py`. The new script, `fluxcalOnScience.py`, has been running since May 2012, monitored by Amber Bauermeister. This task will be transferred to Manuel Fernández-López in the summer of 2012. The scripts and documentation for both the old and new fluxcal on science task can be found on the high site machines in ~`obs/fluxcalOnScience/`.

We describe the new system for fluxcal on science tracks in detail in Section 4.2. We also give a brief description of the old script in Section 4.1 to inform the interpretation of secondary calibrator fluxes prior to May 2012.

4.1. Flux Monitoring Prior to May 2012 (Manual)

The extraction of fluxes from science tracks prior to May 2012 was done using `GetNewCalFlux.py`. With this script, the secondary calibrators given in Table 1 were monitored (this list is hard-coded into the script). Only science datasets which had one of these secondary calibrators as well as an acceptable primary calibrator (Mars, Uranus or Neptune) were processed.

The `GetNewCalFlux.py` script by Scott Schnee performs a simple calibration of each dataset, extracting the flux of the secondary calibrator in the first wide-band window in the dataset using **bootflux**, which uses the planet brightness temperature written in the dataset by the CARMA control system. The script displays plots of amplitude versus time and phase versus time on the calibrator, and amplitude versus uv distance on the planet so that the user can determine if any antennas or baselines are bad. The user must then flag bad data by hand and re-run the script. The error in the flux is set to either 15% of the flux value or the rms variation in the flux over the course of the track. The larger value is used, which is almost always 15% of the flux value.

calibrator name	3mm flux (Jy)	calibrator name	3mm flux (Jy)
0136+478	2.35	1911-201	2.00
0238+166	1.93	1924-292	10.18
0359+509	7.28	2148+069	2.99
0423-013	5.22	2229-085	2.72
0457-234	-	2232+117	2.67
0522-364	-	3C84	10.71
0530+135	2.26	3C111	3.12
0730-116	3.99	3C120	1.82
0854+201	4.60	3C273	14.34
0927+390	4.83	3C279	14.90
1058+015	3.81	3C345	4.43
1337-129	3.66	3C446	4.21
1517-243	1.81	3C454.3	22.18
1658+076	1.43	BLLAC	4.34
1733-130	3.93	MWC349	1.23
1751+096	4.51		

Table 1: Secondary calibrators monitored by fluxcal on science tracks prior to May 2012, with average 3mm fluxes from `FluxSource.cat` (from March 2007 to February 2012). No historical fluxes were found for 0457-234 or 0522-364.

Fluxes were extracted in this way for all data files on the high site which began with 'cx', 'c0' or 'c1' (science tracks) and which had acceptable primary and secondary calibrators (see above). We also made a cut based on the length of the track (typically must be longer than 1 to 2 hours) and the weather grade (B and above typically accepted). This flux monitoring was kept up on a weekly basis: every Friday, one of the members of the fluxcal on science tracks team ran this analysis on the datasets from the past week on the high site computers. A rotation was kept so that each of the four team members did this task approximately monthly. More information on the week-to-week upkeep of fluxcal on science tracks can be found on the team's wiki site here:
http://badgrads.berkeley.edu/doku.php?id=carma_fluxcal

4.2. Flux Monitoring After May 2012 (Automated)

The automatic extraction of fluxes from science tracks is done on the archive computers at Illinois as tracks are copied to the archive. The script that handles the automated reduction is `fluxcalOnScience.py`, which is adapted from John Carpenter's automatic flux extraction script, `fluxes.py`. The reduction method in `fluxcalOnScience.py` is mostly the same as the original script, with added functionality for amplitude flagging, spectral index fitting and logging. In the following sections we describe the method in which fluxes are extracted by this script and then give an overview of how the script is used.

4.2.1. Method

The main steps of the reduction (with intermediate data files indicated in bold) are as follows

1. **Initial evaluation:** Determine the sources observed, the track length and weather grade. The weather grade is calculated from the average rmspath and tau230 values using the method in the quality script. Tracks must meet the following criteria in order to move forward:
   - Track is a science track: filename starts with 'cx', 'c0' or 'c1'.
   - Track must have a weather grade of B- or better.
   - Track must be at least 1 hour long.
   - Track must contain an acceptable primary calibrator: Mars, Uranus or Neptune.
2. **Remove bad windows** for sci2 and CARMA23 observations, writing **vis_sci2.mir**:
   - Sci2, 1cm: remove USB, windows 17-32
   - Sci2, 3mm: remove LSB, windows 1-16
   - CARMA23 (3mm): remove LSB, windows 1-8

   Note that as of this writing (May 2012), band 16 of the wide-band correlator is bad. Therefore, we remove windows 16 and 32 from sci2 tracks as well.
3. **Extract wide-band windows:** Read the correlator configuration and extract only the wide-band windows to **wb.mir**. At this point, we also extract only the LL polarization.
4. **Basic flagging:**
   - Flag shadowed antennas using **`csflag`**
   - Flag data at elevations above 85 degrees.
   - Flag 3 edge channels on each side of each window (controlled by EDGECHAN_FLAG)
5. **Flag spuriously large amplitudes:** This is done on each calibrator individually, with planets evaluated differently from other calibrators.
   - **Planets:** Use **`uvmodel`** with `options=divide` to divide the uv data by the planet model, writing **wb_planet.mir**. An average amplitude is calculated for each baseline using **`uvaver`** with `interval=10000`, writing **wb_planet_avg.mir**. Note that the averaging method used for non-planet calibrators cannot be used here since **`smauvplt`** does not average planet data. The data divided by the model should have amplitude of 1 in the

case of perfect data, so we consider any average data points outside (0.5,2.0) bad. Note that we only evaluate points within the first null since division by the model at the nulls results in large values which are not necessarily bad.

- **All other calibrators:** The planet data are first removed from the dataset, writing **wb_noplanet.mir**, so that averaging can be done with **smauvplt**. The average amplitude of each baseline is extracted using **smauvplt** with `average=10000` and `options=scalar`. The minimum acceptable amplitude is set to 0.5 times the median of the average amplitudes (this is controlled by allowAmpFac_min). The maximum acceptable amplitude is set to the minimum of 1.5 times the median (controlled by allowAmpFac_max) and the absolute cut value. Historically (as recorded in `FluxSource.cat`) a flux over 55 Jy has not been observed. Therefore, we set the absolute cut value to 100 Jy for sources 3C273, 3C279, 3C454.3, 3C84 (historically the brightest) and 50 Jy otherwise. This absolute cut is in place in order to catch cases of extremely bad data which would drive the median amplitude so high that the bad data would not be caught by this method.

Based on the criteria described above, in each window, each baseline in each calibrator observation (time slice) is evaluated as good or bad. The script then searches for trends in the bad points:

- if $>$ 25% of time slices are bad for a given baseline, the baseline is flagged
- if $>$ 25% of baselines in a time slice are bad, the time slice is flagged
- if $>$ 25% of baselines with a particular antenna are bad in a particular time slice, flag the antenna in that time slice
- if an antenna is bad in $>$ 25% of time slices, flag the antenna for all times

The parameter trendFraction controls the fraction (25%) of present data points above which a badness trend is established. For each window, if at least one time slice remains not flagged, with at least 50% of antennas not flagged, the window is considered good. Each calibrator is flagged for spurious amplitudes separately, but if a window is found to be bad for any calibrator, it is flagged for the entire dataset.

6. **Passband calibration:** An initial passband and phase-only gain calibration is performed using **mfcal** in order to calculate the signal to noise on each calibrator. The (non-planet) calibrator with the highest signal to noise is then used to calibrate the passband with **mfcal** using an interval of 0.5 minutes (controlled by parameter INTERVAL_PB). Antennas with average gain amplitudes within $\pm$30% of the median gain amplitude are considered 'good'. If there are at least three 'good' antennas and the reference antenna is one of them, the passband calibration is considered successful. If successful, **pb.mir** is written, removing the 'bad' antennas. If not successful, see step 10.

7. **Phase-only self-calibrate** the planet and each calibrator separately using **selfcal** with an interval of 0.5 minutes (controlled by parameter INTERVAL_FLUXCAL), writing **primary.mir** and **source.mir** respectively.

8. **Derive the flux** of each calibrator in each window using **bootflux**. The planet brightness temperature is set for each window separately, using the following planet models:

   - Uranus: (134.7 K)$(\nu/100\ GHz)^{-0.337}$ (CARMA control system power law)
   - Neptune: (129.8 K)$(\nu/100\ GHz)^{-0.350}$ (CARMA control system power law)
   - Mars: For 3mm and 1mm data, the seasonal variation model returned by MIRIAD program **marstb** is used (this model does not currently include diurnal variations). When running on the high site, the seasonal and diurnal variation model returned by `szaCalcMars` is used for the 1cm data. When running at Illinois, the 1cm flux is extrapolated using a straight line calculated from **marstb** values at 43 and 50 GHz, which fairly closely matches the `szaCalcMars` values.

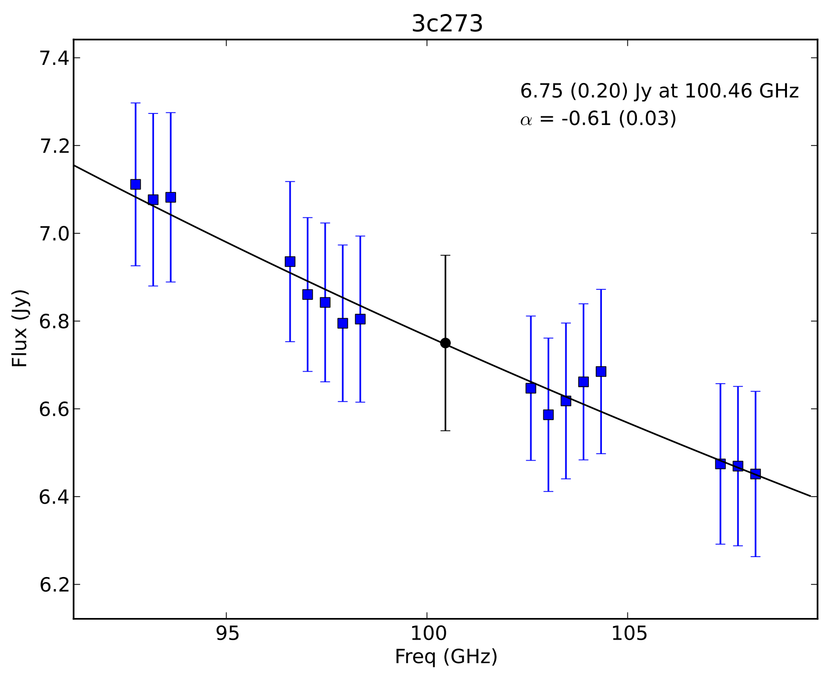

Fig. 3.— Flux versus frequency for calibrator 3C273 in the dataset `c0834.5D_97B5.2.miriad`, from the `fluxcalOnScience.py` script. The blue points show the flux (with error bars showing the rms time variation) in each window. The black line shows the fit, with the derived flux (and error) at the central frequency indicated by the black point and error bar. The flux, flux error, spectral index (α) and spectral index error from the fit are listed in the top right corner of the plot.

N_t	N_{wb}	flux	σ_{flux}	α	σ_α
1	1	`bootflux` value	-999.99	-	-
1	2	average value	-999.99	slope	-999.9
1	> 2	fit value	-999.99	fit value	fit error
> 1	1	median `bootflux` value	rms time variation	-	-
> 1	2	average value	rms time variation	slope	-999.99
> 1	> 2	fit value	time rms + fit error	fit value	fit error

Table 2: Summary of what is reported (flux, flux error (σ_{flux}), spectral index (α), spectral index error (σ_α)) based on the number of wide-band windows (N_{wb}) and the number of times a calibrator is observed in a track (N_t). No spectral index is reported when only one wide-band window is present (indicated by -). When we are unable to accurately estimate the error in the flux or spectral index, the padding value of -999.99 is reported. For calibrators with > 1 time slice and > 2 wide-band windows, the flux error reported is the rms time variation and the fit error added in quadrature.

9. **Fit the spectral index:** If there are enough wide-band windows, calculate the spectral index, α ($flux \propto \nu^\alpha$). If there are 2 wide-band windows, calculate α directly. If there are more than 2 wide-band windows, perform a least-squares fit on the fluxes in each window (using errors reported by `bootflux`). The fit gives the spectral index (and error) and the flux at the central frequency (and error). As an example, the resulting fit for calibrator 3C273 in dataset `c0834.5D_97B5.2.miriad` is shown in Figure 3.

10. **Success?** If at least one calibrator flux was derived successfully, the script is finished. If not, the script will loop through steps 6 through 9 again using a different reference antenna. The antennas are ordered by median distance to other antennas, and used as the reference antenna starting with the smallest median distance. This is done in order to minimize atmospheric decoherence, which will have a larger effect on longer baselines.

11. **Report:** Table 2 describes how the flux error, spectral index and the spectral index error are reported. In reporting the error in the flux, we find that the variation in the measured flux with time due to the weather and system is orders of magnitude larger than the error reported by `bootflux` based on the phase scatter of the data. Therefore, we only report an error when the calibrator is observed multiple times over the course of the track. In this case, we calculate the standard deviation of the flux measurements in each window, and use the largest of these standard deviations as the error in the flux at the central frequency. When a fit is performed, we add the reported fit error in quadrature with the time variation error, but note that the effect of this is negligible since the fit error is always very small due to the nature of the fitting.

notScience	track does not start with ‘c0’, ‘c1’ or ‘c2’
Len/Grade	track is shorter than 1 hour or has a weather grade worse than B-
NoPrimCal	track does not contain a primary calibrator (Mars, Uranus or Neptune)
noConfig	track does not contain a correlator configuration (in **`uvindex`** output)
noWBwin	track does not contain any wide-band windows
allAmpFlag	all of the data were flagged in the amplitude flagging step
configProb0/1/2	more than one correlator configuration in the original file / after wide-band window extraction / after calibrator extraction
passband	unable to successfully passband calibrate with any refant
noGoodFluxes	unable to extract good fluxes with **`bootflux`**
planetRes	**`bootflux`** crashed with a fatal error, likely because the planet is resolved
bootfluxVar	the **`bootflux`** values varied by more than 25% for that calibrator (in this case, this error will be logged for each offending calibrator, with the number of offending observations in parenthesis next to the calibrator name)

Table 3: Failure modes for automated fluxcal on science tracks script, `fluxcalOnScience.py`. If any of these are encountered (except bootfluxVar), fluxes are not derived for the calibrators in the dataset.

4.2.2. Logging and Failure Modes

The script outputs the derived fluxes (to be entered into `FluxSource_newadd.cat`) to the file `fluxcal.cat`. For each dataset, basic information and the nature of the rejection of the dataset (failure mode) is output to the file `fluxcal.log` so that those managing fluxcal on science tracks can monitor how well the script is extracting fluxes and be alerted to data issues. The possible failure modes are described in Table 3.

4.2.3. Verification

Before transitioning to the new, automated script, the Berkeley team made several comparisons between the fluxes derived with the new script and the old script. We have done a final comparison of one week's worth of datasets, with fluxes derived using the old and new scripts. The result is shown in Figure 4. Most of the fluxes agree within 10%. It is worth noting that two of these outliers come from one dataset, which we carefully reduced by hand to find that the new script was finding the correct flux.

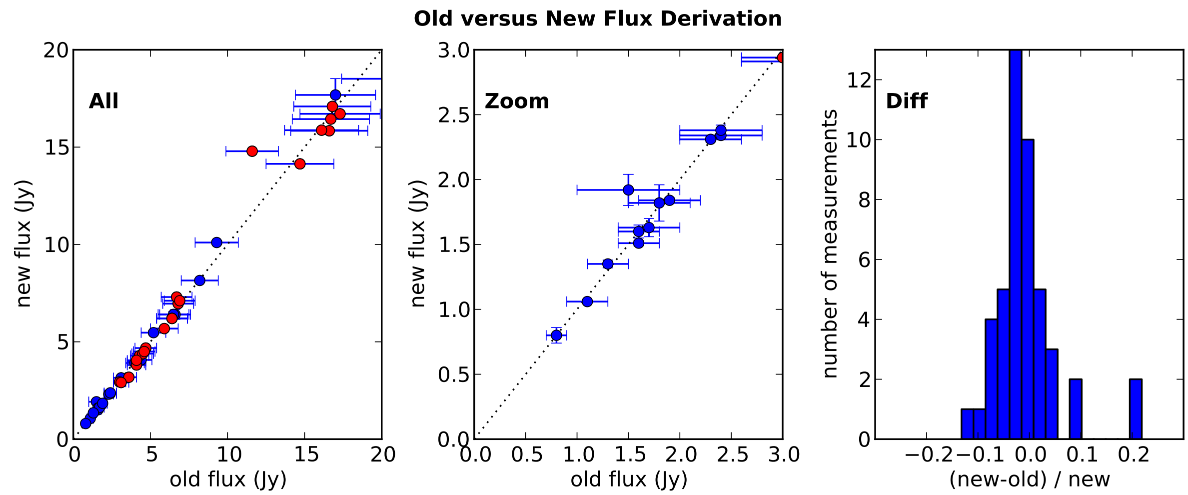

Fig. 4.— Comparison of the fluxes derived by the old and new scripts. In the left and middle panels, the old flux is plotted against the new flux, with one-to-one correspondence indicated by the black dotted line. Errors are plotted in both dimensions when available. The old script flux errors are set to 15% for all calibrators. Points for which an error is not reported by the new script (see Table 2) are plotted in red. The middle panel is the same as the left panel, showing only fluxes less than 3.0 Jy. The right panel shows a histogram of the fractional difference ((new-old)/new) in the derived flux, demonstrating that the old and new scripts almost always agree within 10%.

4.2.4. Management

The task of managing fluxcal on science tracks now consists of looking over the derived fluxes and associated logfile once a week to check that things are running smoothly and investigate any strange behavior that arises. Before final deployment, the vetted flux data in `fluxcal.cat` must be copied and added to the temporary holding catalog `FluxSource_newadd.cat` in MIRIAD CVS. The nightly script **`calbuild`** by Ted Yu merges `FluxSource_newadd.cat` into `FluxSource.cat`, the master flux table.

5. Summary

The fluxes of secondary calibrators are monitored in two ways at CARMA: through dedicated fluxcal tracks and by extracting calibrator fluxes from science tracks. In this memo we have described how each of these is carried out, emphasizing the changes recently implemented in flux monitoring using science tracks. We also describe how these fluxes may be used by a CARMA user and how the historical flux data can be accessed. We discuss both the intrinsic variability of calibration sources (the motivation for flux monitoring) as well as system-induced variations.

REFERENCES

Woojin Kwon, Leslie Looney, and Edmund Sutton, 2006, Report on CARMA flux calibration http://cedarflat.mmarray.org/fluxcal/reportflux20061211.pdf

A. Example Script for Flux Calibration

In this appendix, we go through the steps required to set the flux scale of a dataset. This is given as well in the script `fluxtest` in the `$MIR/src/scripts/` directory of your MIRIAD repository. The test dataset used is `flux_test.miriad`, which may be found in `$MIR/data/` (if it is not there, instructions for downloading it can be found in the README file in the `$MIR/data/` directory). The test dataset may also be downloaded from ftp://ftp.astro.umd.edu/pub/carma/data.

First, a bit of set-up: set test file name, calibrator names and refant:

```
set visfile = flux_test.miriad
set gaincal = 3C273
set bandpasscal = 3C273
set primcal = MARS
set refant = 8
```

Now we go through 7 steps to flux-calibrate our dataset.

1. Make a copy of the dataset to work on:

```
if (-e orig.mir) then
  rm -rf orig.mir
endif
uvcat vis=$visfile out=orig.mir
```

2. Calibrate the bandpass with **mfcal**:

```
mfcal vis=orig.mir interval=0.5 refant=$refant select='source('$bandpasscal')',-auto
```

   Look at the passband solution if desired:

```
smagpplt vis=orig.mir device=/xs options=bandpass,nofit xaxis=ch yaxis=amp yrange=0,3
smagpplt vis=orig.mir device=/xs options=bandpass,nofit,wrap xaxis=ch yaxis=ph yrange=-180,180
```

3. Extract planet and gain calibrator data, applying the bandpass solutions:

```
if (-e prim.mir) then
  rm -rf prim.mir
endif
uvcat vis=orig.mir out=prim.mir options=nocal select='source('$primcal')',-auto
if (-e gaincal.mir) then
  rm -rf gaincal.mir
endif
uvcat vis=orig.mir out=gaincal.mir options=nocal select='source('$gaincal')',-auto
```

4. Phase-only **selfcal** planet and gain calibrator data:

```
selfcal vis=prim.mir refant=$refant select=-auto interval=0.5 options=phase,apriori
selfcal vis=gaincal.mir refant=$refant select=-auto interval=0.5 options=phase,apriori
```

   Look at the solution if desired:

```
gpplt vis=prim.mir device=/xs yaxis=ph nxy=5,3 yrange=-180,180
gpplt vis=gaincal.mir device=/xs yaxis=ph nxy=5,3 yrange=-180,180
```

5. Use **bootflux** to get the flux of the gain calibrator.

   (a) First average all windows together:

```
bootflux vis=gaincal.mir,prim.mir select='source('$gaincal,$primcal')',-auto
  line=chan,1,1,156,156 taver=10 primary=$primcal
```

   This gives an average flux of 6.573 Jy at 100.5 GHz.

   (b) Now try doing each window individually using **marstb** to get the appropriate brightness temperature. From **uvlist**, we calculate the central frequencies of the four windows: 98.34, 92.74, 102.59, and 108.19 GHz. Now we get the brightness temperature for Mars at each of these frequencies using **marstb**:

```
foreach f (98.34 92.74 102.59 108.19)
  marstb epoch=12APR13:04:40:11 freq=$f
end
```

   This gives brightness temperatures (K): 195.145, 194.865, 195.355, 195.626. For reference, the brightness temperature in the dataset (at the LO frequency) is 195.011 K (stored in variable pltb: `varplt vis=prim.mir device=/xs yaxis=pltb log=pltb.log`) Now we can run **bootflux** in each window individually, forcing the desired brightness temperature using `primary=PlanetName,Tb` via the following commands:

```
bootflux vis=gaincal.mir,prim.mir select='source('$gaincal,$primcal')',-auto
  line=chan,1,1,39,39 taver=10 primary=$primcal,195.145
bootflux vis=gaincal.mir,prim.mir select='source('$gaincal,$primcal')',-auto
  line=chan,1,40,39,39 taver=10 primary=$primcal,194.865
bootflux vis=gaincal.mir,prim.mir select='source('$gaincal,$primcal')',-auto
  line=chan,1,79,39,39 taver=10 primary=$primcal,195.355
bootflux vis=gaincal.mir,prim.mir select='source('$gaincal,$primcal')',-auto
  line=chan,1,118,39,39 taver=10 primary=$primcal,195.626
```

   ...or, for a quick printout of the average fluxes, use the following:

```
set flux1 = `bootflux vis=gaincal.mir,prim.mir select='source('$gaincal,$primcal')',-auto
  line=chan,1,1,39,39 taver=10 primary=$primcal,195.145
  | grep 'Average Flux:' | awk '{ print $3 }'`
set flux2 = `bootflux vis=gaincal.mir,prim.mir select='source('$gaincal,$primcal')',-auto
  line=chan,1,40,39,39 taver=10 primary=$primcal,194.865
```

```
  | grep 'Average Flux:' | awk '{ print $3 }'`
set flux3 = `bootflux vis=gaincal.mir,prim.mir select='source('$gaincal,$primcal')',-auto
  line=chan,1,79,39,39 taver=10 primary=$primcal,195.355
  | grep 'Average Flux:' | awk '{ print $3 }'`
set flux4 = `bootflux vis=gaincal.mir,prim.mir select='source('$gaincal,$primcal')',-auto
  line=chan,1,118,39,39 taver=10 primary=$primcal,195.626
  | grep 'Average Flux:' | awk '{ print $3 }'`
echo "Average Flux in Each Window:"
echo "  win1: $flux1 Jy at 98.34 GHz"
echo "  win2: $flux2 Jy at 92.74 GHz"
echo "  win3: $flux3 Jy at 102.59 GHz"
echo "  win4: $flux4 Jy at 108.19 GHz"
```

This should give:

```
Average Flux in Each Window:
  win1: 6.608 Jy at 98.34 GHz
  win2: 6.883 Jy at 92.74 GHz
  win3: 6.508 Jy at 102.59 GHz
  win4: 6.263 Jy at 108.19 GHz
```

6. Compare with `FluxSource.cat` values. For datasets with primary flux calibrator observations, it is a good idea to compare the value you get to the historical record of fluxes in `FluxSource.cat`. If your dataset does NOT contain a primary flux calibrator observation, you may have to use the fluxes in `FluxSource.cat` to determine the flux of your gain calibrator.

   This example dataset is from April 13, 2012. Using the calfind website, we see that the average flux in the 3mm band from March 12, 2012 to May 20, 2012 is 6.78 +/- 0.63, which agrees fairly well with our average value of 6.573 Jy at 100.5 GHz

   Specifically, the `FluxSource.cat` entries from the days immediately surrounding the date of our observation are:

3C273	2012-APR-11.0	98.5	6.70	1.00	CARMA8	MARS	-999.99	-999.99	-999.99
3C273	2012-APR-12.0	99.2	6.60	1.00	CARMA8	MARS	-999.99	-999.99	-999.99
3C273	2012-APR-13.0	98.5	6.70	1.00	CARMA8	MARS	-999.99	-999.99	-999.99
3C273	2012-APR-13.0	100.8	5.10	0.80	CARMA8	NEPTUNE	-999.99	-999.99	-999.99
3C273	2012-APR-15.0	99.2	6.50	1.00	CARMA8	MARS	-999.99	-999.99	-999.99

   These agree except for the measurement of 5.10 Jy at 100.8 GHz.

   Another strategy of comparison that may be useful is to use the spectral index information in `FluxSource.cat`, if available. The example dataset was taken before spectral indices were written in `FluxSource.cat`, but as an example, we may look at the first spectral index available for 3C273 in the 3mm band:

3C273	2012-MAY-12.0	107.5	6.28	0.20	CARMA12	MARS	-999.99	-0.80	0.07

A flux of 6.28 Jy at 107.5 GHz with a spectral index of -0.80 implies the following fluxes at the central frequencies of our windows:

win#	freq	our flux	expected flux
win2:	92.74 GHz	6.883 Jy	7.07 Jy
win1:	98.34 GHz	6.608 Jy	6.74 Jy
win3:	102.59 GHz	6.508 Jy	6.52 Jy
win4:	108.19 GHz	6.263 Jy	6.25 Jy

After comparing with `FluxSource.cat` values, we conclude that our derived fluxes are good and we will use these to set the flux scale of the dataset.

7. Set the flux scale of the dataset by performing an amplitude **selfcal**.

   First copy the dataset in order to apply the bandpass solution:

```
if (-e pb.mir) then
  rm -rf pb.mir
endif
uvcat vis=orig.mir out=pb.mir options=nocal select=-auto
```

   Now do amplitude **selfcal** with an interval of 18 minutes (approximate length of the source-gain cal cycle). Which window(s) you use to do this will depend on your particular science goals.

   If our science is in window 2 (for example), we can run **selfcal** on that window only, using our flux estimate at the central frequency of window 2:

```
selfcal vis=pb.mir refant=$refant select='source('$gaincal')',-auto interval=18
  options=amplitude,noscale flux=6.883 line=chan,1,40,39,39
```

   Alternatively, we could use all the channels and use the average flux we found above (6.573 Jy):

```
selfcal vis=pb.mir refant=$refant select='source('$gaincal')',-auto interval=18
  options=amplitude,noscale flux=6.573 line=chan,1,1,156,156
```

   look at the solution if desired:

```
gpplt vis=pb.mir device=/xs yaxis=amp nxy=5,3 yrange=0,4
gpplt vis=pb.mir device=/xs yaxis=ph nxy=5,3 yrange=-360,360
```

   Now pb.mir is fully calibrated with the correct flux scale. You are ready to make images and measure the flux of your science target with confidence! Remember though that any flux measurement will have an inherent uncertainty of 20% due to the uncertainty in the models of the primary flux calibrators.