This draft November 20, 2009

How to run “automatic” flux calibration

Scott Schnee

1. Introduction

This document is meant to explain how to derive fluxes from recently run science projects and add these values to the CARMA/MIRIAD catalogs.

Scott Schnee wrote these python scripts with significant help from Stuartt Corder.

There are three separate steps to accomplish this task:
1) Determine which science tracks have primary and secondary calibrators
2) Determine the fluxes of the secondary calibrators
3) Add these values to CARMA’s catalog of fluxes

2. Which tracks are interesting?

The idea here is to look through the last few science tracks and check which have primary calibrators (Mars, Uranus or Neptune) and a bright secondary calibrator (usually a quasar). Then we can use the known planetary model to derive the flux of the quasar, which can vary significantly with time. Note that “science tracks” here refer to those that begin with the “c0”, “cx” or “c1” (to plan for the future) project codes. All other tracks will be ignored, so that we don’t waste time checking through the “ct”, “rpnt” and other irrelevant tracks.

The python script to look through the recent science tracks is in:
/home/obs/schnee/1mm_flux/GetNewCalFlux.py
on the cedarflat machines at the high site.

It is run using the following example set of commands:
(at prompt) python (to get into python)

```
>> import GetNewCalFlux
>> GetNewCalFlux.getTracksWithFlux(addLastN=10)
```

The only parameter that getTracksWithFlux takes is “addLastN” and it tells the script how many tracks to look through, in this case, the last 10 tracks. In a typical day, the observers will run around 5 science tracks, so if you haven't done this task in a couple days, 10 is a reasonable number to choose.

The output of this get appended to the end of:
/home/obs/schnee/1mm_flux/FluxTable.txt

Here is an example output:

```
c0310.8E_230_21to2.6.mir 09JUL07 92.5098 GHz MARS 2148+069 3C454.3
c0291.1E_230Betelg.5.mir 09JUL07 227.7797 GHz MARS 3C84 3C120
```

The first column shows the file name (the file is in /opt/sdp/sciencedata), the second column shows the date of the observations, the third column shows the frequency of the observations, the fourth column shows the units of frequency (GHz), the fifth column shows the name of the primary calibrator, and the subsequent columns has the names of the bright quasars in the tracks.

Incidentally, to know exactly how many tracks to look through, just look at the last track in the FluxTable.txt. Then, do an "ls" to see which files are in the science data directory:

```
ls -ltrd /opt/sdp/sciencedata/c0*.mir /opt/sdp/sciencedata/cx*.mir
```

and count back the number of files until you reach the last one in FluxTable.txt. That number is what value you can set addLastN to when you run GetNewCalFlux.getTracksWithFlux().

3. What is the flux of the quasar?

So, now you know which tracks you are interested in, so let's determine the fluxes of the quasars. From the previous example we know that the file c0310.8E_230_21to2.6.mir has a planet and some quasars, so copy it to whatever directory you want to work in. For instance, this could work:

```
cd /home/obs/schnee/1mm_flux
cp -rf /opt/sdp/sciencedata/c0310.8E_230_21to2.6.mir .
```

Now, use *uvindex* to determine which quasars are in the track:

uvindex vis=c0310.8E_230_21to2.6.mir

There are several, but the brightest is probably 3C454.3, so let's start with that. In the same python window as before, type:
>> GetNewCaluFlux.doFluxCal('c0310.8E_230_21to2.6.mir', 'MARS', '3C454.3', '3C454.3")

The first string is the file name. The second is the name of the primary flux calibrator. The third is the name of the secondary calibrator. The fourth is the name of your passband calibrator. There are also several optional parameters that doFluxCal() can take, which will be described later.

This routine will do all the calibration that's needed to determine the flux of the secondary calibrator. You will get several screens of graphical output, in this order:
1) Flux vs time for the secondary calibrator for all antennas for the first wideband window
2) Flux vs time on each baseline separately for the first wideband window
3) Phase vs time on each baseline
4) Flux vs baseline length for the primary calibrator

You will then see the output from *bootflux* in the form of a table. If you liked the results (i.e., no time ranges or antennas need to be flagged and the weather or other problems didn't ruin the track), then you can have doFluxCal() write some data to later be put into the CARMA/MIRIAD official catalogue. If you trust the results, enter 'Y' (capital Y, without the quotes) when asked "Add values to personal files? [Y/N]". Otherwise, type anything other than 'Y' ('N' would do just fine).

The "personal files" this is referring to is "science_fluxes.txt", in the same directory that you are running python in.

3.1. Flagging

If you want to clean the data up before adding the new flux value to the table of fluxes, then can quit out of doFluxCal() with a "control-C" whenever you like. Then, use *uvflag* to flag the bad times or antennas or whatever else you like, and run doFluxCal() again.

3.2. doFluxCal Options

There are several options when running doFluxCal(). They are, along with the defaults:
1) windowList=[1,2,3,4,5,6] - If you don't want it to check all windows, you can telll doFlux-

Cal() which windows to consider. Either way, it stops after getting the flux from one window.
2) plots="Y" - If you don't want to see the plots, then add plots="N" to the call to doFlux-Cal()
3) refAnt=8 - If you want to use a different reference antenna for the phase calibration, use this parameter
4) badRes=25.0 - This is a parameter used by *bootflux*. In this case, the lowest 25% of the primary calibrator's flux measurements are not used when fitting the planetary model to the data. Since the signal to nosie is worse for these points, you don't necessarily want then included in the fit.

4. How do I add these values to the CARMA catalogue?

First, install MIRIAD on your computer, as explained on Peter's webpage:
http://www.astro.umd.edu/˜teuben/miriad/

Now, go to where the catalogue files are kept. For instance, on my machine it is in:
˜myproject/miriad/cat/

The file with the new quasar fluxes is "FluxSource_newadd.cat". To add the new fluxes that you just arrived to the official MIRIAD/CARMA catalogues, we use this three-step CVS process.

1) Make sure that your copy of FluxSource_newadd.cat is the most current one by typing "cvs update FluxSource_newadd.cat"
2) Add the fluxes in science_fluxes.txt to the end of FluxSource_newadd.cat using the text editor of your choice.
3) Add these fluxes to the official list by typing "cvs commit -m 'science fluxes added' Flux-Source_newadd.cat"

At this point you don't need science_flux.txt anymore, so you can delete it or rename it or move it to another directory for future reference. You may as well delete all but the last line of FluxTable.txt, since you won't be going back to the old files again.

The next time there is a build, the new fluxes from FluxSource_newadd.cat will be automatically put in the CARMA/MIRIAD catalogue of fluxes.

You're done! Whew.

4.1. Technicalities ...

1) For installing miriad, there is elaborate information on:
http://carma.astro.umd.edu/wiki/index.php/Miriad

In particular, one has to make sure that the necessary requirements are met:
http://carma.astro.umd.edu/wiki/index.php/Requirements

E.g. on a mac, for miriad to run it might be necessary to install a fortran compiler (see 'HPC compiler' in the above link)

2) In order to update miriad (or just individual files) using CVS (e.g. to use the above 'cvs update FluxSource_newadd.cat'), it will be necessary to type 'cvs login' followed by 'enter' (blank password). This is only necessary once and updating should work fine thereafter.

3) Uploading the new, updated FluxSource_newadd.cat via 'cvs commit ...' is slightly more complicated and does not work straight out of the box. Generally, a very good description of the necessary steps can be found at:
http://www.astro.umd.edu/ teuben/miriad/cvs.html

Here is, however, a (slightly compressed) step by step description: First, one needs to generate an encrypted password using the perl command on the above cvs webpage (and don't forget the apostrophes!). The output of this command (as well as a desired user name) needs to go to peter teuben (teuben@astro.umd.edu) as he will create an account that will allow to update (upload) files via CVS.

The following two lines need to be added to .cshrc:
setenv CVSROOT :pserver:USER@cvs.astro.umd.edu:/home/cvsroot
setenv CVSEDITOR emacs where USER is the respective user name.

(the same lines for the bash shell are:
export CVSROOT=:pserver:USER@cvs.astro.umd.edu:/home/cvsroot
export CVSEDITOR=emacs)

Eventually, the entry (should only be one line) in the file 'Root' in miriad/cat/CVS/ needs to be changed: the 'anonymous' user needs to be changed to the respective user name (this enables committing changes via CVS only for the CAT directory, which should be sufficient)

Then, type 'cvs login' and provide the (above chosen) password. This has to be done only once (the password will be stored in .cvspass in the home directory).

...done! from now on, 'cvs update' should work for everything and 'cvs commit' should work in the CAT directory.