User Documentation for The MAST/GALEX Photon Database and Tools

Chase Million[1], Bernie Shiao[2], Scott Fleming[2], Myron Smith[2]
1. Million Concepts, chase.million@gmail.com, 2. Space Telescope Science Institute

Table of Contents

- Raison d'Etre
- Getting Started
- Installation Instructions
    - Linux
    - Mac
    - All Operating Systems
- Testing Your Build
- gPhoton
    - Convert GALEX Time to UNIX Time
- Database Tools
  - gFind Data Location Tool
    - Basic Usage
    - Tweakable Search Parameters
    - Alternate I/O Formats
  - gAperture Photometry Tool
    - Basic Usage
    - Relative Response
    - High Quality Background
    - Recommended Usage
    - Creating a JPEG Stamp
    - Alternate Input Formats
    - Coadding
    - Verbosity and Headers
    - gAperture CSV File Column Defintions
  - gMap Image Creation Tool
    - Basic Usage
    - Image Headers
- Common Questions, Issues, and Gotchas
  - "My data is not available."
  - Exposure Time
    - Shutter Correction
    - Deadtime Correction
      - Method 1 - Direct Measurement
      - Method 2 - Empirical Formula
    - Effective Exposure Time
  - Relative Response
  - Gain Sag

Raison d'Etre

The MAST/GALEX photon database and tools ("standalone pipeline") exist in an effort to maximize the flexibility and utility of the GALEX data set. The GALEX detectors are microchannel plates which record detector position and time-of-arrival information for every detected photon event with a time resolution of five thousandths of a second, composing a huge and rich short time domain survey of the UV. Due to digital storage space and processing limitations, the data was formally released as integrated images. The photon list files--internally known as x-files--were only available by special request to the GALEX team, and there was little to no additional support for their calibration or use.

The official GALEX calibration pipeline software--written in a half dozen languages with a sprawling network of complex dependencies--has also never been successfully ported to any system outside of the GALEX internal network at Caltech. Even though the source code for this pipeline will be made publicly available through MAST and perhaps elsewhere, the end of the GALEX project would effectively mark the end of the capability to generate photon level data specifically and revisit the GALEX calibration more generally. A software tool known as gPhoton has been developed by the author with the support of MAST at STScI which reproduces a large portion of the official GALEX pipeline in Python and makes it possible for individual researchers to both generate their own photon level data or integrated images and also adjust or improve upon the calibration.

Additionally, the author and MAST have undertaken to process all of the GALEX data with gPhoton and store the photon level data in a database. Once the database is complete, perhaps in late 2013, it will comprise 180 terabytes and contain approximately 1.5 trillion photon events. The author has created a software tool known as gPhoton to generate photon level data, a tool called gAperture which will helps a user to query this database to create calibrated light curves and a third tool called gMaps to create calibrated image or movie files.

Getting Started

If you are a new user of the GALEX data or have not familiarized yourself with the intricacies of how the GALEX detectors and calibration pipeline work, please read the official Technical Documentation. Pay particular attention to Chapter 3 - Pipeline Overview - Imaging. This will answer a lot of questions that you're sure to have about both the data corpus (e.g. What's the difference between an observation and an eclipse? What's a "dither correction?" How is a "relative response map" different from a flat field? What is a *stim*?)

Installation Instructions

Because the standalone pipeline is written in Python, it is theoretically cross platform. However, the software has only been thoroughly tested on Ubuntu Linux. It has been successfully installed on Mac and Fedora. We're currently unable to do a successful installation on Debian because some of the required libraries aren't yet supported.

Linux

1. With your preferred method, install python2.7, numpy, pyrex, and requests. Here are the recommended commands. If you're using Fedora, substitute "yum" for "apt-get" everywhere.

```
      sudo apt-get install python-setuptools
      sudo apt-get install python-numpy python-scipy
      sudo apt-get install cython
      sudo easy_install pyrex
```

You should use pip to get the latest version of "requests." If requests is already installed, upgrade by appending the --upgrade flag to the following call.

```
      sudo pip install requests
```

You should also use pip to install the latest version of "astropy."

```
      sudo pip install astropy
```

Mac (Warning: harder and not well tested)
1. Install the following libraries using your favorite package management system. We recommend MacPorts with the MacPorts version of Python and the Python libraries. You could also use Fink.

```
python27
py-py27
pytools-py27
pyrex-py27
pyx-py27
setuptools-py27
numpy-py27
wcstools
requests-py27
scipy-py27
cython-py27
```

You may need to install `pip` and `PIL`.

You should also use pip to install the latest version of astropy.

```
      sudo pip install astropy
```

All Operating Systems
2. Download and extract the standalone tools. You will get a folder called *GALEX*. Copy this folder to somewhere convenient to you. (Within this folder are the subfolders cal, e31000, and gPhoton. The cal folder contains calibration reference files. The e31000 folder contains example data for GALEX eclipse 31000. And the gPhoton folder contains the source code for the

standalone tools.)
3. Navigate to the GALEX/gPhoton folder from the command line.
4. Run the example gPhoton commands below. These will either generate errors if a package is missing or they will generate all of the FUV and NUV photons for eclipse 31000. While gPhoton is running, the terminal will update with the photon chunk and current processing rate, mostly just so you know that something is happening. The photons will be dumped into files named [NF]UVPhotons.csv which will be incredibly large (several Gb) so make sure you have disk space. You can kill gPhoton at any time with Ctrl+C. New runs will overwrite the previous csv files.
Protip: The CSV files get updated frequently, so you can generate a "sample" CSV file by letting gPhoton run for a few minutes and then killing it.
Note: If you don't have PYTHONPATH defined, then you will need to put "python" in front of the calls (e.g. `python ./gPhoton.py … … ….`).

Testing Your Build
In the gPhoton folder there is a file called `testscript`. This script will run every example from this User Guide. If there are any errors, it almost certainly indicates that there is a problem with your build. For troubleshooting help during *beta*, please forward the error to *chase.million@gmail.com*.

WARNING: Running this script to completion will take a long time (i.e. hours).

```
./testscript
```

gPhoton
From the command line, after navigating to the GALEX/gPhoton folder, try the following. The first command generates a photon list in FUV and the second (which will take longer) does the same for NUV.

```
./gPhoton.py -r '../e31000/MISWZN11_12494_0315_0002-fd-raw6.fits' -a
'../e31000/MISWZN11_12494_0315_0002-asprta.fits' -c '../cal/' -o
'../e31000/FUVphotons' -s '../e31000/MISWZN11_12494_0315_0002-scst.fits' -d
'../e31000/SSD_fuv_31000.tbl'
```

```
./gPhoton.py -r '../e31000/MISWZN11_12494_0315_0002-nd-raw6.fits' -a
'../e31000/MISWZN11_12494_0315_0002-asprta.fits' -c '../cal/' -o
'../e31000/NUVphotons' -s '../e31000/MISWZN11_12494_0315_0002-scst.fits' -d
'../e31000/SSD_nuv_31000.tbl'
```

Note that gPhoton determines the band from the raw6 filename. If you have a raw6 file with a non-standard filename, you can feed gPhoton the band with the --band flag.

Iff you have a working internet connection, the aspect file (-a) is optional. If none is provided, the aspect correction will be queried from a database at MAST.

Additionally, the Stim Separation Data (SSD) file (-d) is optional. If none is supplied, one will be generated internally to gPhoton. If a SSD filename is provided for a file does not exist, the data will be generated and written to the provided filename. Try the following.

```
./gPhoton.py -r '../e31000/MISWZN11_12494_0315_0002-fd-raw6.fits' -c '../cal/'
-o '../e31000/FUVphotons' -s '../e31000/MISWZN11_12494_0315_0002-scst.fits'
```

Detector events that cannot be aspect corrected either because they fall in a time range for which no aspect solution is available or because they are off the detector entirely (e.g. stims) appear in the CSV output file with NULL entries for RA and Dec. Such data are called "NULL" data. If you wish to output such data to a separate file, you may do so with the -u flag. This will create a second CSV file with "_NULL.csv" appended to the base filename.

```
./gPhoton.py -r '../e31000/MISWZN11_12494_0315_0002-fd-raw6.fits' -a
'../e31000/MISWZN11_12494_0315_0002-asprta.fits' -c '../cal/' -o
'../e31000/FUVphotons' -s '../e31000/MISWZN11_12494_0315_0002-scst.fits' -u
```

For multi-visit eclipses, you can specify more than one aspect file for a single raw6 file. The syntax for the -a keyword in this case is as follows.

```
-a '../FOO_sv01-asprta.fits,../FOO_sv02-asprta.fits,../FOO_sv03-asprta.fits'
```

You can always get more information about the gPhoton command line options by using `--help`.

```
./gPhoton.py --help
```

If you want to dive into the source code that gPhoton uses to generate photon lists, the primary utility is contained in PhotonPipe.py. You can also run PhotonPipe from the Python interactive dialog.

Convert GALEX Time to UNIX Time

Time stamps for gPhoton and all of the photon tools use "GALEX Time" which is a simple offset from UNIX Time. The conversion fact, in seconds, is:

(GALEX Time) + (315964800) = (UNIX Time)

Photon File Column Definitions

1. t - time of the event (GALEX time)
2. x - detector x position
3. y - detector y position
4. xa

5. ya
6. q
7. xi - calibrated detector position
8. eta - calibrated detector position
9. ra - right ascension of the event
10. dec - declination of the event
11. flags - status information, see below

Photon File Flag Column Definitions

1. successful calibration
2. N/A
3. Skipped
4. N/A
5. N/A
6. Stim
7. Masked (by hotspot mask)
8. BadAsp (unknown aspect solution)
9. Range (off of detector prior to the wiggle correction)
10. BadWalk (off of detector prior to the walk correction)
11. BadLin (off of detector prior to the linearity correction)
12. BadDist (off of detector prior to the stim distortion correction)
13. AspJump (occurred during a jump or gap in the aspect solution or is bracketed by one or more flagged aspect values)
14. N/A
15. N/A
16. N/A

Database Tools

The database tools are the command line programs that provide basic functionality for interacting with the photon database at MAST to produce scientifically useful data products.

gFind Data Location Tool

Basic Usage

The data location tool called gFind will report all time ranges for which data exists at a specific location on the sky. These ranges are computed by finding aspect positions within a detector radius of the desired target and using their associated time stamps to find contiguous ranges and then comparing that against data that has actually been loaded into the photon database at MAST. Attempt the following example command.[1]

```
./gFind.py -b 'NUV' -r 176.919525856024 -d 0.255696872807351
```

Tweakable Search Parameters

The default allowable gap between two time stamps for them to be considered contiguous is 1 second because this is the default spacing between adjacent aspect solutions. This parameter is adjustable, however, with the `-g` or `--gap` parameter. To consider data with gaps of 100 seconds to be contiguous--a reasonable to collect data within each eclipse--try the following command.

```
./gFind.py -b 'NUV' -r 176.919525856024 -d 0.255696872807351 --gap 100
```

If you only want contiguous exposure ranges longer than some number, you can use the --minexp parameter.

```
./gFind.py -b 'NUV' -r 176.919525856024 -d 0.255696872807351 --minexp 100
```

The --gap and --minexp parameters can be used in conjunction.

```
./gFind.py -b 'NUV' -r 176.919525856024 -d 0.255696872807351 --gap 100
--minexp 100
```

If you do not want to include time ranges where the source only appears near the edge of the detector, you can adjust the `--detsize` parameter (defaults to 1.25 degrees, detector diameter). This effectively excludes exposure time ranges in which the source is outside of *detsize*/2 from the center of the detector. (Note: Time ranges may be included if `--gap` is set and the source passes through the more narrowly defined usable detector, e.g. sources right on the edge or as may happen frequently with scan mode data.)

[1] Throughout the User Guide, we will use the M dwarf flare star GJ 3685A as the example target. The GALEX observation of this flare was described in Robinson, et al. "GALEX observations of an energetic ultraviolet flare on the dM4e star GJ 3685A." *The Astrophysical Journal* 633.1 (2005): 447.

```
./gFind.py -b 'NUV' -r 176.919525856024 -d 0.255696872807351 --detsize 1
```

Alternate I/O Formats
As with gAperture and gMap, you can use the alternate formulation of the sky position.

```
./gFind.py -b 'NUV' --skypos '[176.919525856024,0.255696872807351]'
```

For easy copy/paste into gAperture and gMap command lines, try the alternate output format using the --alt flag switch.

```
./gFind.py -b 'NUV' --skypos '[176.919525856024,0.255696872807351]' --alt
```

And if you are only interested in how much exposure is available but not the particular time ranges, pass the --exponly flag.

```
./gFind.py -b 'NUV' --skypos '[176.919525856024,0.255696872807351]' --exponly
```

gAperture Photometry Tool

Basic Usage

The photometry tool called gAperture computes light curves by making web queries to the MAST photon database. If requested, the light curve data is written to a CSV file. You must specify band, RA, dec, and a time range. Optionally, you can provide the inner and outer radius of an annulus from which a background measurement will be taken. The following call simply write the background subtracted flux and related values of the specified object to lightcurve.csv.

```
./gAperture.py -b 'NUV' -r 176.919525856024 -d 0.255696872807351 -a 0.03 -i
0.03 -o 0.04 --t0 766525332.995 --t1 766526576.995 -f 'lightcurve.csv'
```

If no CSV filename is specified, the AB Mag values will simply be printed to the command line.

```
./gAperture.py -b 'NUV' -r 176.919525856024 -d 0.255696872807351 -a 0.03 -i
0.03 -o 0.04 --t0 766525332.995 --t1 766526576.995
```

If you wish to generate a light curve, you can specify the step size with the -s flag. To generate a light curve of the specified object with 100 second increments and write the result to lightcurve.csv, try the following.

```
./gAperture.py -b 'NUV' -r 176.919525856024 -d 0.255696872807351 -a 0.03 -i
0.03 -o 0.04 --t0 766525332.995 --t1 766526576.995 -f 'lightcurve.csv' -s 100.
```

Relative Response

If you're willing to wait slightly longer, you may apply an estimated relative response to the calculated flux with the `--response` flag. Rather than building a relative response map, this computes the mean relative response within the aperture over the specified time range. Note that the reliability of the relative response is questionable below a few hundred seconds of exposure.

```
./gAperture.py -b 'NUV' -r 176.919525856024 -d 0.255696872807351 -a 0.03 -i
0.03 -o 0.04 --t0 766525332.995 --t1 766526576.995 -f 'lightcurve.csv' -s 100.
--response
```

<u>High Quality Background</u>
The default background correction performed by gAperture simply measures the count rate per area within the specified background annulus, scales it to the area of the photometry aperture, and substracts it from the count rate inside the aperture. This method is extremely susceptible to stray light from nearby stars. An improved background estimate--which excludes light from nearby stars as identified by the MCAT database--can be requested with the `--hrbg` flag.

```
./gAperture.py -b 'NUV' -r 176.919525856024 -d 0.255696872807351 -a 0.03 -i
0.03 -o 0.04 --t0 766525332.995 --t1 766526576.995 -f 'lightcurve.csv' -s 100.
--hrbg
```

<u>Recommended Usage</u>
The highest quality measurement is made by using both the response and the high quality background measurement. As a shortcut, you can initiate both of these flags simultaneously with `--best`.

```
./gAperture.py -b 'NUV' -r 176.919525856024 -d 0.255696872807351 -a 0.03 -i
0.03 -o 0.04 --t0 766525332.995 --t1 766526576.995 -f 'lightcurve.csv' -s 100.
--best
```

<u>Creating a JPEG Stamp</u>
If you wish to create a JPEG preview image of your target with a diameter equal to the diameter of either your aperture or the outer boundary of our annulus if one is specified, then you can pass a filename to the --stamp flag. To write a preview image to lightcurve.jpeg, try the following.

```
./gAperture.py -b 'NUV' -r 176.919525856024 -d 0.255696872807351 -a 0.03 -i
0.03 -o 0.04 --t0 766525332.995 --t1 766526576.995 -f 'lightcurve.csv' --stamp
'lightcurve.jpeg'
```

<u>Alternate Input Formats</u>
For your convenience, you can use an alternative list-style input for the sky position with the format and the time range. Specifying the time range in this way allows you to request multiple non-contiguous time ranges. The flags are --skypos and --tranges and the format is as follows

```
./gAperture.py -b 'NUV' --skypos '[176.919525856024,0.255696872807351]' -a
```

```
0.03 -i 0.03 -o 0.04 --tranges '[[766525332.995, 766526576.995],
[919755500.995, 919755916.995]]'
```

If no time ranges are specified, gAperture will compute the requested flux values over *all available data*.

```
./gAperture.py -b 'NUV' --skypos '[176.919525856024,0.255696872807351]' -a
0.03 -i 0.03 -o 0.04
```

Coadding

As a default, gAperture will treat separate time ranges as distinct and will not compute fluxes across gaps. However, if you want to find the coadded flux value across all specified time ranges, use the --coadd flag. As with gFind, you can specify how long of a `--gap` signals the end of one exposure and start of the next; the default is 1 second.

```
./gAperture.py -b 'NUV' --skypos '[176.919525856024,0.255696872807351]' -a
0.03 -i 0.03 -o 0.04 --coadd
```

Verbosity and Headers

You can always choose to see additional output with the -v or --verbose flag. Setting this to 1 will print some extra output and setting it to 2 will print much more output for debugging purposes. If you would like to add a copy of the command line to the header of the CSV file behind a # delimiter, use the --addhdr switch.

gAperture CSV File Column Defintions

1. t0 - slice start time
2. t1 - slice end time
3. radius - aperture radius in degrees
4. exptime - effective exposure time in seconds (which will be less than t1-t0 because it includes the dead time and shutter corrections)
5. cps - counts per second within the aperture (-a) less the background cps from the annulus (-i and -o) adjusted for the area of the aperture, with the response applied if requested (--response)
6. error - Poisson error of the cps measurement (i.e. sqrt(counts))
7. flux - calculated flux based on cps (erg sec^-1 cm^-2 A^-1)
8. flux_error - based on cps error
9. mag - AB magnitude based on cps
10. mag error - based upon the cps error
11. inner annulus - background annulus inner radius in degrees (if not requested == ‘False’)
12. outer annulus - background annulus outer radius in degrees (if not requested == ‘False’)
13. background - estimated background within the radius (if not requested == 0), determined by computing the count rate within the background annulus and scaling it to the aperture area.

14. response - estimated mean response within the radius (if not requested == 1)
15. counts - raw total number of counts within the aperture radius (-a or (3))
16. apcorrect1(radius,band) - aperture correction in delta AB magnitude based on Figure 4 from Morissey, et al., 2007. This is the more trustworthy generic aperture correction. Subtract this value from the mag (9) to get the aperture corrected magnitude.
17. apcorrect2(radius,band) - aperture correction in delta AB magnitude based on Table 1 from www.galex.caltech.edu/research/techdoch-ch5.html

gMap Image Creation Tool

The image creation tool called gMap will generate FITS images of arbitrary size, shape and depth. Images can be equivalent to GALEX count, intensity, or response maps. You can make movie versions of each of these map types as well.

Basic Usage

You can create a count image from the command line using the following command.

```
./gMap.py -b 'FUV' -r 176.919525856024 -d 0.255696872807351 --angle 0.5 --t0
766525332.995 --t1 766526576.995 --count 'count.fits'
```

To create a movie file, specify the --frame keyword by passing the number of seconds over which each frame of the movie should be integrated. Setting --framesz to zero is equivalent to creating a count map integrated over the entire requested.

Try the following to create a movie file of count images with 100 second frames.

```
./gMap.py -b 'FUV' -r 176.919525856024 -d 0.255696872807351 --angle 0.5 --t0
766525332.995 --t1 766526576.995 --count 'count.fits' --frame 100
```

As with gAperture, you can use the alternate formats for both sky position and time range(s).

```
./gMap.py -b 'FUV' --skypos '[176.919525856024,0.255696872807351]' --angle 0.5
--tranges '[[766525332.995, 766526576.995], [919755500.995, 919755916.995]]'
--count 'count.fits'
```

You can also specify --coadd to generate same.

```
./gMap.py -b 'FUV' --skypos '[176.919525856024,0.255696872807351]' --angle 0.5
--tranges '[[766525332.995, 766526576.995], [919755500.995, 919755916.995]]'
--count 'count.fits' --coadd
```

And if you do not specify any time range, gMap will automatically use all available exposure.

```
./gMap.py -b 'FUV' --skypos '[176.919525856024,0.255696872807351]' --angle 0.5
```

```
--count 'count.fits' --coadd
```

As with gFind, you can specify how long of a `--gap` signals the end of one exposure and start of the next; the default is 1 second.

If you wish to generate an intensity map (i.e. a response corrected image), you can pass a filename to the --intensity flag. Be warned that this will add considerably to the runtime because of the high computational cost of the interpolations required to create the relative response map. Hopefully this will be faster in the future.

```
./gMap.py -b 'FUV' --skypos '[176.919525856024,0.255696872807351]' --angle 0.5
--count 'count.fits' --intensity 'intensity.fits' --coadd
```

If you would additionally like to write a relative response map to disk, you can pass a filename to the --response flag.

```
./gMap.py -b 'FUV' --skypos '[176.919525856024,0.255696872807351]' --angle 0.5
--count 'count.fits' --intensity 'intensity.fits' --response 'response.fits'
--coadd
```

Image Headers

FITS images created by gMap have headers which describe the World Coordinate System (WCS) information that describes them and also has the effective exposure time information for the observation as a whole. For multi-frame images (i.e. movies) the exposure time information is contained in a table in the FITS secondary HDU. This table describes the start time, stop time, and effective exposure time for each frame with appropriately labeled columns.

Common Questions, Issues, and Gotchas

"My data is not available."

You can verify that data for your desired target does or does not exist in the current version of the database by using gFind.

The photon database at MAST is still being populated, and only a small fraction of the GALEX data has been uploaded so far. If you have a particular object of interest and know that it was observed by GALEX, please contact Chase Million (chase.million@gmail.com) or Bernie Shiao (shiao@stsci.edu) and we will try to prioritize importation of the associated data.

Exposure Time

When calculating the exposure time of a GALEX observation, one cannot simply subtract the end time from the start time. This *raw exposure time* (t_raw) must be adjusted into an *effective exposure time* (t_eff) that accounts for several peculiarities of GALEX.

Shutter Correction

GALEX did not observe all parts of the sky at all times. Even when GALEX was observing a particular part of the sky, there are times when there is no data or no usuable data. Data might exist but be unusable for a large variety of reasons including, mostly commonly, that the spacecraft was not observing or observing the wrong part of the sky, a valid aspect solution could not be reconstructed from the available data, or there is a temporary gap in the data due to a spacecraft or data transmission anomaly. The *shutter correction* accounts for these gaps by accounting for the time that a "virtual shutter" was closed. It is computed by integrating the 0.05 second gaps in all data covering a particular source and within a particular time range. The shutter correction is then subtracted from the raw exposure time.

Deadtime Correction

The GALEX micro-channel plates can only read a single event at a time; that is, when one phone event is being read into the electronics, any other incident photon events are missed. This effectively reduces the exposure time of any observation by a ratio known as the "deadtime" correction. The deadtime correction is the estimated fraction of events that are not detected due to detector readout. The deadtime changes from observation to observation and even moment to moment but generally scales as a function of field brightness or global count rate. There are two ways to estimate the deadtime.

Method 1 - Direct Measurement

The GALEX detectors have four locations in which electronic pulses are superimposed on the detector in parallel with real photon events. These pulses, known as "stims," have a known absolute count rate of 79 counts per second. Dividing the measured count rate by the known absolute count rate provides a direct measure of the deadtime ratio. This is the method employed by the original GALEX pipeline at Caltech, and it works well when estimating the dead time over long exposures. However, the variance in the stim rate due to simple counting statistics over short exposure times (one to hundreds of seconds) is so high that it produces inconsistent and essentially unusable effective exposure times.

```
deadtime_1 = (average measured stim count rate) / 79.0
```

Method 2 - Empirical Formula
An empirical formula for the deadtime correction as a function of global count rate was derived from flight data by Patrick Morrissey and Tim Conrow [personal communication]. The formula is the same for both bands with a combined offset of zero, though in practice there may be a few percent difference. The formula is

```
deadtime = (tec2fdead) * (global count rate) / feeclkratio
```

where tec2fdead = 5.52e-6 and feeclkratio = 0.966. This method produces very consistent results across all time ranges, so it is the method used by the photon tools.

Effective Exposure Time
The effective exposure time is computed by subtracting the shutter correction from the raw exposure and scaling the result by the deadtime.

```
t_eff = (t_raw - shutter) * deadtime
```

Relative Response

Unlike a CCD, the GALEX detectors do not have true “pixels” and the telescope boresight does not remain stationary on the sky (i.e. it dithers). So to appropriately apply a flat field correction to an *un-dithered* image, you must create a map that represents the detector flat in fixed *sky coordinates*. To do this, the GALEX pipeline “stamps” the flat onto a *response map* at 1 second intervals, centered and rotated correctly to the spacecraft boresite pointing at the time, and scales each stamp by both the effective exposure time over that second and a global scale factor that accounts for changing detector sensitivity over the mission. You can then simply divide the integrated *count map* by the *response map* to get the calibrated, flat adjusted *intensity map* from which astronomical measurements can be made.

The response maps produced by gMap are created in the way described above. However, the interpolations required to reposition the flat on the relative response map are, at this time, very slow. Instead of creating a response map, gAperture instead identifies the the location on the flat (or detector) where the targeted source would fall and extracts the average response over the requested aperture. This is done for several reasons including that gAperture does not create count images to compute light curves, so you could not properly divide a response image from a count image, response creation is very slow, and there is typically little variation in the flat field over the diameter of a typical source extraction aperture.

Gain Sag

Coming soon.