Using Pype-It for LBT MODS reduction

Zach Stevens - last updated 7/24/23

Contrasts between Katherine and Zach's work

A Note:

Link to Katherine's Work: https://docs.google.com/document/d/1etrvsHlItrwcd9sO7ob7E9FN7bu9ZB4p3hf7nAoSSac/edit?usp=sharing

Katherine has done a wonderful job of leaving behind a set of notes that has walked me through my work and will likely be immensely helpful to whoever is reading this in the future. You will already likely see similarities between how I will be laying out my notes and how she has done hers - she did a great job so I will be imitating it in many ways. However, I have changed some of the naming conventions of some things and wanted to give clarification on things that I ran into issues with. Also, I work in a Windows system, whereas Katherine works in a Linux environment. As a person without much prior coding experience, I'll outline the things I discovered that helped me in case the person following me is in a similar boat. If you know what you're doing with code, congrats! You're a fair few steps ahead of me already :)

I tried to follow the naming conventions of each section as documented by Katherine, and as such they should be able to be followed in parallel, with mine mostly as clarification that I found notable. If a section is omitted, it is because I found that Katherine laid it out well, and didn't need to add clarification.

Notes on PypeIt Installation & Katherine's .fits Manipulation Programs:

Installation and Setup:

To run most of this code, I have been using the Anaconda Navigator for Python. This can be found at https://www.anaconda.com/products/distribution. I am then using the Anaconda prompt for file movement.

When running Python commands, I found that I struggled to get basic commands to run, as I did not have the correct path set up. This can be done by searching within your PC for Advanced System Settings. Click on the *Environment Variables* button, and click *New.* You can then input the Path where you want your code to run out of.

For me, for example, this looked like “C:\Users\13zac\Python\Python39\Scripts\”.

You should, similarly, add a path as well to the PypeIt commands. This path for me looked like “C:\Users\13zach\Anaconda3\envs\pypeit\Scripts”.

This will allow you to run code much smoother instead of having to put the entire path to the code before running it each time, and will function easier as shown in Katherine’s documentation.

File Organization and Data:

As suggested by Katherine, my system of reduction thus far has been to download and reduce files for each object individually before moving onto the next. I wanted to give clarification here particularly since I have used different naming conventions, and when I was downloading the files on my own I was unsure as to which files to download at first. Hopefully this should give a good guide as to what images specifically to download and where to find them.

Go to archive.lbto.org/, and find the night where your object was observed. Going into the MODS link you can find the list of each mods filename (far right) and what they are. The list of desired images are listed below.

Data Needed

For reductions, the files you need will be:

- **Object images:** Titled something with UV at the beginning in ProjectID, and as OBJECT in the Object column

- **Standard star calibration images:** ID'd as CALIBRATION and in the Object column tend to have 'dual grating' in the title
- **Slitless flat frames:** ID: Calibration, Object: VFLATX.X Clear/UG5 Dual Slitless Pix
    - A note on this - for the slitless flats for mods1/2**b**, there will be a UG5 filter, while for mods1/2**r** there will not be. As such, you will end up with 10 slitless flats for the blue side, while for red you will only have 5.
- **Slit flat frames:** ID: Calibration, Object
- **Lamps:** This is a set of three images including Ne + Hg[Ar] Lamps, Xe+Kr Lamps, and Ar Lamp.
- **Bias frames:** This is usually 5 images labeled as Bias 8Kx3K.

The unzipping of downloaded .gz files can be done using a program such as 7-zip, which can be found at https://www.7-zip.org/download.html.

Data Pre-processing

For bias, object, and standard folders, the default combine.py code should be run on all objects (just median combining). This process is well documented in Katherine's notes.
Within the lamps folder, use the mean combine function, adding the -m to the code as shown by Katherine.

In the slit_flat and slitless_flats folders, the different filters should be median combined (the ND1.5 all median combined and the UG5 all median combined for the slit flats, or for the slitless flats, median combine the 5.0 and then separately median combine the 10.0). Then, the master median combined image of each filter should be combined using the mean average function.

Within each directory that has been made inside each of the mods files, I have added the combine.py file so that I can run it within that directory. I've found it to make combinations a little easier.

While Katherine has created an add.py program, I did not use it at any point during reductions, and instead opted to use the median combination.

Running PypeIt:

Setup:

If you need to manually identify an object due to it being too dim, when you run `pypeit_setup,` add the parameter **-m**, and in your .**pypeit** file, a column titled **manual** will appear. The following link to the documentation describes how to modify your .pypeit file to include a manual extraction. https://pypeit.readthedocs.io/en/release/manual.html

Modify the .pypeit file:

Parameter block:

While Katherine's notes work well and are accurate through MODS1B, MODS1R, and MODS2B, they fail for MOSD2R. PypeIt seems to identify an extra slit within the 2R images, and runs into errors with it. As such, we can make it so PypeIt only runs on the center slit, greatly reducing runtime and also making it so PypeIt works with 2R! **This modification should be applied to all of the MODS images**, as it will cut the PypeIt runtime down to about a third of what it is without.

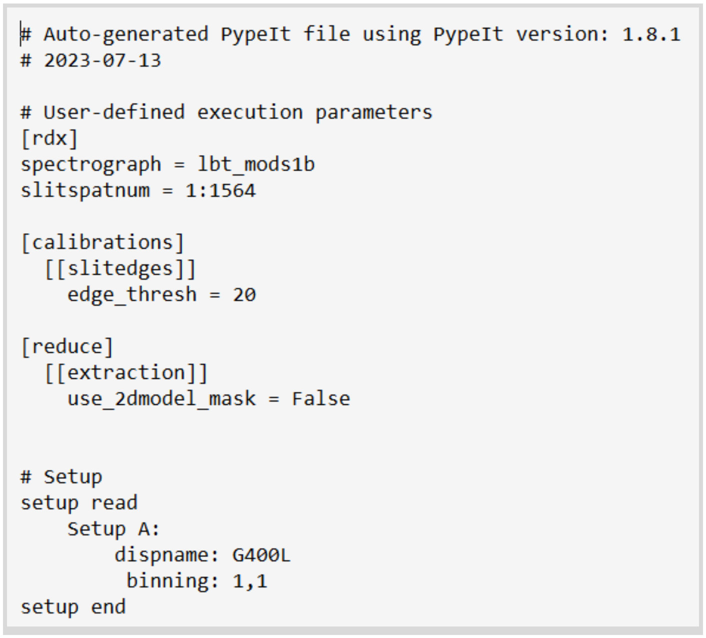
Add under the rdx, along with what Katherine put: `slitspatnum = 1:1564`

As such, your final block that you will add under the parameter section will look like:

```
slitspatnum = 1:1564

[calibrations]
  [[slitedges]]
    edge_thresh = 20

[reduce]
  [[extraction]]
    use_2dmodel_mask = False
```

You may also need the following line, if your object is not detected. You can inspect the QA folder after reduction has finished, and if there is no object detected, but an evident bump where the object is in the slit, you can reduce the snr threshold to capture the object.

```
[reduce]
  [[findobj]]
    sig_thresh = <number>
```



```
# Auto-generated PypeIt file using PypeIt version: 1.8.1
# 2023-07-13

# User-defined execution parameters
[rdx]
spectrograph = lbt_mods1b
slitspatnum = 1:1564

[calibrations]
  [[slitedges]]
    edge_thresh = 20

[reduce]
  [[extraction]]
    use_2dmodel_mask = False


# Setup
setup read
    Setup A:
        dispname: G400L
         binning: 1,1
setup end
```

Data block:

Important clarification here because of naming conventions and how many images of each type we have - I have had to change the frametype a lot, but commonly what I've found is that after running setup, the PypeIt file usually misses the identification of my flats and my standard. Here is the original (or what I've found it most likely does) and the final after I've made changes. The order of the files does not seem to matter.

Original:

```
# Read in the data
data read
 path RAWDIR
|                 filename |                       frametype |
|      master_standard.fits |                            None |
| masterslitless_flat.fits |                            None |
|          masterlamps.fits |                        arc,tilt |
|           masterbias.fits |                            bias |
|      masterslit_flat.fits | pixelflat,illumflat,trace |
|         masterobject.fits |                         science |
data end
```

After Changes:

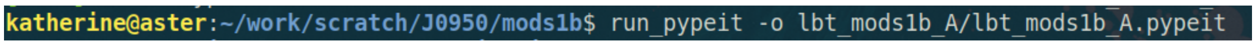
```
# Read in the data
data read
 path RAWDIR
|                 filename |                 frametype |
| masterslitless_flat.fits |       pixelflat,illumflat |
|       masterstandard.fits |                  standard |
|          masterlamps.fits |                  arc,tilt |
|           masterbias.fits |                      bias |
|      masterslit_flat.fits |                     trace |
|         masterobject.fits |                   science |
data end
```

Run Pipeline:

It's now time to run PypeIt! Do this from the main folder. Thankfully, this is a straightforward task. Execute:

`run_pypeit -o <pypeit reduction file>`

The -o argument overwrites any previous Science output files.



```
katherine@aster:~/work/scratch/J0950/mods1b$ run_pypeit -o lbt_mods1b_A/lbt_mods1b_A.pypeit
```

It will take anywhere from 20 min to an hour to run. The PypeIt pipeline will bias-correct the images, flat field them, correct the tilt, wavelength calibrate them, find objects in the slit(s), and perform sky subtraction. It also incorporates bad-pixel masks specific to each spectrograph.

Fluxing:

The fluxing part of the process is what I ran into the most issues with. Unfortunately, the pixel scale of the standard stars within the PypeIt database is much higher than the scale of the standard stars imaged by the LBT. As a result, we got a sensitivity function that was inaccurate and produced various bumps within our final spectra that were illegitimate. There were also some small issues between the chip boundaries, where the sensitivity was slightly different. There were a few fixes we found for this process, and they are documented below.

Mods Blue:

Creating/modifying the sens_file.txt

The first step of the fluxing process for the blue end is slightly different than Katherine documents, and is actually more similar to her documentation of the red, so head down to that section of her document. We do not need to crop the blue, so ignore that part. Instead, you will first create a sens_file, as Katherine explains. Make a simple .txt file, naming it sens_file.txt. However, what you will put in this file is different. This file will tell PypeIt what parameters to use while it is running, and the necessary parameters that I have found work best are listed below. Simply copy and paste them into your sens_file.txt.

```
[sensfunc]
  algorithm = UVIS
  [[UVIS]]
     nresln = 5
     polycorrect = False
     resolution = 1500
     balm_mask_wid = 5.0
```

If you are interested in what these parameters do, take a look at their documentation here: https://pypeit.readthedocs.io/en/release/pypeit_par.html. I would suggest saving a master copy of this file in a separate folder, then copying it into the Science folder each time you are fluxing a new object.

Once this file is saved, execute `pypeit_sensfunc` with the following arguments:

```
pypeit_sensfunc -s sens_file.txt -o sens.fits <standard star spec1d
ending in .fits>
```

You will then run `pypeit_flux_setup .`
From there, make the same edits as Katherine specifies to the .flux file. I'll add these below for continuity.

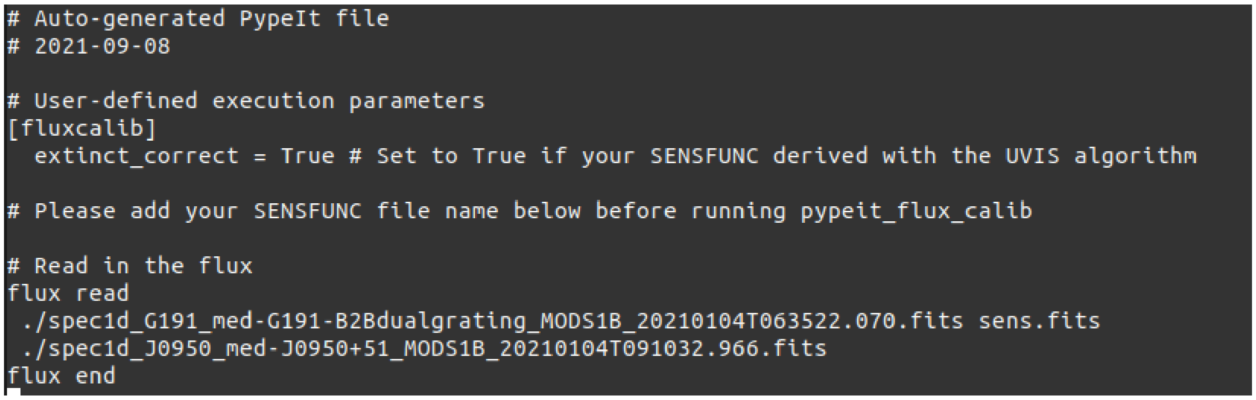
"Make the following changes to the flux file:

- `extinct_correct = True` (for UVIS algorithm only). This corrects for atmospheric extinction
- After the first file is listed under flux read, add a space and the filename for the `sens.fits` file created in step 1. Like so:

```
flux read
   spec1dfile1 sensfile
   spec1dfile2
      ...
      ...
flux end
```

-

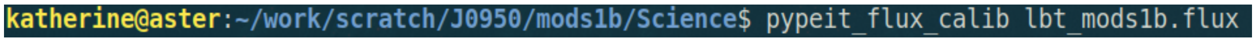
With all changes made, you .flux file should look something like this:



```
# Auto-generated PypeIt file
# 2021-09-08

# User-defined execution parameters
[fluxcalib]
  extinct_correct = True # Set to True if your SENSFUNC derived with the UVIS algorithm

# Please add your SENSFUNC file name below before running pypeit_flux_calib

# Read in the flux
flux read
 ./spec1d_G191_med-G191-B2Bdualgrating_MODS1B_20210104T063522.070.fits sens.fits
 ./spec1d_J0950_med-J0950+51_MODS1B_20210104T091032.966.fits
flux end
```

Save the file and continue.

1. The final step is to run the `pypeit_flux_calib script.` It is a simple step, execute the following in your terminal:
   `pypeit_flux_calib <name of the .flux file>`



```
katherine@aster:~/work/scratch/J0950/mods1b/Science$ pypeit_flux_calib lbt_mods1b.flux
```

This will flux calibrate your 1D spectra! It replaces the original spectral files with flux calibrated ones. I suggest plotting the spectra to make sure everything looks OK. You may want to use crop_spec.py to remove the ends of the object spectrum, as they will be very noisey. You've finished reducing your Mods Blue spectra."

Unfinished note: atmospheric correction does not account for molecular…

Absolute Flux Calibration - needs updating

An Introduction:

The next part of this work is not documented in Katherine’s work, and has been done independently by me. A brief summary of the steps is listed below.

1. Download acquisition images and reduce them.
2. Compare the acquisition images to the SDSS images.
3. Plot flux vs CCD counts and find new flux of the quasar.

Download images:

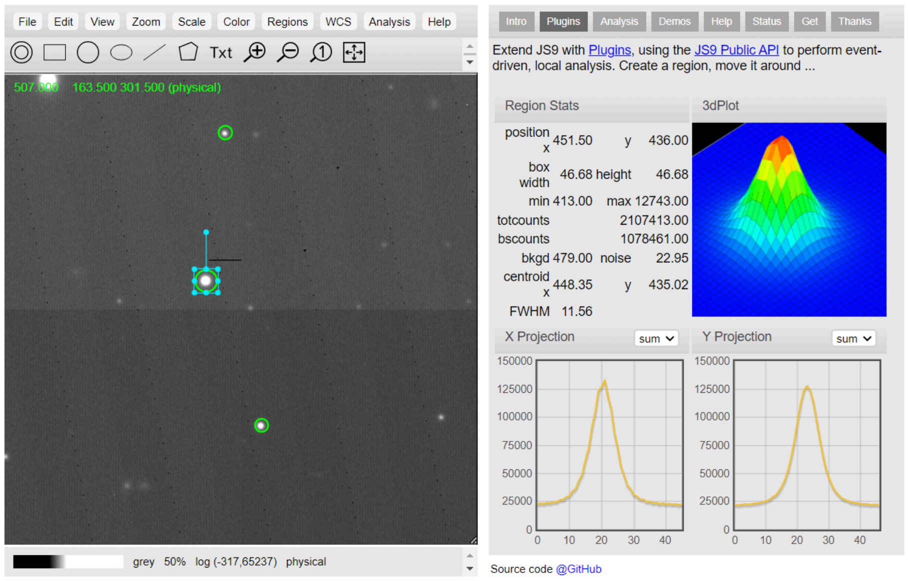
You will want to download the acquisition images for the current object you are working on. These can be found as well at http://archive.lbto.org/ and will be listed as Acquisition. These acquisition images show our object with other stars nearby in the same frame. There is usually a set of 3 - I have found that the second image tends to be the one that we want to use. It has the object in the center, with the surrounding stars shown in the field.

Using JS9 and SDSS data to analyze images:

We will now be importing our images into JS9, the link to which is https://js9.si.edu/. Import the acquisition image into the page by going to *file*, and click *open local*, where you can then input your master acquisition image.

You will also want to open the data release 16 (DR16) from SDSS, found at https://skyserver.sdss.org/dr16/en/tools/chart/navi.aspx. Input the RA and dec of your object into the image finder, and zoom until you find the same field as your acquisition image. You may also need to rotate your image in JS9 by going to *Zoom* and rotating as needed. Once you have the same field in the DR16 finder as your acquisition image, you are ready to begin comparison. Find objects that are particularly bright in your acquisition image, and find them in DR16. If they are categorized as a **STAR** in the top right corner, this is a good object to note.

You will now begin to find the counts in each star around your quasar. Go to *Regions* at the top of JS9 and select **Circle**. Place the circular region around each star that you plan on using, keeping the region tightly around the star without losing too much light (the scale can be changed by left clicking and dragging). On the right side of JS9, go into the *Plugins* tab, and find the **bscounts** line in Region Stats. This is the number of *background subtracted counts* within the region you have selected. If you have multiple regions on your screen, it shows the **bscounts** for whichever region you have selected at the moment. Make note of this number for each region. Your screen should look something like this.


File Edit View Zoom Scale Color Regions WCS Analysis Help
Txt
Intro Plugins Analysis Demos Help Status Get Thanks
3dPlot
X Projection sum
Y Projection sum
Source code @GitHub

507.000 163.500 301.500 (physical)

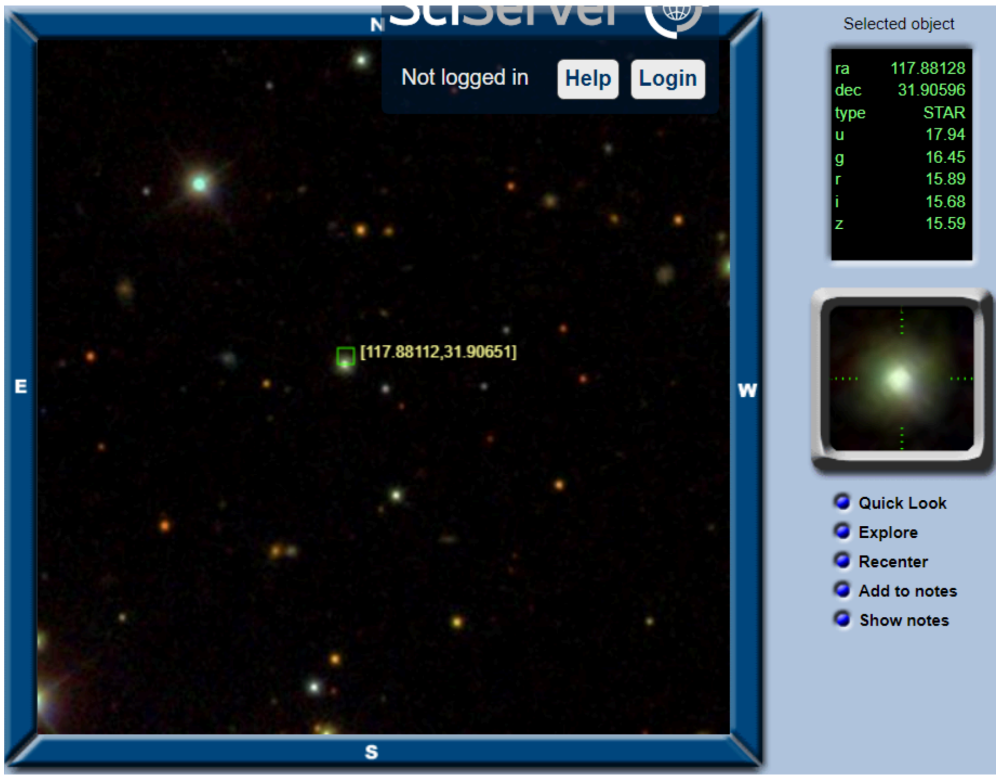
grey 50% log (-317,65237) physical

Extend JS9 with Plugins, using the JS9 Public API to perform event-driven, local analysis. Create a region, move it around ...

Region Stats

position x	451.50	y	436.00
box width	46.68	height	46.68
min	413.00	max	12743.00
totcounts			2107413.00
bscounts			1078461.00
bkgd	479.00	noise	22.95
centroid x	448.35	y	435.02
FWHM	11.56		

You will also put a region around your quasar, and make note of the **bscounts** there. From here, it will be important to have an Excel page open, or something that can do similar work. You will now want to find those same stars (not your quasar!) in the DR16 database. Click on them, and find their magnitude in the r-band. This number is found in the top right corner, where the **r** is.



Selected object

ra	117.88128
dec	31.90596
type	STAR
u	17.94
g	16.45
r	15.89
i	15.68
z	15.59

By now, you will have a direct conversion from CCD counts to magnitude. You will likely also want to find your flux. Going from magnitude to flux is relatively easy. By using the following two equations, we end up with a quick conversion between the two. The first equation gives flux in units of “nanomaggies,” and the second equation converts the flux in nanomaggies to ergs/s/cm^2/Å. Our m is magnitude, and the lambda used for the red filter is 6200Å.

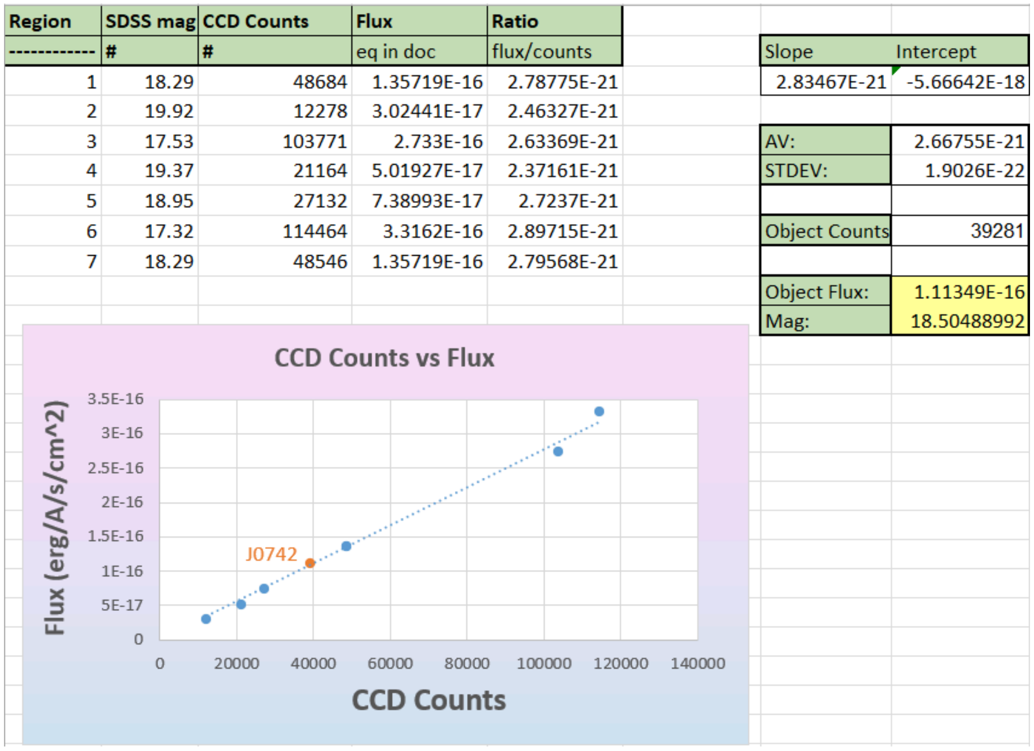
$$f_{nmg} = 10^9 \times 10^{-m/2.5}$$

$$f_\lambda = f_{nmg} \times \frac{1.08\times10^{-13}}{\lambda_A^2} \quad W/m^2/\text{Å}$$

We arrive at a final equation as follows, where m is our magnitude.

$$f_\lambda = 2.8096 \times 10^{-9} \times 10^{-m/2.5}$$

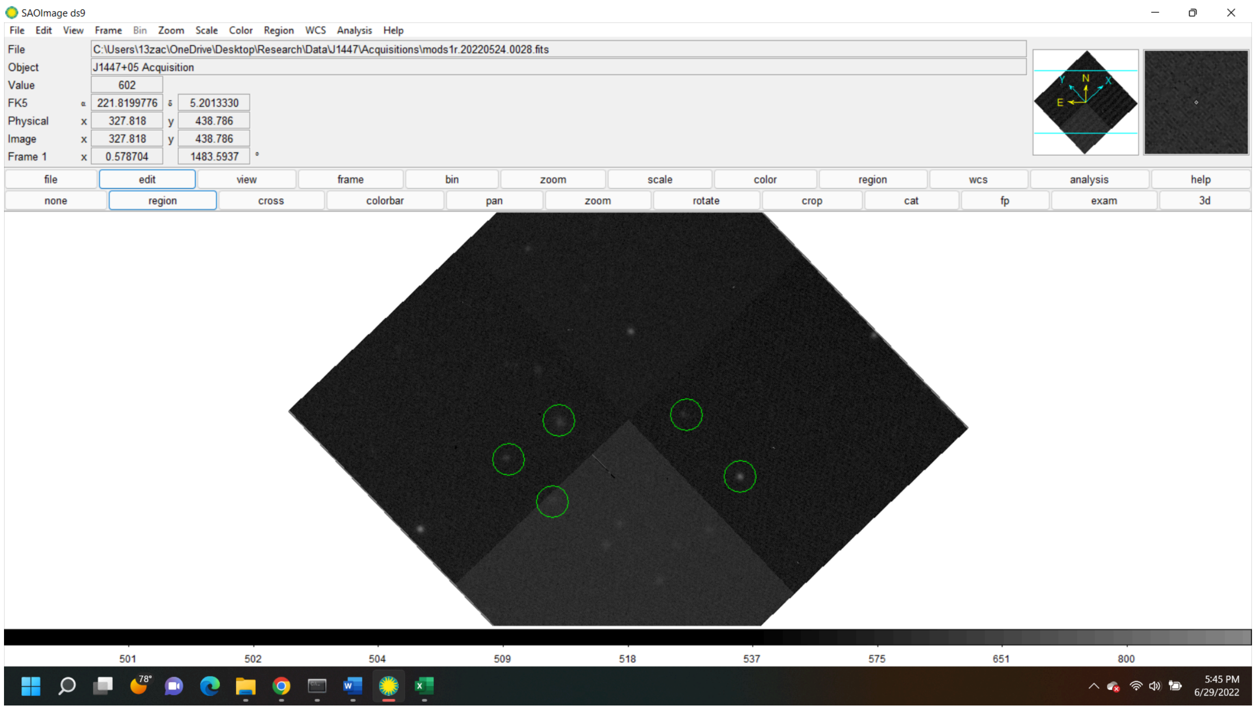
This equation can be input into the cells of Excel to automatically convert your magnitude values into flux. You will then plot your flux versus your CCD counts, take the slope, and multiply your object counts by this value to get the flux of your object!



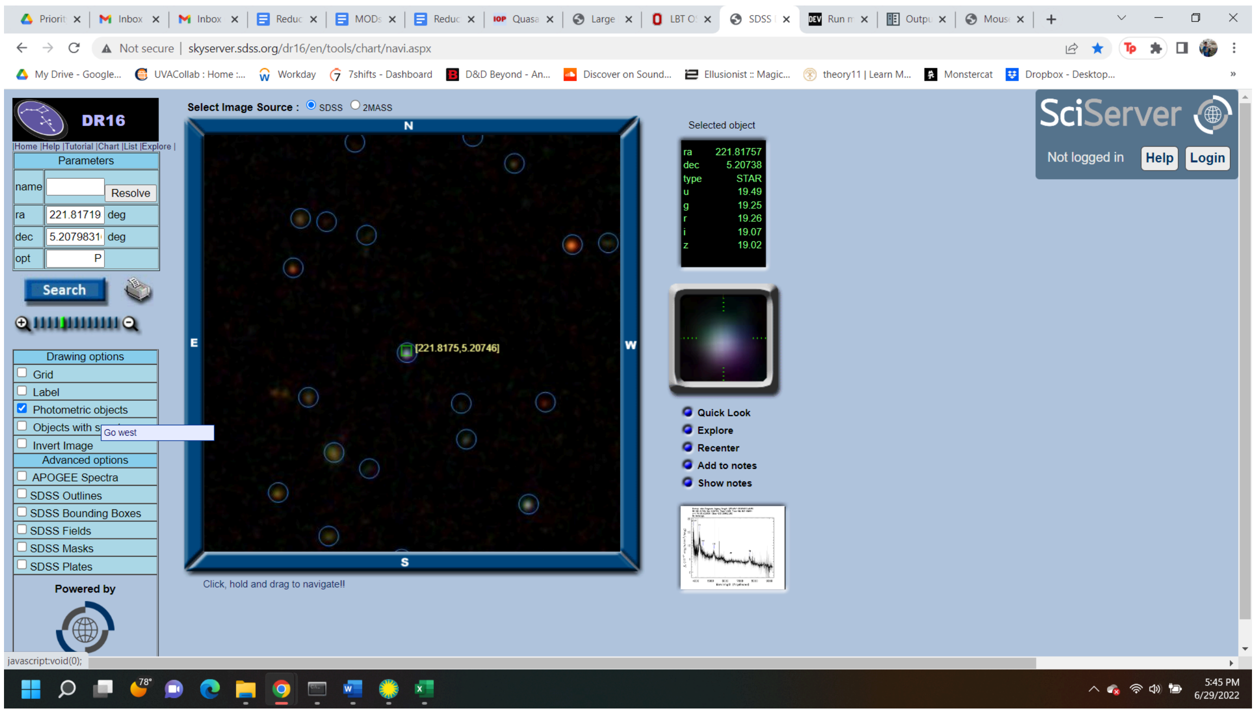
Region	SDSS mag	CCD Counts	Flux	Ratio
------------	#	#	eq in doc	flux/counts
1	18.29	48684	1.35719E-16	2.78775E-21
2	19.92	12278	3.02441E-17	2.46327E-21
3	17.53	103771	2.733E-16	2.63369E-21
4	19.37	21164	5.01927E-17	2.37161E-21
5	18.95	27132	7.38993E-17	2.7237E-21
6	17.32	114464	3.3162E-16	2.89715E-21
7	18.29	48546	1.35719E-16	2.79568E-21

Slope	Intercept
2.83467E-21	-5.66642E-18

AV:	2.66755E-21
STDEV:	1.9026E-22
Object Counts	39281
Object Flux:	1.11349E-16
Mag:	18.50488992


SAOImage ds9
File Edit View Frame Bin Zoom Scale Color Region WCS Analysis Help
file edit view frame bin zoom scale color region wcs analysis help
none region cross colorbar pan zoom rotate crop cat fp exam 3d
N
E
501 502 504 509 518 537 575 651 800
5:45 PM
6/29/2022

File		C:\Users\13zac\OneDrive\Desktop\Research\Data\J1447\Acquisitions\mods1r.20220524.0028.fits		
Object		J1447+05 Acquisition		
Value		602		
FK5	α	221.8199776	δ	5.2013330
Physical	x	327.818	y	438.786
Image	x	327.818	y	438.786
Frame 1	x	0.578704		1483.5937 °


Not secure | skyserver.sdss.org/dr16/en/tools/chart/navi.aspx
DR16
|Home |Help |Tutorial |Chart |List |Explore |
Search
Select Image Source : [✓ SDSS] 2MASS
N
E
W
S
[221.8175,5.20746]
Go west
Click, hold and drag to navigate!!
Quick Look
Explore
Recenter
Add to notes
Show notes
SciServer
Not logged in
Help
Login
Powered by
5:45 PM
6/29/2022

Parameters

name		Resolve
ra	221.81719	deg
dec	5.2079831	deg
opt	P	

Drawing options
- [ ] Grid
- [ ] Label
- [✓] Photometric objects
- [ ] Objects with s
- [ ] Invert Image

Advanced options
- [ ] APOGEE Spectra
- [ ] SDSS Outlines
- [ ] SDSS Bounding Boxes
- [ ] SDSS Fields
- [ ] SDSS Masks
- [ ] SDSS Plates

Selected object

ra	221.81757
dec	5.20738
type	STAR
u	19.49
g	19.25
r	19.26
i	19.07
z	19.02