How to use PypeIt to Reduce LBT MODS Spectra

Katherine Kauma

A Note:

This document is my advice and understanding of how to best reduce LBT MODS data. The bulk of it will be about using the python package “PypeIt” to do the reductions, however at some points I include steps outside of the PypeIt pipeline. In these cases, I also provide the python programs I made to complete these steps. I am sure as time goes on, some of them will become unnecessary as PypeIt grows and includes more functions built into the pipeline. You can run PypeIt without these additional programs (especially the pre-processing) but I would be especially careful to ensure that Pypeit is doing exactly what you want it to be doing. I include side notes and explanations as footnotes in the document.

A lot of this document is borrowed or adapted from the PypeIt documentation. I will link the relevant portions as we go on - but there will be considerable overlap. If you are confused or uncertain about something, I highly suggest reading the relevant portion of the official documentation, as there are a lot of things that I left out! If you have any persistent errors or burning questions that cannot be answered by this guide or the official documentation, try asking in the PypeIt slack channel, which is meant for that kind of thing (link in next section).

I advise you to reduce and work with only one detector at a time (eg MODS 1B, 2B, 1R, or 2R). The steps are slightly different for reduction across red and blue detectors. This guide runs through the steps necessary to reduce data from a single detector at a time. You will need to repeat the reduction steps (with some variation) for 1B, 2B, 1R, and 2R. The reduction steps differ slightly across red and blue spectrographs.

Finally, I have included an example of each step following through the full reduction of an object.

Official PypeIt Resources:

Pypeit documentation: https://pypeit.readthedocs.io/en/release/index.html
Pypeit Users Slack channel: (invitation link here) https://github.com/pypeit/PypeIt/issues/676
Pypeit Github: https://github.com/pypeit/PypeIt

Installation and Setup

Requirements

In order to run Pypeit, you will need Python version 3.7 or greater. You will also need to install pip (or pip3), which is a python package manager and installer: https://pip.pypa.io/en/stable/installation/ . For running PypeIt, the developers suggest having 32 GB of RAM or greater, as the pipeline is memory intensive.[1] You should use pip (or pip3) to install the python package virtualenv. You should also install `specutils`, which is a python package I use in one of my scripts. This can be installed via

```
pip install specutils
```

Or

```
pip3 install specutils
```

You will also need to install and run PypeIt using the command line, there is no GUI.

PypeIt Installation

An in-depth description of the installation procedure is provided here: https://pypeit.readthedocs.io/en/release/installing.html

In summary:

1. Run `virtualenv pypeit` from the home directory
2. Run `source pypeit/bin/activate` to activate the virtualenv. You will need to run this code each time you open a new terminal and want to use pypeit
3. Run `pip install pypeit[pyqt5]`

The specifics above may be slightly different in terms of syntax for different operating systems.

Additionally, you will need to download the atmospheric model grid files used for flux calibration on mods1r and mods2r. These are not included in the initial installation. They can be accessed in the Telluric directory of the Pypeit Google Drive folder. These are large files, so only download the one you need. You should download the "`TelFit_MountGraham_5500_10500_R10000.fits`" file. Move these files to "`path/to/pypeit/data/telluric/atm_grids`"

Katherine's .fits manipulation programs

While PypeIt is a very useful and powerful reduction tool, there were some parts of the pipeline that I felt I needed to supplement with my own programs. The majority of these are basic fits file manipulations used on the data before running the pipeline.

---

[1] Pypeit regularly uses up to 70% of my 32 GB of RAM when running the main pipeline.

You can access these python scripts in the “Katherine’s programs” folder of the Google Drive. To install, download them and make the files executable - in linux or MacOS, to do this run `sudo chmod +x <filename>.` You can either put the files in a folder either already added to your PATH or add the folder you put them in to your PATH. Doing this allows you to run the programs from anywhere in your command line.[2]

You also need to edit the first line of each of the programs. The programs start with a line that has
`#! usr/bin/python3`
You need to replace the path usr/bin/python3 to the path that points to your own python install. You can determine what path you should use by running the following command in terminal
`which python`
Or
`which python3`
The command will output the appropriate path.

The programs included are:

- `add.py` - adds 2D spectra together
- `combine.py` - median or mean combines 2D spectra together
- `crop_spec.py` - truncates 1D spectra to a specified wavelength range
- `show_spec.py` - a viewer for 1D spectra

File Organization and Data

Again, these instructions are only for reducing data from one detector at a time. You will need to repeat all steps for each detector, but I suggest fully reducing the data in one before moving on, instead of reducing all of them in parallel.

Data needed

Get data from: archive.lbto.org/

To process MODS data you will need science frames (your object), standard star frames, arc frames, bias frames, slitless flat frames, and at least one slitted flat frame. Make sure that all of the frames have the same image size and binning, and that the science and slitted flat frame have the same slit width.
Some specifics for each frame, and what they are called in the PypeIt vernacular

- Science frames - `science` - These are frames of your object. Have at least three.
- Standard star frames - `standard` - Your standard star. Have at least three.

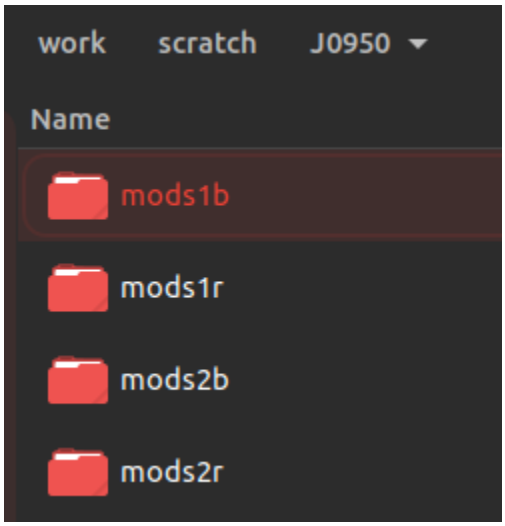
[2] Adding a folder to PATH is different across operating systems. If you are unsure how to do this, there are many resources online.

- Arc frames - `arc` and `tilt` - these are your arcs/lamps/comps. There are usually three that have the spectra of different lamps. They are used for wavelength calibration and for correcting the tilt of 2D spectra.
- Bias frames - `bias` - Make sure that the ones you use are the same size (8kx3k, usually) as your science frames. Have at least three.
- Slitless flat frames - `illumflat` and `pixelflat` - These are the frames used in flat-fielding. Make sure they are not saturated. For MODS 1B and 2B, you will need two sets of three flats, with the UG5 filter and the clear filter. Have at least three for each red detector; have double that for each blue detector.
- Slitted flat frame - `trace` - This frame is used to determine the location of slit edges on the detector and the traces. This is a necessary frame, if it is not present, the trace will not follow the object all the way down the detector in later reduction. Have one frame.

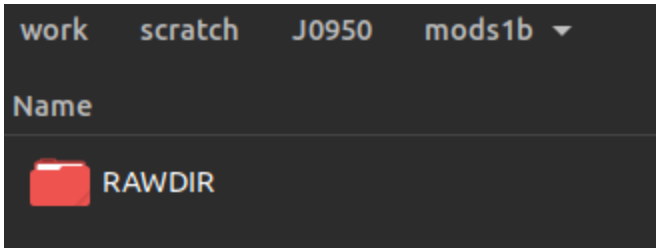
Your science and standard frames should be from the same night. However, arc, bias, and flat frames are stable over observing runs and you can use ones taken on other nights of the observing run. A quick note about the file types - files downloaded from archive.lbto.org are usually gzipped (end in .gz). You should unzip them before doing any pre-processing or reduction.

File/Folder Organization

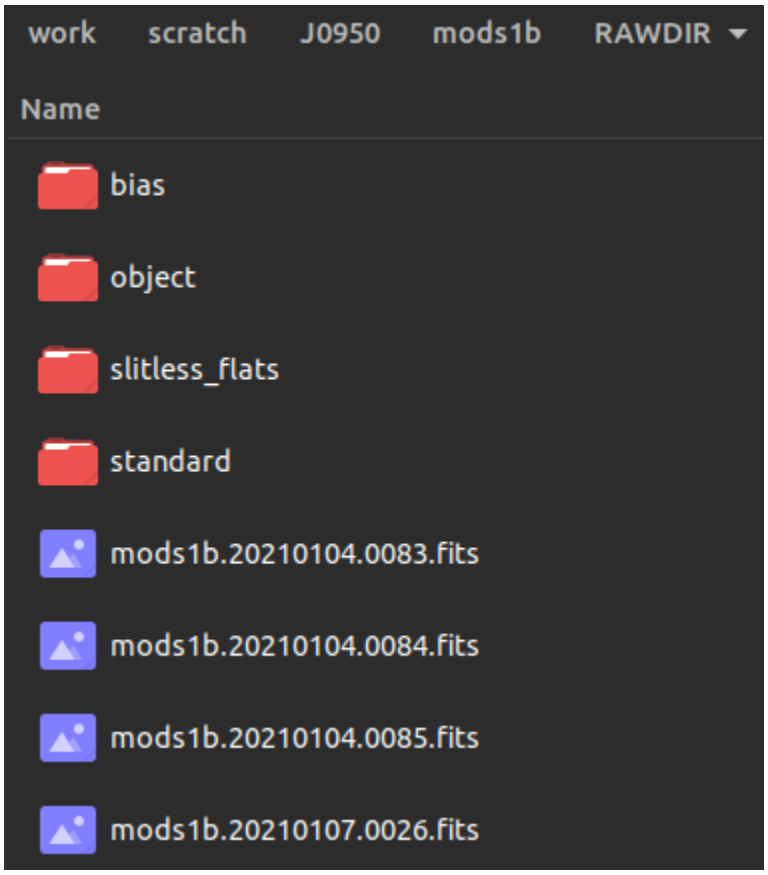
These are my suggestions - you do not have to follow this exactly, but it makes life easier. Have a main folder for your object, and in it, include a folder for each detector: 1b, 2b, 1r, and 2r. I will refer to the detector-specific folders as the “main” folder for reduction of data for each detector.



In your main detector folder, you need a subdirectory for raw files from that detector, which we will call RAWDIR.



This is the only folder you need to run PypeIt, as PypeIt will create and organize directories of its output as it runs. Within the RAWDIR folder, you will want to place the data files that you want to use in the pipeline. I suggest including another folder in your RAWDIR for your raw object, standard, bias, and slitless flat files for pre-processing. You can put your arc and slitted flat files directly in the RAWDIR folder, as those do not need to be modified.



After you have completed the data pre-processing (see next section), you should include your pre-processed data frames in the first level of the RAWDIR directory, as PypeIt only uses those files.

Data Pre-processing

Before running PypeIt, you should median (or mean, if it suits your needs better) combine some of the frames. This step should be done for the science, bias, standard, and slitless flat frames separately. The process is slightly different for the blue flat frames.

Combining 2D spectra:

Use the combine.py program to median or mean combine spectra.

```
katherine@aster:~$ combine.py -h
usage: combine.py [-h] -o OUTPUT [-m] F [F ...]

Median or mean combine any number of 2D spectra

positional arguments:
  F                     Input files to be combined

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        Output filename
  -m, --mean            Mean combine the input files (default: median combine)
```

To median combine spectra, type in the command line
`combine.py <spectra1name>.fits <spectra2name>.fits <etc> -o <outputfilename>.fits`
If you want to mean combine the files, include `-m` or `--mean` in your command.

Example:

```
katherine@aster:~/work/scratch/J0950/mods1b/RAWDIR/bias$ combine.py mods1b.20210107.
0032.fits mods1b.20210107.0033.fits mods1b.20210107.0034.fits mods1b.20210107.0035.f
its mods1b.20210107.0036.fits -o mods1b_bias.fits
```

If your files are in the mods1b.data.number format, you can also run combine the filenames into one input, using `mods1b.data.[range of numbers]`.

```
katherine@aster:~/work/scratch/J0950/mods1b/RAWDIR/bias$ combine.py mods1b.20210107.
003[2-6].fits -o mods1b_bias.fits
```

Combining blue flats:

For the red flats, you can simply combine the files as normal. However, for the blue flats, there are two sets using different filters. First, you will median combine files of the same filter, then you will add together the median combined UG5 file and the median-combined Clear filter file.

To add two spectra, use the `add.py` program.

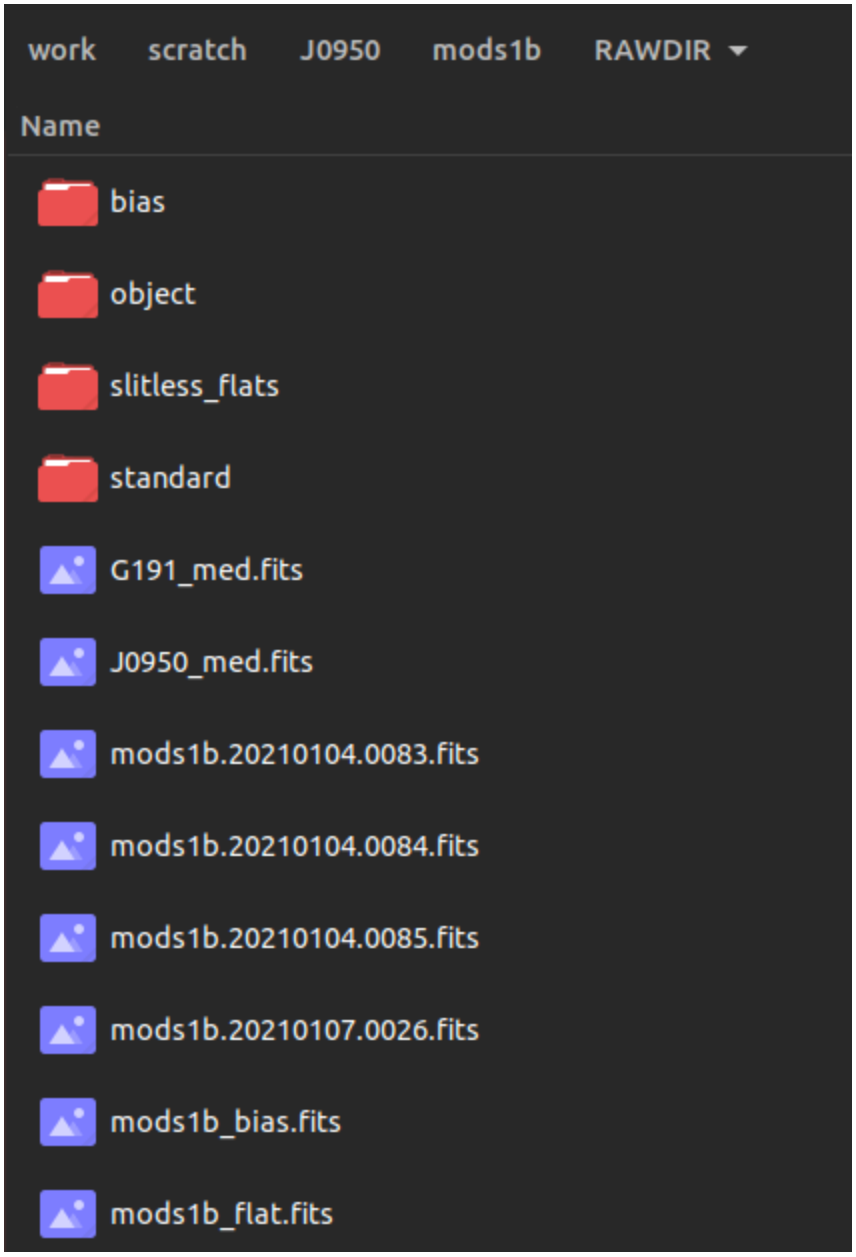
```
katherine@aster:~$ add.py -h
usage: add.py [-h] -o OUTPUT F [F ...]

Add 2D spectra together

positional arguments:
  F                     Input files to be summed

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        Output filename
```

To add spectra, type in the command line
`add.py <spectra1name>.fits <spectra2name>.fits -o <outputfilename>.fits`

Example of blue flat pre-processing:

```
katherine@aster:~/work/scratch/J0950/mods1b/RAWDIR/slitless_flats$ combine.py mods1b
.20210104.007[3-7].fits -o mods1b_clear.fits
katherine@aster:~/work/scratch/J0950/mods1b/RAWDIR/slitless_flats$ combine.py mods1b
.20210104.007[8-9].fits mods1b.20210104.008[0-2].fits -o mods1b_ug5.fits
katherine@aster:~/work/scratch/J0950/mods1b/RAWDIR/slitless_flats$ add.py mods1b_cle
ar.fits mods1b_ug5.fits -o mods1b_flat.fits
```

After pre-processing the science, standard, bias, and flat frames, do not forget to put the new files in the first level of the RAWDIR folder.



Running Pypeit

Now, we can run PypeIt!

Summary of Steps:

1. Run pypeit_setup
2. Modify the .pypeit file
3. Run the pipeline via run_pypeit
4. Inspect spectra outputs

5. Flux calibrate spectra

Setup:

Official documentation: https://pypeit.readthedocs.io/en/release/setup.html

The first step in the pipeline is to run the pypeit_setup script. This prepares the .pypeit file and sorts your data for reduction. The developers suggest running this script twice, the first time to inspect the possible outputs of data sorting, and the second to actually create the .pypeit file and sort the data. Execute this step from the main directory not the RAWDIR.

On the first execution, run
`pypeit_setup -r <path to data> -s <spectrograph name>`
`-r` is the path to the raw files, in our case RAWDIR. `-s` is the spectrograph. For LBT mods, your options are `lbt_mods1b, lbt_mods2b, lbt_mods1r,` and `lbt_mods2r`.

```
katherine@aster:~/work/scratch/J0950/mods1b$ pypeit_setup -r RAWDIR/ -s lbt_mods1b
```

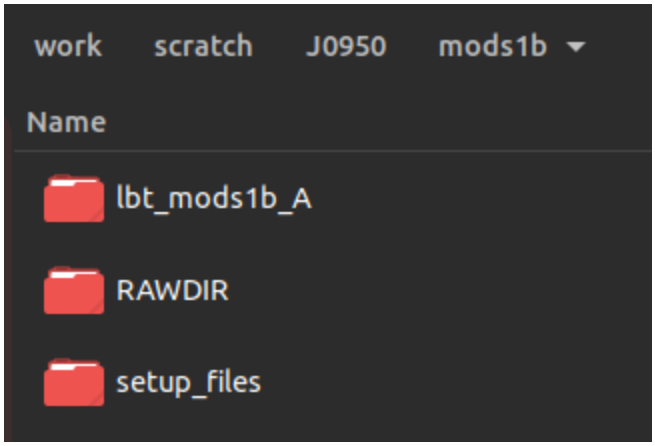
The first execution creates a setup_files directory, which contains a file ending in .sorted. Inspect this file - it will contain one or more configurations of the data, sorted into different "frametypes". Find which configuration best matches your data. If you have organized your data in the way I suggested, configuration A is almost always going to be correct.

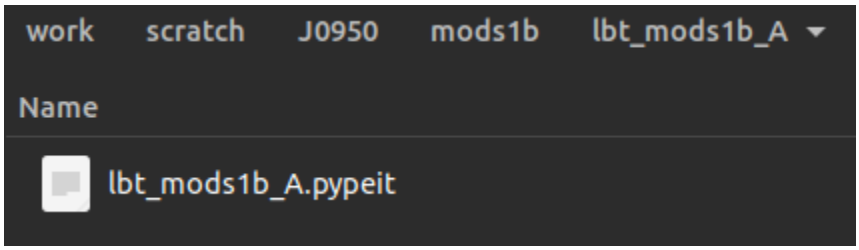
```
##########################################################
Setup A
    dispname: G400L
     binning: 1,1
#---------------------------------------------------------
|                filename |                frametype |                 ra |               dec |                                target | dispname |     decker | binning |          mjd | airmass | exptime |
|            G191_med.fits |                    None |  76.37755416666666 | 52.831100000000006 |               G191-B2B dual grating |    G400L |     LS60x5 |     1,1 | 59218.274561 |    1.09 |    90.0 |
|           J0950_med.fits |                 science | 147.65316666666666 | 51.477222222222224 |                              J0950+51 |    G400L | LS5x60x1.0 |     1,1 | 59218.382326 |    1.08 |   360.0 |
|         mods1b_flat.fits |                    None | 147.65316666666666 | 51.477222222222224 | VFLAT5.0 Clear Dual Slitless Pix Flat |    G400L |    Imaging |     1,1 | 59218.599317 |     1.0 |     3.0 |
| mods1b.20210104.0083.fits |                 arc,tilt | 147.65316666666666 | 51.477222222222224 |                      Ne + Hg Lamps |    G400L | LS5x60x0.6 |     1,1 | 59218.621836 |     1.0 |     2.0 |
| mods1b.20210104.0084.fits |                 arc,tilt | 147.65316666666666 | 51.477222222222224 |                      Kr + Xe Lamps |    G400L | LS5x60x0.6 |     1,1 | 59218.623284 |     1.0 |    15.0 |
| mods1b.20210104.0085.fits |                 arc,tilt | 147.65316666666666 | 51.477222222222224 |                           Ar Lamp |    G400L | LS5x60x0.6 |     1,1 | 59218.624931 |     1.0 |    30.0 |
| mods1b.20210107.0026.fits | pixelflat,illumflat,trace | 159.90308333333334 |  43.10257222222222 |  QTH1 ND1.5 Dual LS5x60x1.0 Slit Flat |    G400L | LS5x60x1.0 |     1,1 | 59221.599151 |     1.0 |    16.0 |
|         mods1b_bias.fits |                      bias | 159.90308333333334 |  43.10257222222222 |                         Bias 8Kx3K |    G400L | LS5x60x0.8 |     1,1 | 59221.633137 |     1.0 |     0.0 |
##end
```

Execute pypeit_setup again, this time including the `-c` argument to write the .pypeit file for a specific setup.
`pypeit_setup -r <path to raw> -s <spectrograph name> -c <letter corresponding to the right configuration>`

```
katherine@aster:~/work/scratch/J0950/mods1b$ pypeit_setup -r RAWDIR/ -s lbt_mods1b -c A
```

This step creates a folder called <spectrograph name>_<config letter> and in it, a file ending in .pypeit.





Modify the .pypeit file:

The pypeit reduction file instructs PypeIt on what data to reduce and how. It has three main parts, a parameter block, a setup block, and a data block. The parameter block allows you to customize how you want the data reduced - for example, you can specify the number and location of objects you want PypeIt to find. You only need to include parameters if you want to change them from the instrument-specific defaults. For LBT Mods data, this turns out to be necessary. Next is the setup block - this can be left alone. Finally, the data block contains the data file names and frame types. You will likely need to change the frame types on some files, as PypeIt may not recognize the type or will mischaracterize it.

The original .pypeit file will look something like this:

```
# Auto-generated PypeIt file
# 2021-09-04

# User-defined execution parameters
[rdx]
spectrograph = lbt_mods1b

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

# Read in the data
data read
 path RAWDIR
|                 filename |                frametype |                 ra |                dec |                                target | dispname |     decker | binning |          mjd | airmass | exptime |
|             G191_med.fits |                    None |  76.37755416666666 | 52.831100000000006 |             G191-B2B dual grating |    G400L |     LS60x5 |     1,1 | 59218.274561 |    1.09 |    90.0 |
|         mods1b_flat.fits |                    None | 147.65316666666666 | 51.477222222222224 | VFLAT5.0 Clear Dual Slitless Pix Flat |    G400L |    Imaging |     1,1 | 59218.599317 |     1.0 |     3.0 |
| mods1b.20210104.0083.fits |                 arc,tilt | 147.65316666666666 | 51.477222222222224 |                    Ne + Hg Lamps |    G400L | LS5x60x0.6 |     1,1 | 59218.621836 |     1.0 |     2.0 |
| mods1b.20210104.0084.fits |                 arc,tilt | 147.65316666666666 | 51.477222222222224 |                    Kr + Xe Lamps |    G400L | LS5x60x0.6 |     1,1 | 59218.623284 |     1.0 |    15.0 |
| mods1b.20210104.0085.fits |                 arc,tilt | 147.65316666666666 | 51.477222222222224 |                       Ar Lamp |    G400L | LS5x60x0.6 |     1,1 | 59218.624931 |     1.0 |    30.0 |
|          mods1b_bias.fits |                     bias | 159.90308333333334 |  43.10257222222222 |                   Bias 8Kx3K |    G400L | LS5x60x0.8 |     1,1 | 59221.633137 |     1.0 |     0.0 |
| mods1b.20210107.0026.fits | pixelflat,illumflat,trace | 159.90308333333334 |  43.10257222222222 | QTH1 ND1.5 Dual LS5x60x1.0 Slit Flat |    G400L | LS5x60x1.0 |     1,1 | 59221.599151 |     1.0 |    16.0 |
|            J0950_med.fits |                  science | 147.65316666666666 | 51.477222222222224 |                        J0950+51 |    G400L | LS5x60x1.0 |     1,1 | 59218.382326 |    1.08 |   360.0 |
data end
```

Parameter block:

In order for PypeIt to run without errors, you will need to add the following lines to the parameter section:

```
[calibrations]
  [[slitedges]]
    edge_thresh = 20
```

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

The first chunk increases the sensitivity for finding the edges of the slit(s) in the trace file. The default is 100, which results in no edges being found and an error. The second chunk is a preventative measure - PypeIt often masks bright portions of the standard star spectrum, thinking they are cosmic rays. By making use_2dmodel_mask False, PypeIt does not use the mask (though the mask will still show up when you view the 2D spectra).

These are the parameters necessary to run, but you may want to include more. If, for instance, your object is very faint, you can decrease the threshold in the object finding algorithm:

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

You can also manually input the location of the object. A full list of parameters and their uses are available here:
https://pypeit.readthedocs.io/en/release/pypeit_par.html

Data block:

Next, make sure the frametypes are correct for you data. The most frequent changes I need to make are specifying the standard star, adding the “pixelflat,illumflat” label to my flat, and removing the “pixelflat,illumflat” label from my trace.

With all of these changes, my final .pypeit file looks like this:

```
# Auto-generated PypeIt file
# 2021-09-04

# User-defined execution parameters
[rdx]
spectrograph = lbt_mods1b

[calibrations]
  [[slitedges]]
    edge_thresh = 20

[reduce]
 [[extraction]]
   use_2dmodel_mask = False
 [[findobj]]
   sig_thresh=3

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

# Read in the data
data read
 path RAWDIR
|                 filename |                frametype |                 ra |               dec |                                target | dispname |     decker | binning |          mjd | airmass | exptime |
|             G191_med.fits |                 standard |  76.37755416666666 | 52.831100000000006 |                 G191-B2B dual grating |    G400L |     LS60x5 |     1,1 | 59218.274561 |    1.09 |    90.0 |
|          mods1b_flat.fits |       pixelflat,illumflat | 147.65316666666666 | 51.477222222222224 | VFLAT5.0 Clear Dual Slitless Pix Flat |    G400L |    Imaging |     1,1 | 59218.599317 |     1.0 |     3.0 |
| mods1b.20210104.0083.fits |                  arc,tilt | 147.65316666666666 | 51.477222222222224 |                      Ne + Hg Lamps |    G400L | LS5x60x0.6 |     1,1 | 59218.621836 |     1.0 |     2.0 |
| mods1b.20210104.0084.fits |                  arc,tilt | 147.65316666666666 | 51.477222222222224 |                      Kr + Xe Lamps |    G400L | LS5x60x0.6 |     1,1 | 59218.623284 |     1.0 |    15.0 |
| mods1b.20210104.0085.fits |                  arc,tilt | 147.65316666666666 | 51.477222222222224 |                           Ar Lamp |    G400L | LS5x60x0.6 |     1,1 | 59218.624931 |     1.0 |    30.0 |
|          mods1b_bias.fits |                      bias | 159.90308333333334 |  43.10257222222222 |                        Blas 8Kx3K |    G400L | LS5x60x0.8 |     1,1 | 59221.633137 |     1.0 |     0.0 |
| mods1b.20210107.0026.fits |                     trace | 159.90308333333334 |  43.10257222222222 | QTH1 ND1.5 Dual LS5x60x1.0 Slit Flat |    G400L | LS5x60x1.0 |     1,1 | 59221.599151 |     1.0 |    16.0 |
|            J0950_med.fits |                   science | 147.65316666666666 | 51.477222222222224 |                              J0950+51 |    G400L | LS5x60x1.0 |     1,1 | 59218.382326 |    1.08 |   360.0 |
data end
```

You may need to go back and change the parameters if the reduction does not work on the first try.

Run pipeline:

Official Documentation: https://pypeit.readthedocs.io/en/release/running.html

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.

Inspect outputs:

Official Documentation: https://pypeit.readthedocs.io/en/release/outputs.html

As it runs, PypeIt will create a Masters folder, a QA folder, and a Science folder. The Masters folder contains the master bias, master flats, master arcs, etc. The QA folder has quality assurance files for the wavelength calibration and tilts. The Science folder will contain processed 2D and 1D spectra for the science object and the standard star. If you need to re-run PypeIt, make sure to delete these folders, otherwise it will use the Masters folder in the next run.

2D spectrum documentation: https://pypeit.readthedocs.io/en/release/out_spec2D.html

Once PypeIt has finished running, you should view the 2D spectra of the science object and the standard star to ensure that PypeIt has correctly identified the objects and reduced the data. Use the `pypeit_show_2dspec` script to view the 2D spectra in ginga. It will open a ginga window and show 4 channels: the original science image, the sky subtracted image, and two residual plots. On these images, the red and green lines indicate the locations of slit edges. Orange lines are objects found by the PypeIt object-finding algorithm, and blue lines are objects that were manually specified.

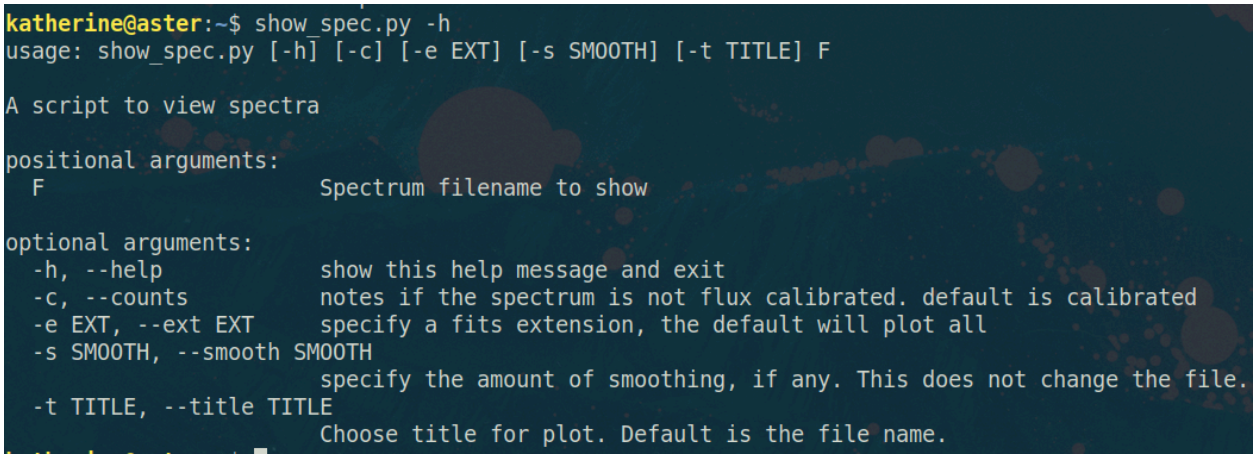
You will want to make sure that the objects you are interested in were identified correctly. If they were not found, then you should rerun pypeit and either manually indicate the location or adjust the `sig_thresh` parameter. If PypeIt identified more than one object in your slit, it is not an issue. From left to right, the order of traces is the order of the spectra in the fits file extensions. For example, if your object is the third trace from the left, it will be in extension 3. This is necessary for plotting the spectra and accessing the spectra after reduction. If you want to limit the number of traces found, you can do that with the `maxnumber [[findobj]]` parameter in the .pypeit file.

If everything in the 2D spectra looks good, then you can view the 1D spectra. Otherwise, re-run pypeit with the necessary changes.

1D spectrum documentation: https://pypeit.readthedocs.io/en/release/out_spec1D.html

Pypeit comes with a built-in 1D spectrum viewer, however I did not like it very much so I also made my own version. You can use either. To view using Pypeit, use the `pypeit_show_1dspec` script.

To view using my program, execute `show_spec.py <spec1d filename>`. If run without any optional arguments, this script will plot every extension of the fits file on one figure and make the title the file name. The default y-axis units are 10^-17 ergs AA^-1 s^-1 cm^-2. Using optional arguments, you can switch the y units to counts (`-c`). You can also specify a single extension to plot (`-e <extension number>`), smooth the spectrum (`-s <# pixels to smooth by>`) or specify a title (`-t <title>`).



If you are satisfied with the results of the pipeline, you can proceed to fluxing.

Fluxing:

Official documentation: https://pypeit.readthedocs.io/en/release/fluxing.html

BEFORE DOING ANY FLUXING, MAKE A COPY OF YOUR SCIENCE DIRECTORY. The fluxing process changes the files. If you make a mistake and need to try again, it's useful to have a copy of the original files available. Otherwise, you may need to re-run the pipeline, which takes time. Name it something like Science_original.

In pypeit, fluxing has three main steps:

1. Running the pypeit_sensfunc script
2. Running pypeit_flux_setup script and modifying the .flux file
3. Running the pypeit_flux_calib script

However, there are smaller steps in between that differ for the blue and the red mods detectors. I will describe the fluxing method for blue and red spectra below. For all of these steps, run them from inside the Science directory.

Mods Blue:

1. In the science directory, the first thing you will do is run the command `pypeit_sensfunc --algorithm UVIS -o sens.fits <file name for standard star ending in .fits>`

This command runs the pypeit script `pypeit_sensfunc`, which creates a sensitivity function for your standard star. You do not need to specify which standard star you are using, as PypeIt will use the RA and DEC in the fits header to determine it. For MODS1B and 2B, you should specify the `UVIS` algorithm. Pypeit recommends using this algorithm for spectra that go below 7000 AA. It uses a spline to fit the sensitivity function. The `-o` command names the sensitivity file - you can change it from sensfunc.fits if you would like. The default is just very long to type in later steps.

```
katherine@aster:~/work/scratch/J0950/mods1b/Science$ pypeit_sensfunc --algorithm UVIS
 -o sensfunc.fits spec1d_G191_med-G191-B2Bdualgrating_MODS1B_20210104T063522.070.fits
```

`pypeit_sensfunc` outputs a number of files, including `sensfunc.fits, sensfunc.par, sensfunc_QA.pdf,` and `sensfunc_throughput.pdf. Sensfunc.fits` is the sensitivity function and `sensfunc.par` are the parameters used in creating the sensitivity function. `sensfunc_QA` and `sensfunc_throughput` are files that can be checked for quality assurance. `sensfunc_QA` plots the real spectrum (in log y) and the model used to fit it, so you can diagnose the accuracy of the model by eye.

2. Next, you will run
   `Pypeit_flux_setup .`

This command runs the `pypeit_flux_setup script` and the "`.`" argument tells it that the science folder is located in the current directory. The `pypeit_flux_setup` script creates three files ending in `.flux`, `.coadd1d`, and `.tell`. For our purposes, the file ending in .flux is the only one we are interested in.

```
katherine@aster:~/work/scratch/J0950/mods1b/Science$ pypeit_flux_setup .
[WARNING] :: sensfunc.fits is not a standard PypeIt output.
[WARNING] :: sensfunc.par is not a standard PypeIt output.
[WARNING] :: sensfunc_QA.pdf is not a standard PypeIt output.
[WARNING] :: sensfunc_throughput.pdf is not a standard PypeIt output.
[INFO]    :: PypeIt file written to: lbt_mods1b.flux
[INFO]    :: PypeIt file written to: lbt_mods1b.coadd1d
[INFO]    :: PypeIt file written to: lbt_mods1b.tell
```

Open the .flux file in a text editor. It will look like this:

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

# User-defined execution parameters
[fluxcalib]
  extinct_correct = False # 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
 ./spec1d_J0950_med-J0950+51_MODS1B_20210104T091032.966.fits
flux end
```

The first section is for user-defined parameters. This section is only useful for the UVIS algorithm. The second section includes all of the files that you want to be flux calibrated. The names of the standard star and the object files should appear. If one does not, make sure that it exists and include it.

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.

3. 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.

Mods Red:

The fluxing process for the red spectra is a bit different than for the blue. This stems from the fact that for red spectra, we use the IR algorithm instead of UVIS. The IR algorithm uses a polynomial fit instead of a spline. It also corrects for telluric absorption when creating the sensitivity function.

The first step for red fluxing is to crop the spectra. Blueward of 5800 AA, the red spectrograph sensitivity drops sharply. This change is too drastic for the polynomial fit of the IR algorithm to

accurately characterize without going to extremely high polynomial order. The easiest solution is to truncate the spectra and remove the ends. For this purpose, I created the `crop_spec.py` script.

`Crop_spec.py` can truncate one or more spectra to a wavelength range. The default wavelength range is 5800 AA- 10200 AA. This wavelength range can be changed to a user-specified one. This program will overwrite your original spectrum with the truncated one, so make sure that you have a copy of the original if you make a mistake (for instance, in your Science_unfluxed folder).

```
katherine@aster:~$ crop_spec.py -h
usage: crop_spec.py [-h] [-r RANGE RANGE] F [F ...]

A script to truncate the wavelength range of 1D spectra

positional arguments:
  F                     files to be cropped. This overwrites the original files, so
                        make sure to have a copy saved elsewhere

optional arguments:
  -h, --help            show this help message and exit
  -r RANGE RANGE, --range RANGE RANGE
                        the wavelength range (in angstroms) that you want to have.
                        include two numbers for the minmum and maximum, it does not
                        matter which order. The default range is 5800 AA to 10200
                        AA meant for MODS red
```

To run the program, execute `crop_spec.py <spec1d filenames>`. Running without any arguments will crop the spectra to the default wavelength range. To specify your own wavelength range, use the argument `-r <minimum wavelength in AA> <maximum wavelength in AA>`.

Example using the default wavelength range:

```
katherine@aster:~/work/scratch/J0950/mods1r/Science$ crop_spec.py spec1d_J0950_med-J0
950+51_MODS1R_20210104T091028.560.fits spec1d_G191_med-G191-B2Bdualgrating_MODS1R_202
10104T063515.245.fits
```

Example specifying a wavelength range:

```
katherine@aster:~/work/scratch/J0950/mods1r/Science$ crop_spec.py -r 5800 10200 spec1
d_J0950_med-J0950+51_MODS1R_20210104T091028.560.fits spec1d_G191_med-G191-B2Bdualgrat
ing_MODS1R_20210104T063515.245.fits
```

After cropping the spectra, you will create a `sens_file`. Unlike for the blue, the `--algorithm` tag in `pypeit_sensfunc` does not work for IR. instead, you will provide a `--sens_file` that directs pypeit on how to create the sensitivity function. Once you have created a `sens_file`, you can re-use the same one for any red reductions.
To make the `sens_file`, create a simple .txt file and name it sens_file.txt. In it, copy the following:

```
[sensfunc]
  algorithm = IR
  polyorder = 8
  [[IR]]
     telgridfile =
/path/to/your/python/site-packages/pypeit/data/telluric/atm_grids/Tel
Fit_MountGraham_5500_10500_R10000.fits
```

```
[sensfunc]
  algorithm = IR
  polyorder = 8
  [[IR]]
    telgridfile = /home/katherine/.local/lib/python3.8/site-packages/pypeit/data/telluric/TelFit_Moun
tGraham_5500_10500_R10000.fits
```

For the telgridfile parameter, you will need to make the path point to wherever `TelFit_MountGraham_5500_10500_R10000.fits` is stored. This should be in the pypeit folder where your Python packages are installed. You can change the order of the polynomial used to fit the data by increasing or decreasing the `polyorder` parameter. I recommend starting with a polynomial of order 8.

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>`

```
katherine@aster:~/work/scratch/J0950/mods1r/Science$ pypeit_sensfunc -s sens_file.txt
 -o sens.fits spec1d_G191_med-G191-B2Bdualgrating_MODS1R_20210104T063515.245.fits
```

This will take longer to run for red spectra than it did for the blue.

The next command you run is `pypeit_flux_setup .`

```
katherine@aster:~/work/scratch/J0950/mods1r/Science$ pypeit_flux_setup .
```

After running this, edit the file ending in .flux. You will only need to add the sens.fits file after the first spectrum, like so:

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

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

# User-defined execution parameters
[fluxcalib]
  extinct_correct = False # 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_MODS1R_20210104T063515.245.fits sens.fits
 ./spec1d_J0950_med-J0950+51_MODS1R_20210104T091028.560.fits
flux end
```

Finally, you will run
`pypeit_flux_calib <name of .flux file>`

```
katherine@aster:~/work/scratch/J0950/mods1r/Science$ pypeit_flux_calib lbt_mods1r.flux
```

Check your final spectra results- if all looks good, congratulations! You have successfully reduced your MODS red data.