PyCOS Tutorial

PyCOS is a spectral reduction pipeline for the Cosmic Origin spectrograph on the Hubble Space Telescope.

This document attempts to explain the work-flow for PyCOS

1. Process x1dsum fits files

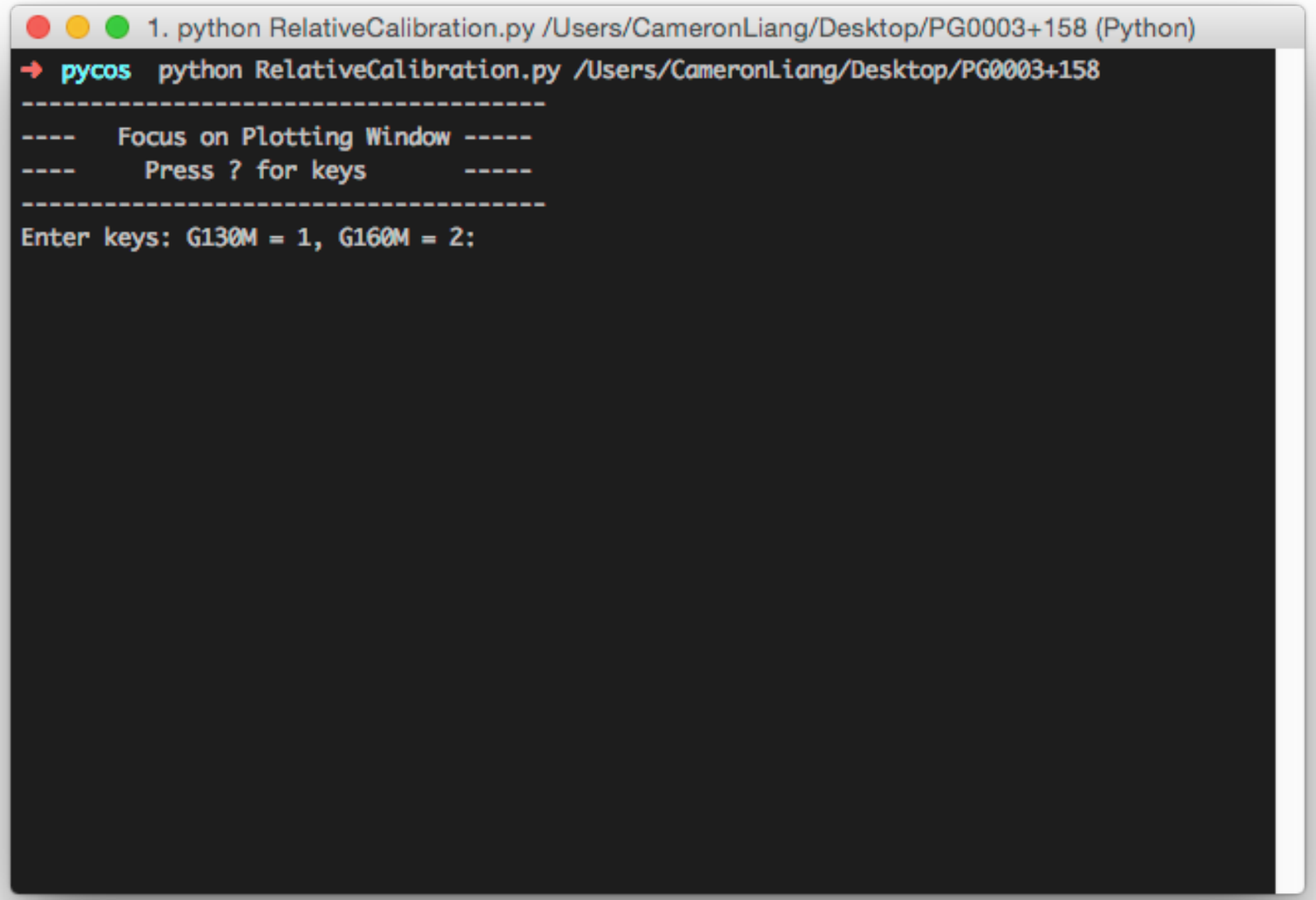
- Download the raw data from HST archive. https://archive.stsci.edu/ (PyCOS only requires intermediate data product with names “xxxxxx_x1dsum.fits. (See COS data handbook for details)

- Put all x1dsum.fits under a directory “raw”. I recommend …/object_name/raw/
    - e.g. /home/username/desktop/PKS0405-123/raw
- Run Process_x1dsum.py as followed:
    - Python Process_x1dsum.py full_path_to_so_name
- This will produce a collection of processed intermediate products output to …/object_name

2. Relative Calibration

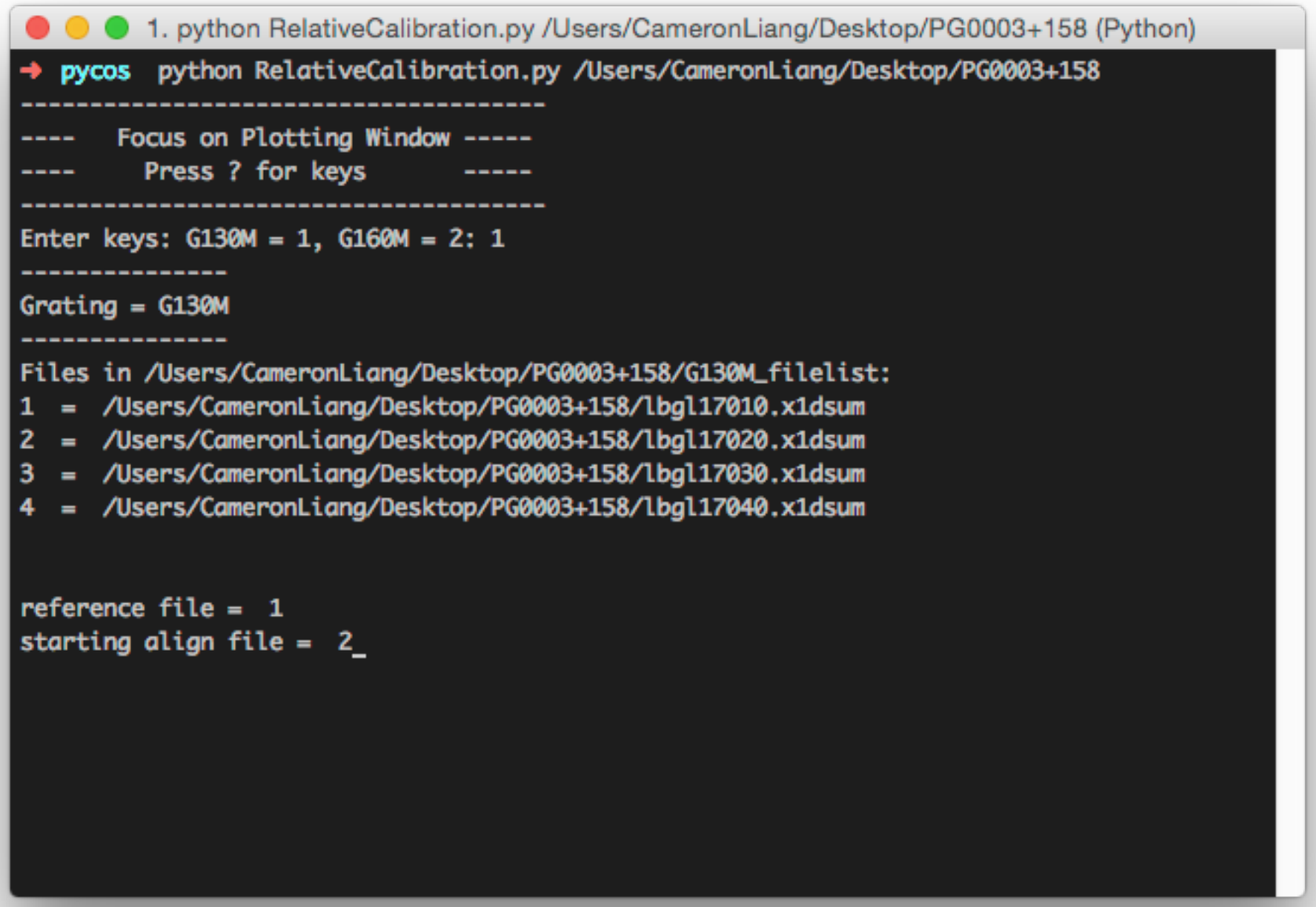
- Run RelativeCalibration.py as followed:
  - python RelativeCalibration.py full_path_to_object_directory

Choose grating (G130M or G160M) to align between exposures



2. Relative Calibration

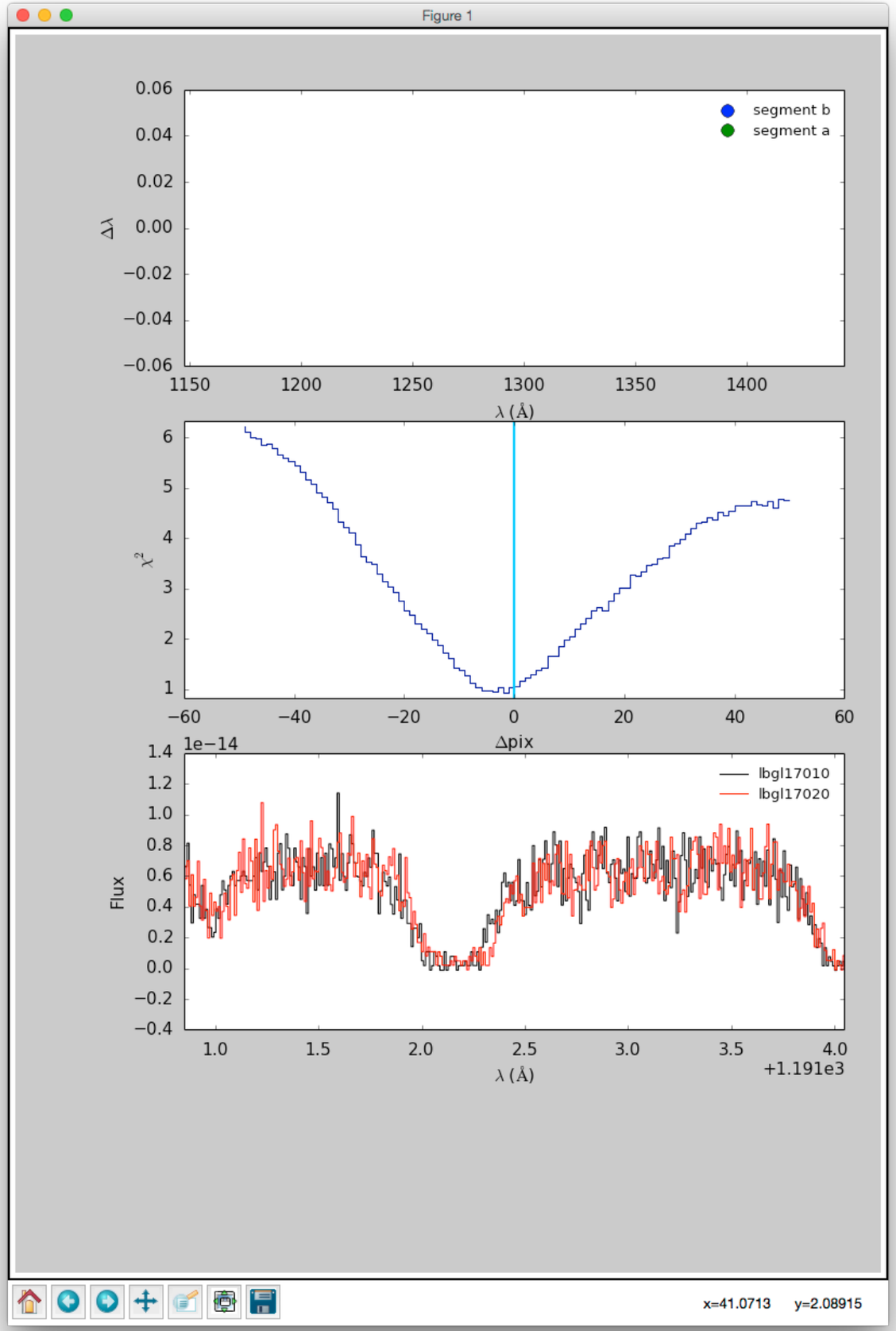
Reference spectrum is automatically chosen
and user to choose another spectrum to align.



2. Relative Calibration

A GUI interface will appear for user to align the spectrum (red) against the reference (black)

Focus on the plotting window, and press “?” on the keyboard - a list of keys mapping to some functions will print in terminal. Follow the key map to proceed for aligning, fitting, and writing a rectified spectrum.



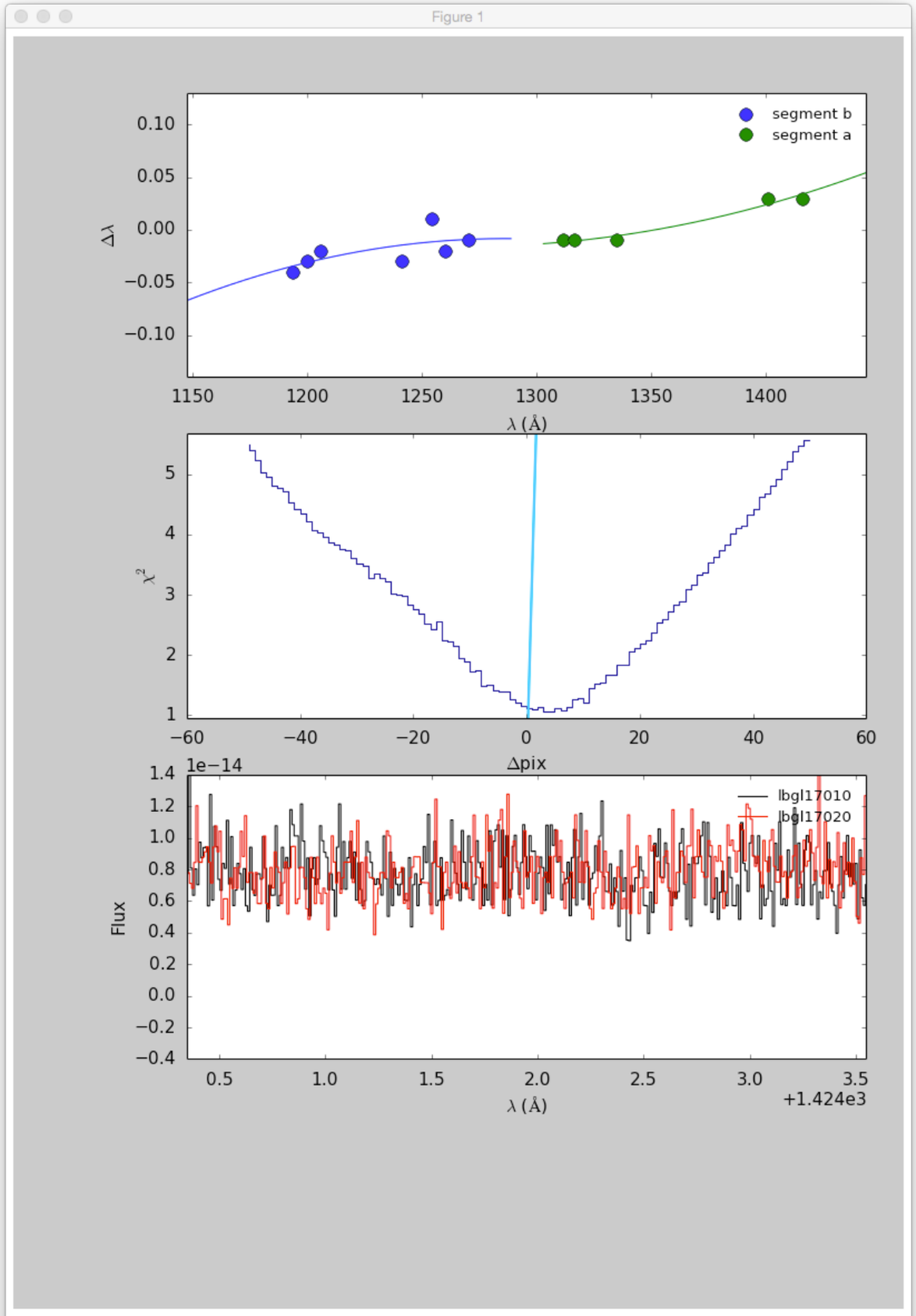
2. Relative Calibration

After a series of alignment between some absorption lines present in the data. There will be a list of points in the top panel. Users can delete points using according to the key map.

Segment a and b can be be fitted independently with a polynomial of order n. We caution users not to unnecessarily high orders due to potential danger in extrapolation near both ends of the spectra.

After writing the new spectrum, users can exit the plotting window. The terminal will now ask whether to proceed to the next available spectra (if available). The process then repeats until no more spectra available.

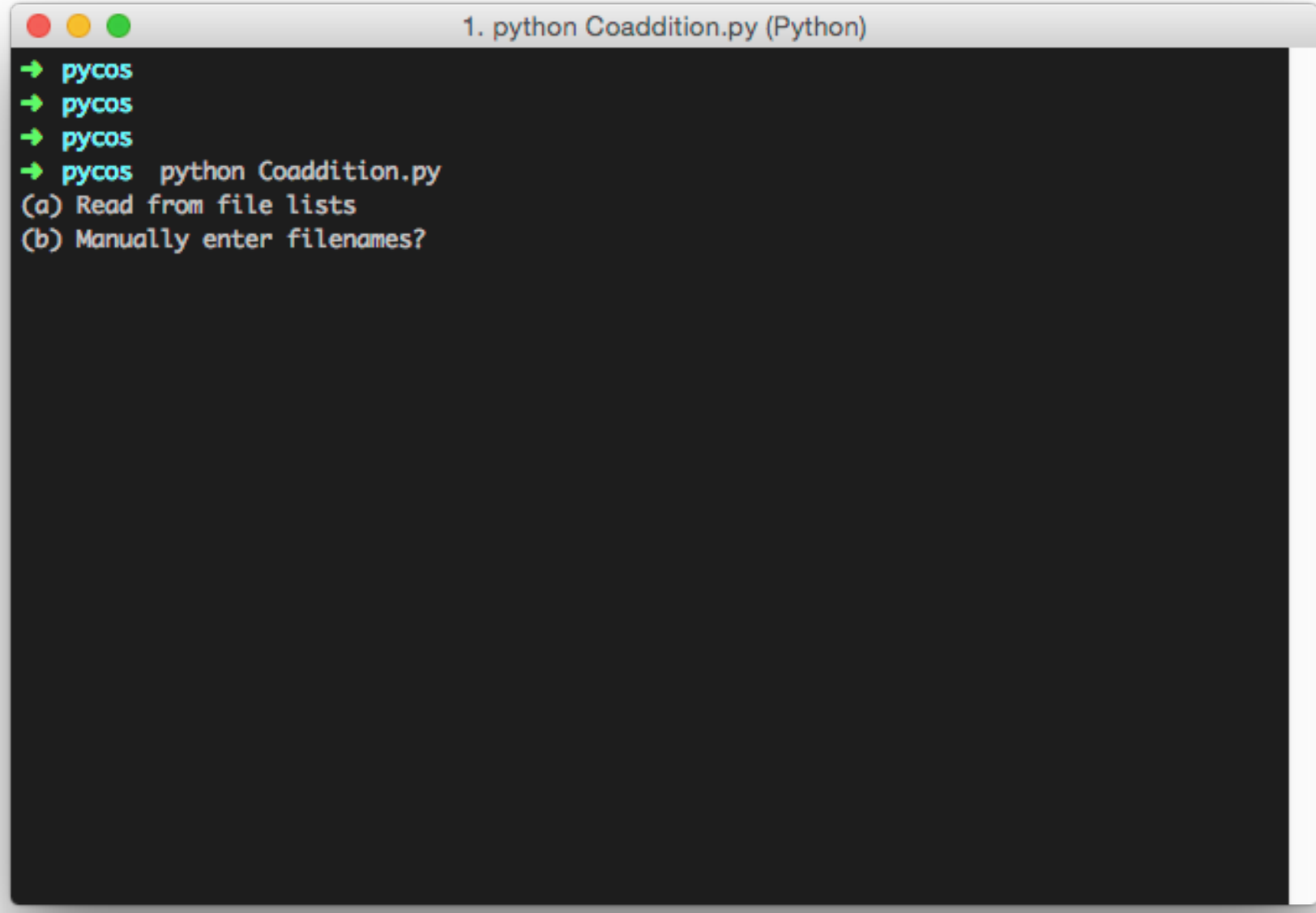
User can choose exit and come back to choose any spectra to align.



3. Co-addition

Run coaddition.py as below:
python Coaddition.py

First, we want to coadd G130M spectra and G160M spectra separately. Choose the first option to read from a file list.



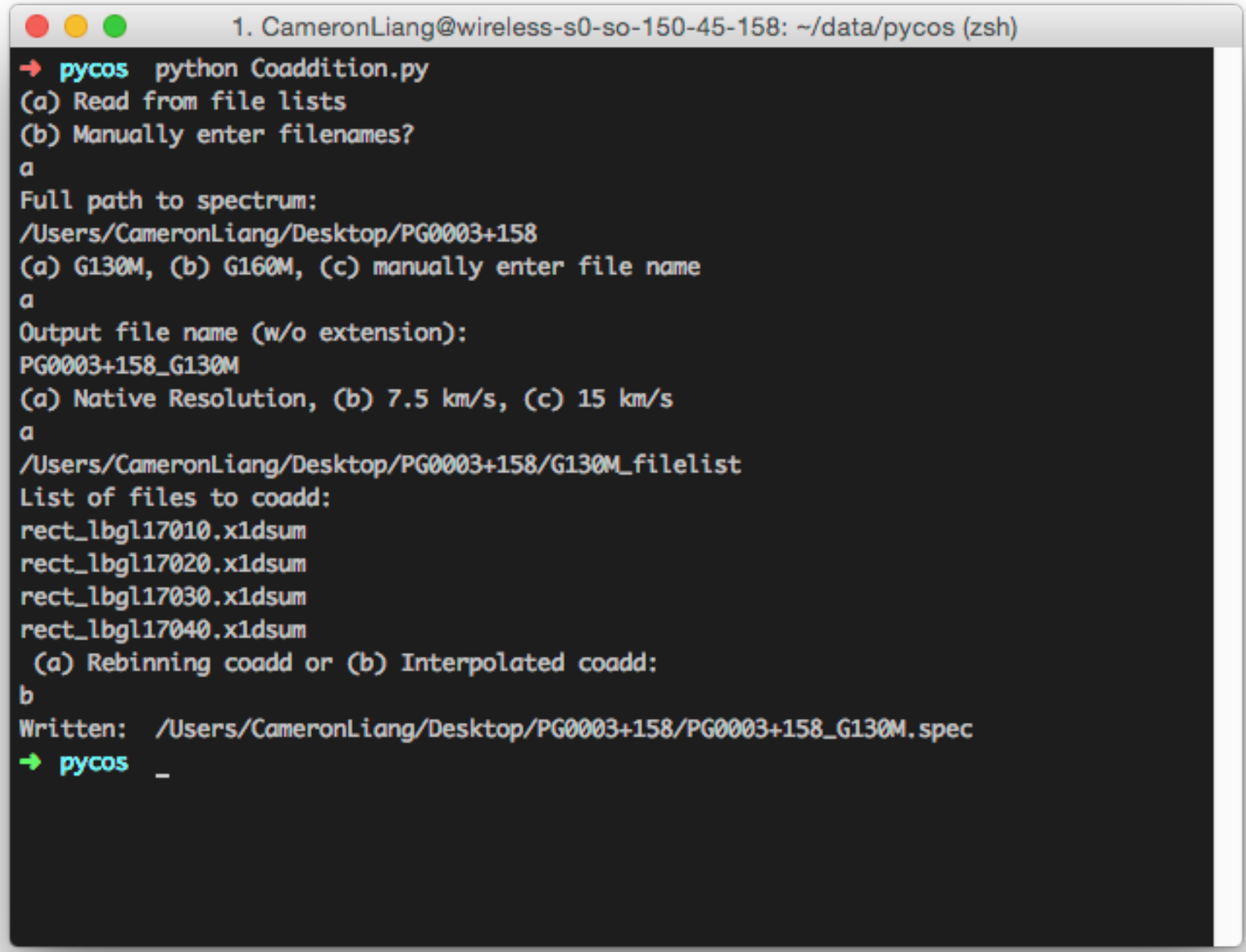
3. Co-addition

Run coaddition.py as below:
python Coaddition.py

First, we want to coadd G130M spectra and G160M spectra separately. Choose the first option to read from a file list (automatically produced by Process_x1dsum.py)

Then choose the resolution of the co-added spectrum. I recommend native resolution, i.e., to use the same wavelength grid as the input.

Repeat the process for G160M data if exists.

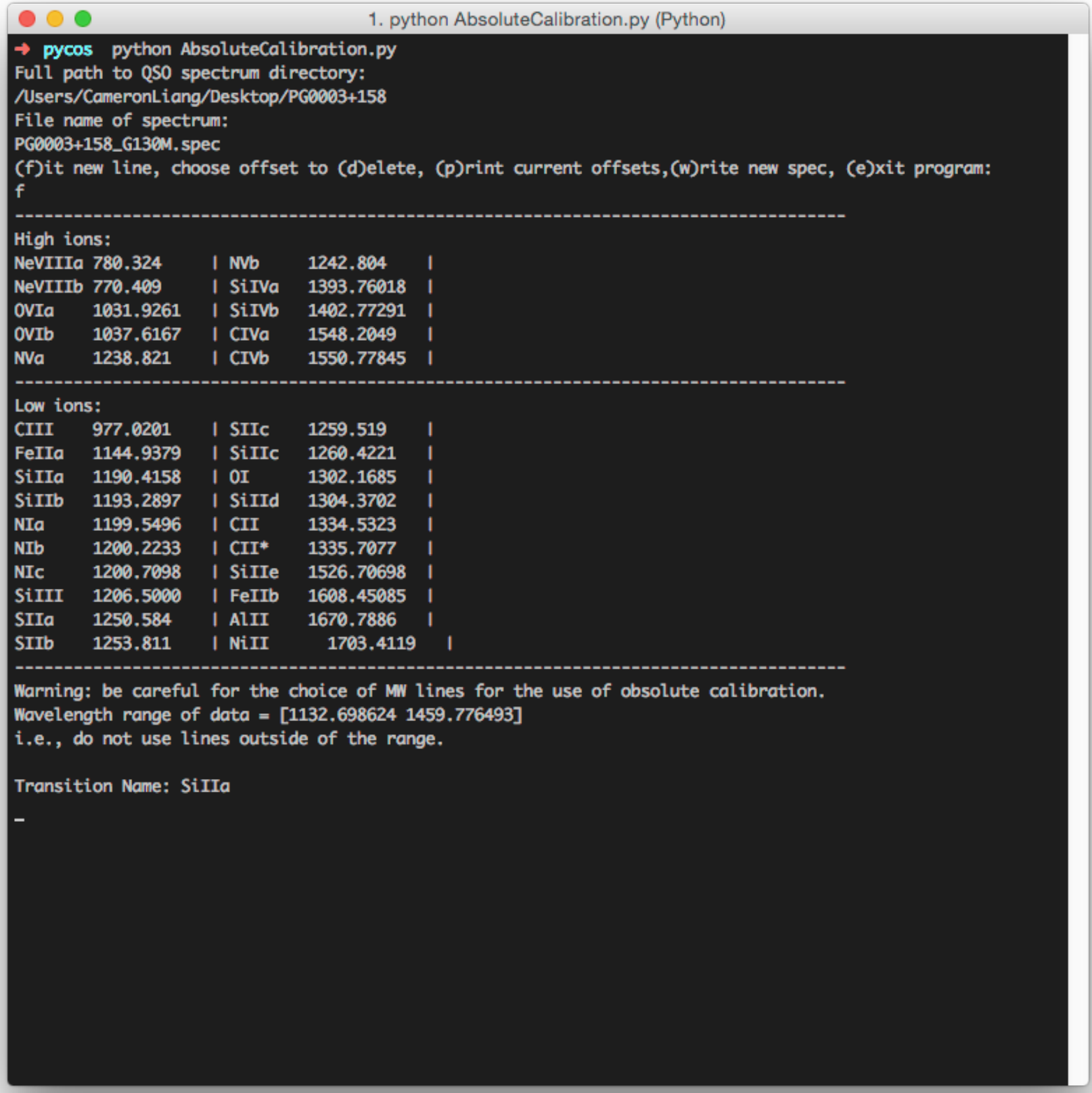


4. Absolute Calibration

Run AbsoluteCalibration.py as below:
python AbsoluteCalibration.py

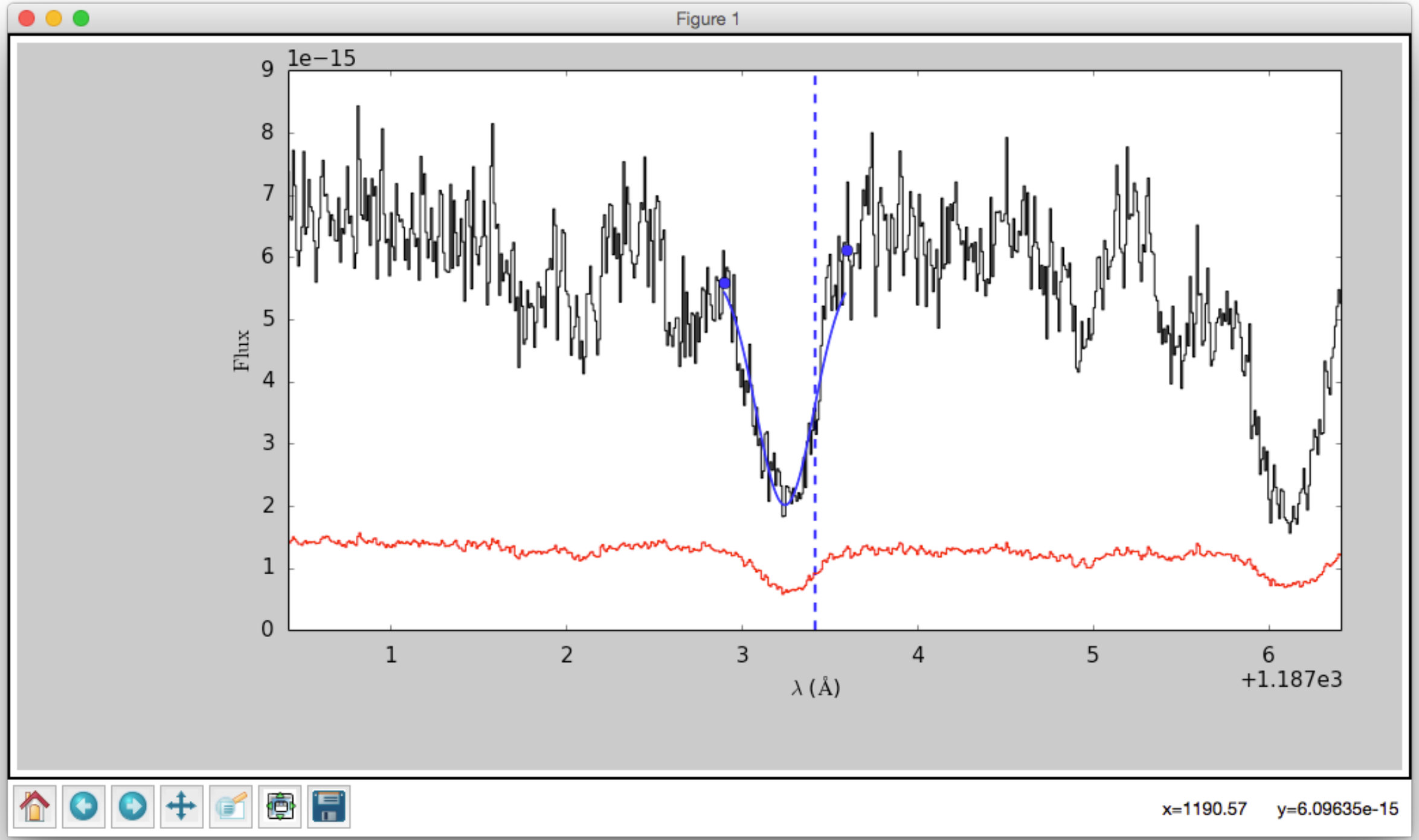
This module works by fitting gaussian to obtain the observed centroid of a given Milky-Way absorption lines to give a measure of absolute wavelength calibration. Then a polynomial fit is used to correct any elongation/ compression of the wavelength array.

For the first time entering the program choose (f) for fitting a new line. Enter the name of the transition to fit.



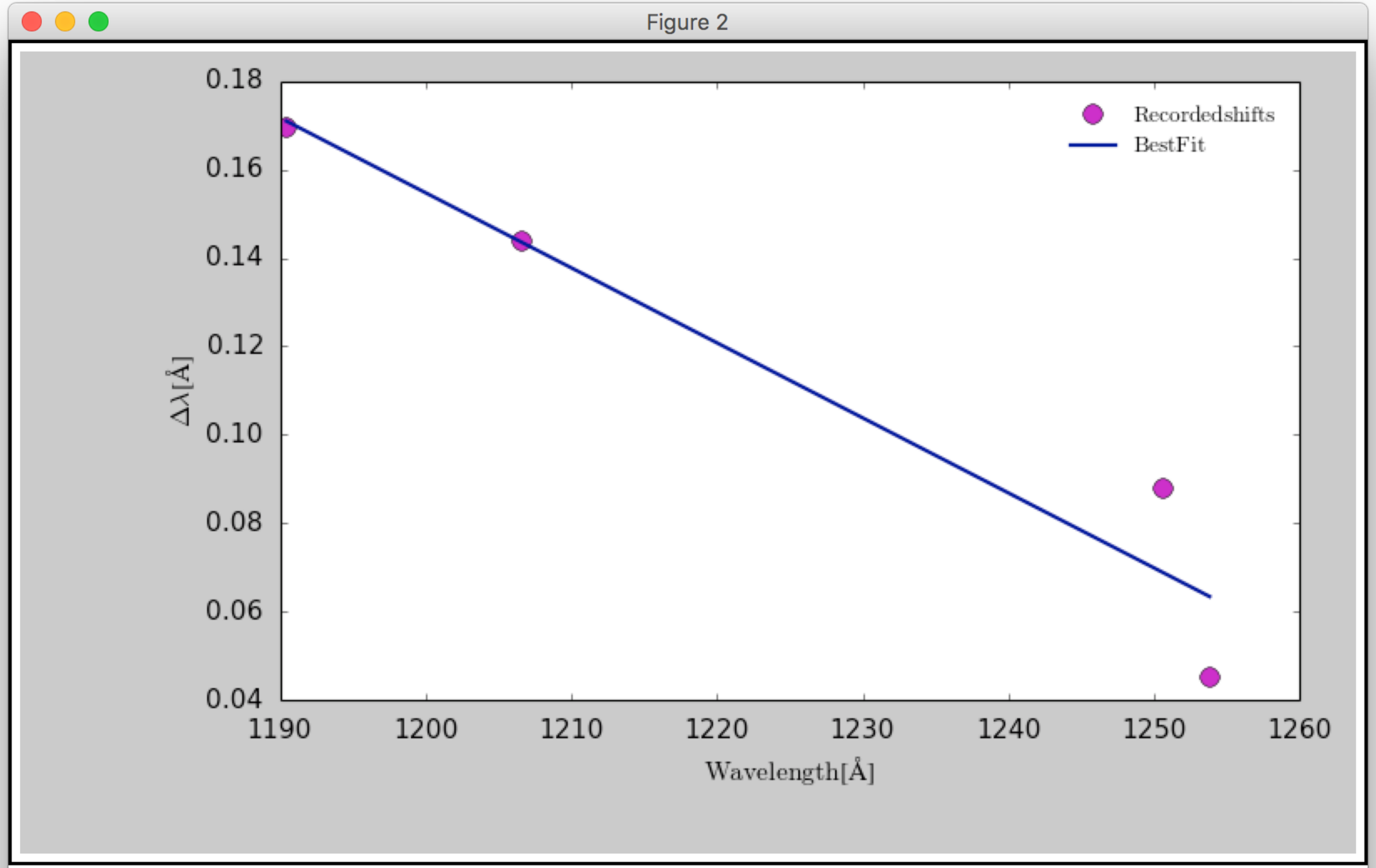
4. Absolute Calibration

A new window will appear. For a key map, press “?” when focusing on the plotting window.Press “a” to add two points then “shift+g” to fit a gaussian. One can repeat the gaussian fitting process by simply choosing different spectral region to add two points with “a” and then “shift+g” again. The fitted centroid is automatically recorded. After “shift+g”, user can close the plotting window.



4. Absolute Calibration

During and after fitting gaussian to Milky-Way lines, user can check if there are any outliers or the best polynomial fit for the current offsets recorded. Press “p” for printing current recorded gaussian fits, and choose the order of polynomial fit for the data (see example of linear fit below). User can delete any of the points. If user decides to write a new spectrum (after inspecting the fit), press “w” to write and choose the same polynomial order. Then repeat the process for the spectrum of the grating.



In principle, that's it. There will be two spectra (G130M and G160M grating). If one desire to combines the two, use Coaddition.py again to combine.