The Spectral Energy Distribution Code

A code has been developed for the display and fitting of stellar spectral energy distributions (generally speaking). This document describes the code and how it can be used. The code is currently still under development.

The code currently consists of the following python files:

```
extinction_code.py
model_utilities.py
position_plot.py
read_vizier_sed_table.py
sed_plot_interface.py
sed_utilities.py
tkinter_utilities.py
```

The main code is the `sed_plot_interface.py` code. It uses `TKinter` for window management. Also required are `matplotlib`, `numpy`, and `astropy` python packages. Other standard python packages such as `bisect`, `math`, and `sys` are used in the code. The code is written in Python 3 and will not work in Python 2.

The code was developed on a Macintosh system. This should not affect the functionality of the code if it is used on another operating system, but the look of the windows will vary in other operating systems. It has also been tested in a Linux system, but not on a Microsoft windows system.

Starting the Interface

The code is started by running the `sed_plot_interface.py` code from the command line. Figure 1 below shows the initial appearance of the window. There is a plot area at right and controls at left. Above the plot area is a text field that shows plot information, which starts out with no information as nothing has been done yet. When the cursor is in the plot window the position is updated each time the cursor moves. What information is in the text field varies somewhat depending on the state of the code.

The code was designed primarily for the display and analysis of the Vizier photometry service output files. The Vizier photometry service is available from the SIMBAD web pages for individual objects, or can be called separately. The aim of the Vizier photometry service is to return all photometric measurements in the literature tracked by Vizier for a given source, based on a position search with a given radius of the nominal object position. Figure 2 shows an example of a SIMBAD search result from http://simbad.u-strasbg.fr/simbad/ using the search by identifier option with the name “HD 56126”. The object information is displayed along with a small image of the immediate sky field at right and the Vizier photometry viewer available below the sky image at left. If one enters a search radius, for example 5′′, and activates the button one gets a new tab in the browser with the display shown in Figure 3. (The Vizier photometry viewer service can also be accessed directly via http://vizier.u-strasbg.fr/vizier/sed/ and used with coordinates. The advantage of going through SIMBAD is simply that the

coordinates are supplied. One can also call the service via the command line with the `wget` command and have the results written to a file.)

The Vizier photometry viewer service is useful but there is also some functionality that is not included. It is to add some additional functions that the current tool was written. If one saves the Vizier photometry from the page to a "vot" file with the "download data" button shown near the bottom edge of Figure 3, that type of file is the input for the current code. The tool is intended to allow some options in the display of the data, as well as to be able to read in other data for the same object and make some calculations on the data. Another type of functionality is to allow one to fit stellar models to the data to derive object properties.

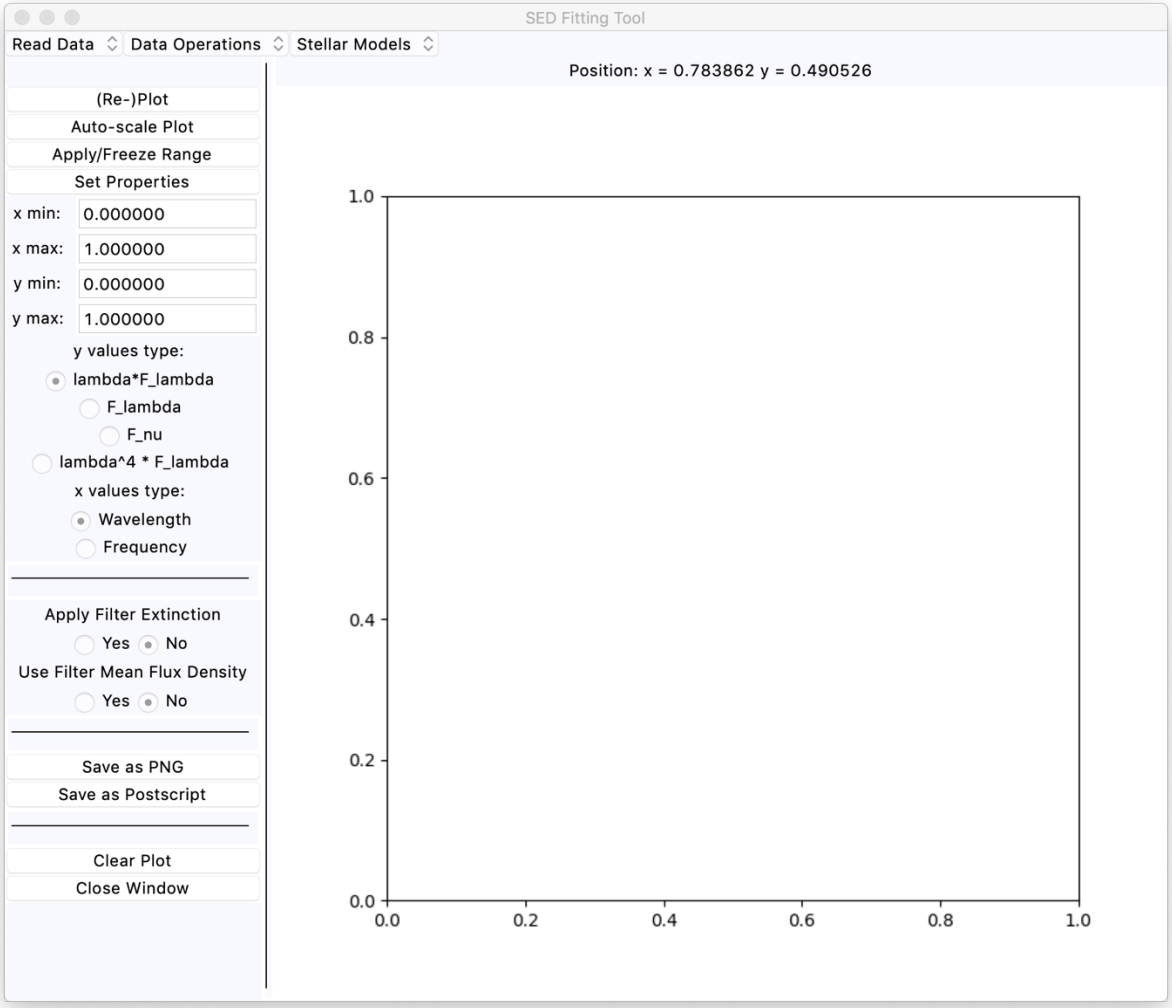

Figure 1: The initial appearance of the main SED fitting window.

A look at the spectral energy distribution plot in Figure 3 shows that some of the points are anomalous. This is quite normal, as the listing is solely position based and there is no certainty that all the data values are for the same object. This is noted explicitly in the Vizier photometry service documentation. In this particular case, one obviously bad point is the single photometry point at about wavelength 12 microns and λF_λ about 5.54 10^{-14} W/m^2. That point is an

extrapolation of the stellar flux density at shorter wavelengths to the IRAS 12 μm filter, and is not an observed photometry value. One can determine this by using the Vizier photometry viewer page to track the source of the photometry. There are also other points that are somewhat off the main spectral energy distribution (hereafter, SED). For some filters, such as the 2MASS J, H, and K_S filters for example, there are multiple entries from different papers with a range of flux density values and sometimes a range of wavelength values as well. Some of the photometry values have uncertainties and others do not. In general it may be difficult to determine which photometry values are most reliable, or even which apply to the object of interest.

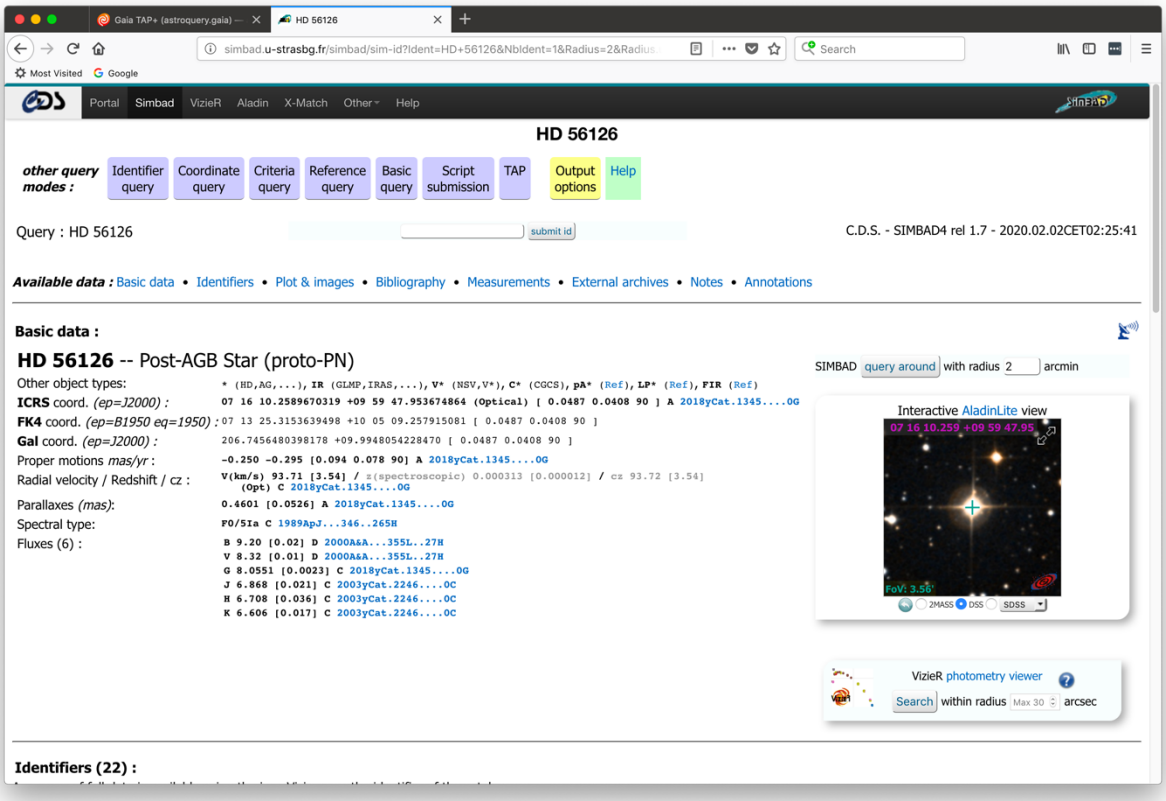

Figure 2: A screen shot of the results of a SIMBAD search for object HD 56126. The Vizier photometry viewer service is available at lower right.

The position plot at right in the Vizier photometry viewer web page is sometimes useful for separating which points are for the object of interest and which are for other objects nearby, but even this is difficult to assess because the positional uncertainties are a strong function of wavelength and may vary between ground-based and space-based sources of photometry. The longer wavelength 2MASS and WISE photometry, while commonly available for brighter objects, may show significant positional discrepancies, possibly due to proper motion of the object but also in part due to the intrinsic positional uncertainties of the surveys.

Due to these uncertainties and the multiplicity of data values, two important goals of the current tool are to allow the user to remove discrepant points from the SED plot, and to allow some type of averaging of the photometry values where there are multiple entries for the same

filter name. Another goal is to allow a few different "views" of the SED information as well as to allow adding other types of data not included in the Vizier photometry service, particularly spectral data, to the plot.

Basic Usage

For illustration purposes the Vizier photometry output file from the HD 56126 search shown in the Figures will be used as the example. The information was saved to a file named `vizier_votable_hd_56126_radius_5p0.vot`.

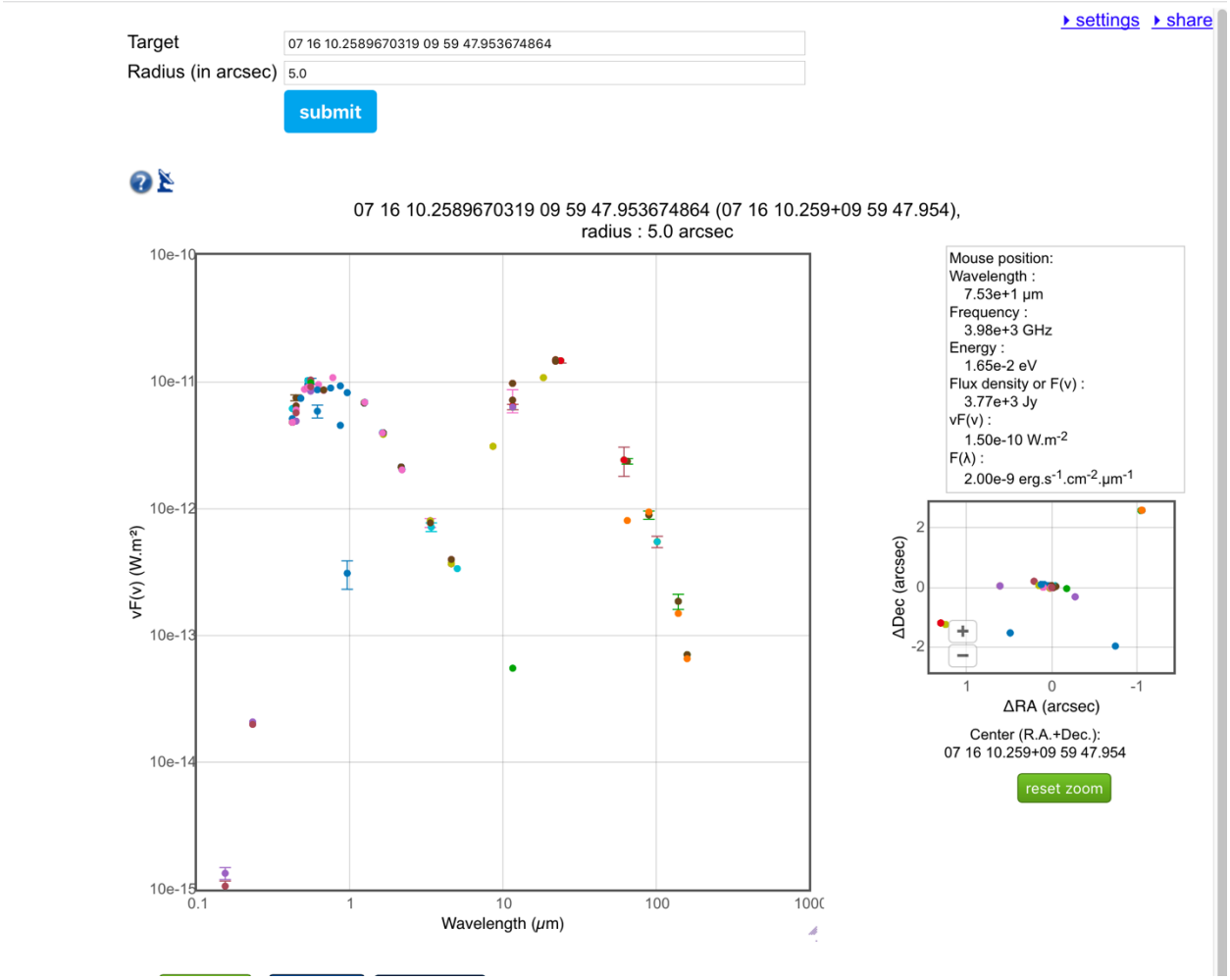

Figure 3: An example of the Vizier photometry service web display for object HD 56126 with a search radius of 5.0″.

Having started the code. one needs to read in data values for an SED plot. Under the "Read Data" menu at upper left in the window one currently has two menu items, "Read Vizier SED File" and "Read Spectrum File". Selecting the "Read Vizier SED File" brings up a standard Tkinter file selection dialog. This assumes that the Vizier file is stored as a virtual observatory table file with a suffix of `.vot`. If one then navigates to the directory where the file

`vizier_votable_hd_56126_radius_5p0.vot` is stored and selects this file, the code then reads in the file and parses the data. It then will automatically plot the values in a way similar to what is shown in Figure 3 above. The initial plot is shown in Figure 4 below. The points correspond to the values in Figure 3 above, but the points are all of the same colour and symbol type rather than having symbols that vary according to the reference from which the values are obtained. The default units are to plot the wavelength in μm and the λF_λ value in units of Watts/meter^2. When the cursor is in the window the position values above the plot are updated, and now that there is data information about the "nearest" point to the cursor position is also given.

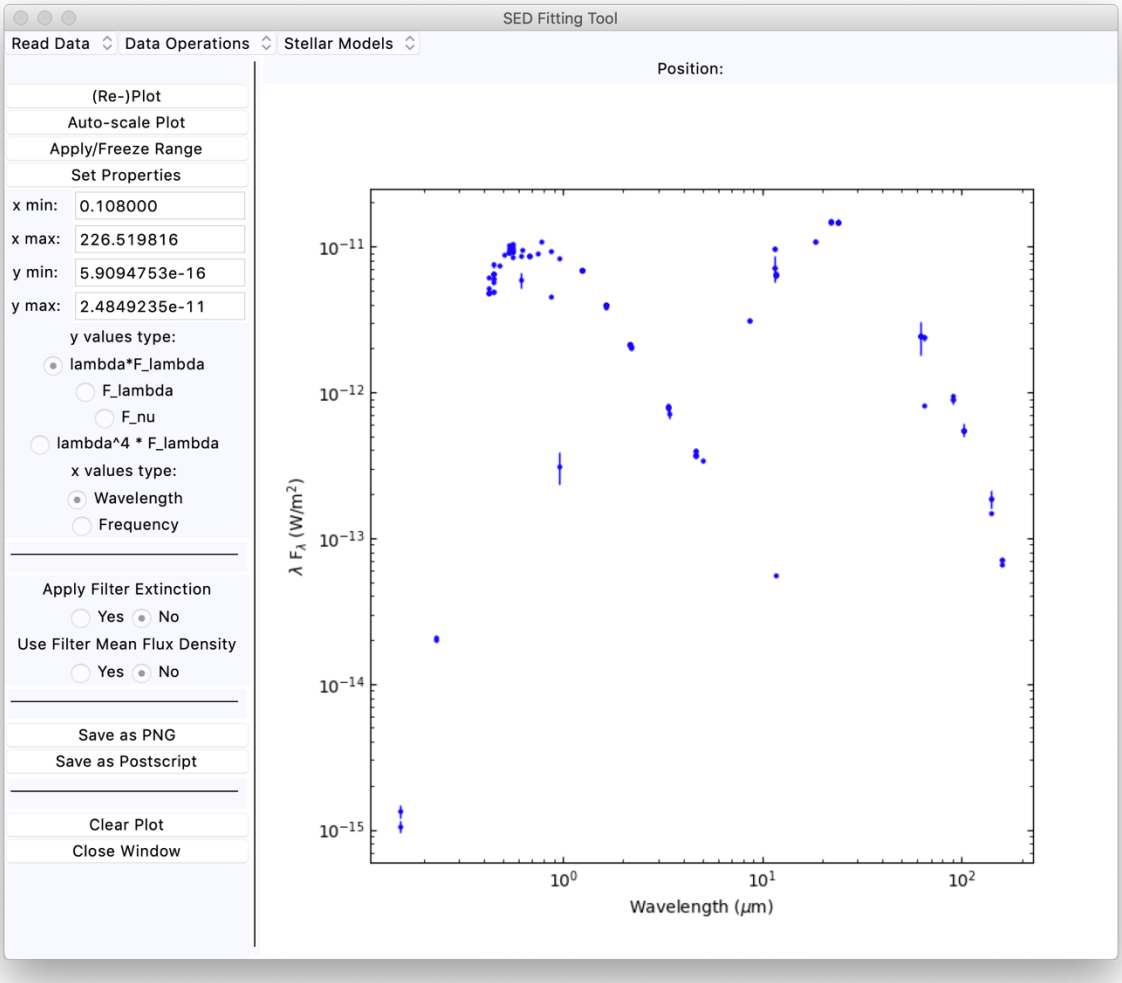

Figure 4: The SED window with the HD 56126 Vizier photometry data for a search radius of 5.0′′.

The determination of the nearest point is not done with the usual distance measurement because the display is using a log-log plot wherein the magnitude of the y axis values are often many orders of magnitude smaller than the magnitude of the x axis values. In such a case one

may get misleading results from the regular distance between points (x_0,y_0) and (x_1,y_1) defined via the equation

$$d = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$$

so instead the "distance" is defined as

$$d = \sqrt{\left(\frac{x_1}{x_0} - 1\right)^2 + \left(\frac{y_1}{y_0} - 1\right)^2}$$

to minimize the relative ratio "distance". This allows one to find the points in the plot when the flux density values are small compared to the wavelength values.

The above "distance" formula is not the same as finding the distance in plot pixel units and finding the closest point. While that could be done, it was deemed too expensive in computation for cursor motion update, as it would require transforming the data values to image pixel coordinates rather than working directly with the data values.

One can change the x-axis units or the y-axis units using the radio button controls at left. One has the choice of wavelength in μm or frequency in GHz for the x axis values. One can select λF_λ, F_λ, F_ν, or $\lambda^4 F_\lambda$ for the y axis values. If one selects frequency the x axis is inverted so that the higher frequency points are at left. Whenever the units are changed with the radio buttons the plot is regenerated, as in Figure 5. Plotting $\lambda^4 F_\lambda$ minimizes the flux density variation with wavelength, as a blackbody spectrum on the long wavelength side of the F_λ peak will be nearly flat in such a plot. As most stars have a spectral shape that is approximately like a blackbody at long wavelengths, their spectra are also flattened out in this type of plot.

The plot limits are put into the entry fields at left when the plot is generated. One can edit these values and apply the desired range with the "Apply/Freeze Range" button at left. Normally the view auto-scales each time the plot is regenerated. The `matplotlib` rounding of the limits does not generally produce nice axis limits for the log-log plot as in the examples above. If one wishes to save the SED plot with the limits being at minor tick marks, one needs to edit the values in the range fields and then use the "Apply/Freeze Range" button to apply these values. Any time that one clicks on the "Auto-scale Plot" button the automatic plot limits will be applied to the plot.

Deleting/Undeleting Data Points

Usually the next thing to do will be to remove bad values from the SED plot. Under the "Data Operations" menu one has a menu selection to do this. The first menu selection is "Delete/Undelete Data Points". When this menu item is selected it sets an internal flag such that when the cursor is in the plot area and a nearest point is identified then this point can be removed from the plot or returned to the plot via a mouse button click release. One can also use the "d" and "u" keys to delete or undelete the nearest point, in case for some reason clicking a mouse button is not possible. The program accepts these key or button commands until this function is exited with either the "x" key or the "q" key. When the delete/undelete function is active, there is notification in the position field to this effect. Also, the position field notes whether the nearest point is masked or not.

As an example, Figure 6 shows the main window appearance after several of the low HD 56126 photometry points and one higher than expected point have been removed from the

window. The cursor was near the extrapolated IRAS 12 μm point that is anomalous when the image was taken for the Figure. One can see the note "[delete/undelete active]" in the first line of the position field. When the cursor is near a masked point the notation for the nearest point includes a "(masked)" notation. After this function is exited with the "q" or "x" key, one has to move the cursor at least a little to cause the position field to update. Then the note about the delete/undelete function being active will disappear.

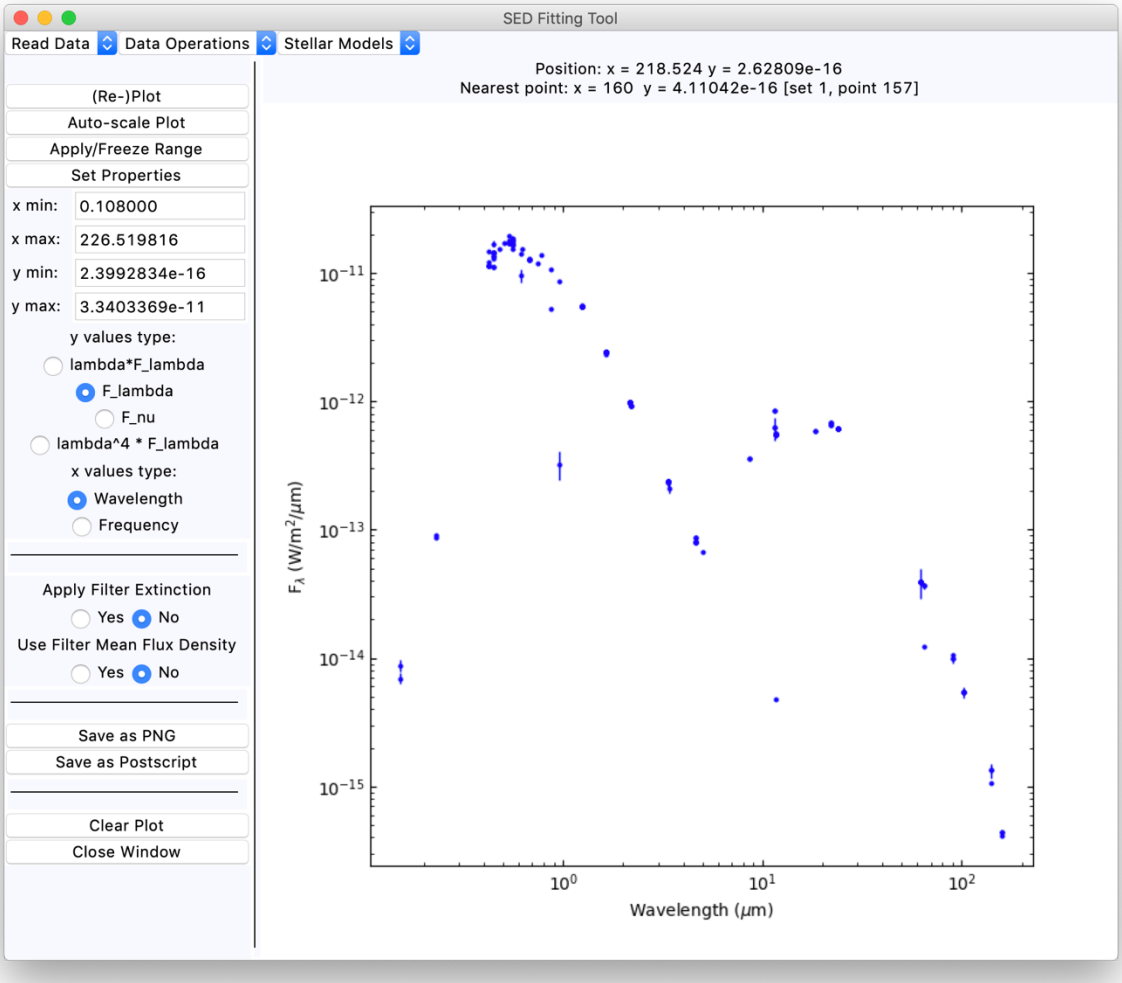

Figure 5: The SED window plot for HD 56126 when the y axis is plotted as F_λ.

As mentioned previously, photometry points may be found multiple times in the Vizier photometry table if the same measurement is given in different papers in the literature. When a point is masked, the code masks any duplicate points with the same wavelength and flux density values. The code does not attempt to delete photometry values by the filter name in the Vizier table, because one may find several slightly different values for any individual filter from the literature. Hence where duplicate filter points exist with different flux densities one can choose between them by deleting some values and not others. It is often the case that even non-variable stars will show different flux densities listed for the same filter, because of different assumptions

about the absolute calibration, in the data reduction process such as in background subtraction and aperture size, or due to rounding of values (more usually the first two rather than the latter).

As it is in general very difficult to get full coverage of the object SED with a fixed aperture size in the Vizier photometry search, one will generally need to exercise judgment about which points to delete and which to retain. There is no general algorithm that can be used for such a purpose. When in doubt, the user will need to look at the detailed catalogue values or the original papers wherein the photometry values are listed.

When points are masked, they are not included in any calculations carried out with the data points. That is why the operation is called deleting/undeleting points even though in actuality the data points remain in the internal arrays used by the code.

One can restore the original set of points with no masking using the "Unmask All Points" menu item. If there are multiple data sets in the plot, this operation is applied to all the data sets. There is no current function to unmask all the points for just one data set in the case where several data sets are being plotted (which has not been discussed as yet).

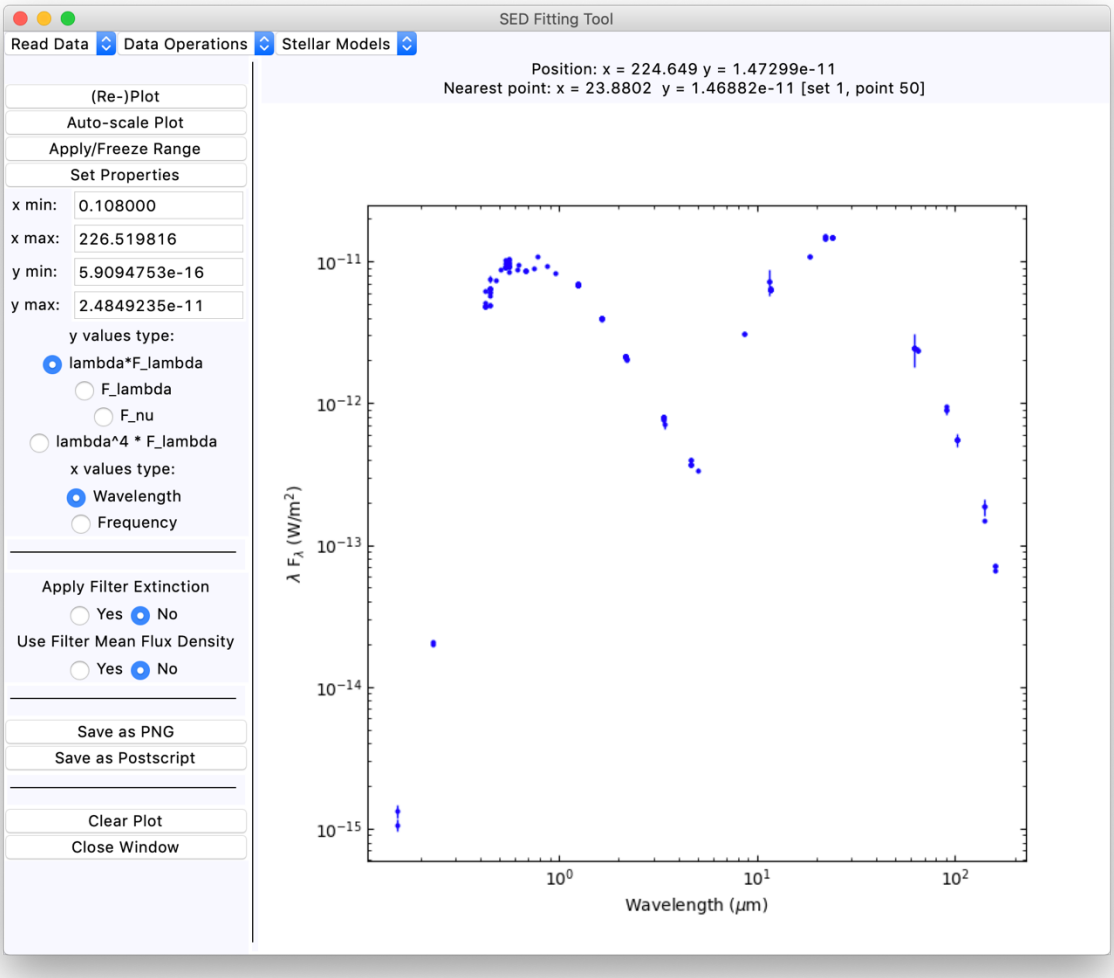

Figure 6: An example of the SED plot for HD 56126 with some of the photometry points masked.

Data Averaging

The Vizier photometry service returns a list of all photometry values listed in the catalogues and literature tracked by Vizier. One may have multiple values of the same photometry listed from different sources, for example with more well-studied sources. As these values for the same filter photometry do not always agree, a function is provided to average the values for each distinct filter listed. The averaging is a straight mean with no consideration of uncertainties, because in many cases no uncertainty is given for the photometry. This produces a new data set with the mean values as a function of wavelength that is re-plotted along with the original data. This function is carried out with the "Average Vizier Data" menu option. The new set of data values is plotted with a different colour and with dashed lines between the points. Carrying out this calculation for the HD 56126 data shown in Figure 6 produces the plot shown in Figure 7. The mean values are now data set number 2, and the original values are data set number 1 (the set numbers internally start at 0 as normal in python, but when providing the value they are numbered starting at 1).

One can delete points in the averaged SED data set if one wants to do so, in the same way as for the original Vizier photometry service data values. One may need to zoom into the plot to isolate these points from the original data points, as will be discussed later. In general, one is less likely to want to mask averaged data points.

Position Plot for the Points

In the Vizier SED tool there is a function to provide a position plot of the points for a source. A similar, although more limited, function is available in this tool. Under the "Data Operations" menu one has the "Show Position Offsets" window. If there is Vizier SED data in the tool, providing positions for each photometry point, a plot can be made. In such a case the code brings up a new window with the offsets from the reference position, either the GAIA DR2 position or the mean position of all the photometry. An example of how this may look is shown in Figure 8 below.

Once the window is up one can find points with the cursor and see how far the nearest point is from the cursor and from the centre position, along with which filter provided the photometry. Note that the masked points are not plotted in the position window. If one changes the masking, one has to close and re-open the position window to get an update.

In the case where only one Vizier SED data set is present in the tool that set will be automatically selected for the plot. Where there are two or more Vizier SED data sets present a pop-up window will appear asking the user for the set number to use in the position plot. As usual in this code numbering starts at 1 rather than 0.

Calculating the Flux of a Source

The code allows one to integrate over the SED to calculate the total flux from the source, and then to use that to calculate a luminosity value if the distance to the object is known. This functionality is invoked with the "Calculate Flux" menu item. When this function is activated, a new window appears which allows the calculation.

The flux calculation is carried out by a simple trapezoidal rule integration over the wavelength flux density values, equivalent to linear interpolation between the data points of the

spectrum. This is done with all unmasked points in the data set under consideration. The first data set is used when the flux calculation window first comes up. The source is assumed to be at a distance of 1 kpc for the initial calculation of the luminosity. Once the window is open, one can select a different data set as well as setting the object distance or parallax to allow the luminosity to be calculated. If one wishes to correct for extinction, the de-reddening functions can be applied to observed data (as will be discussed below) and the flux calculated from the de-reddened values.

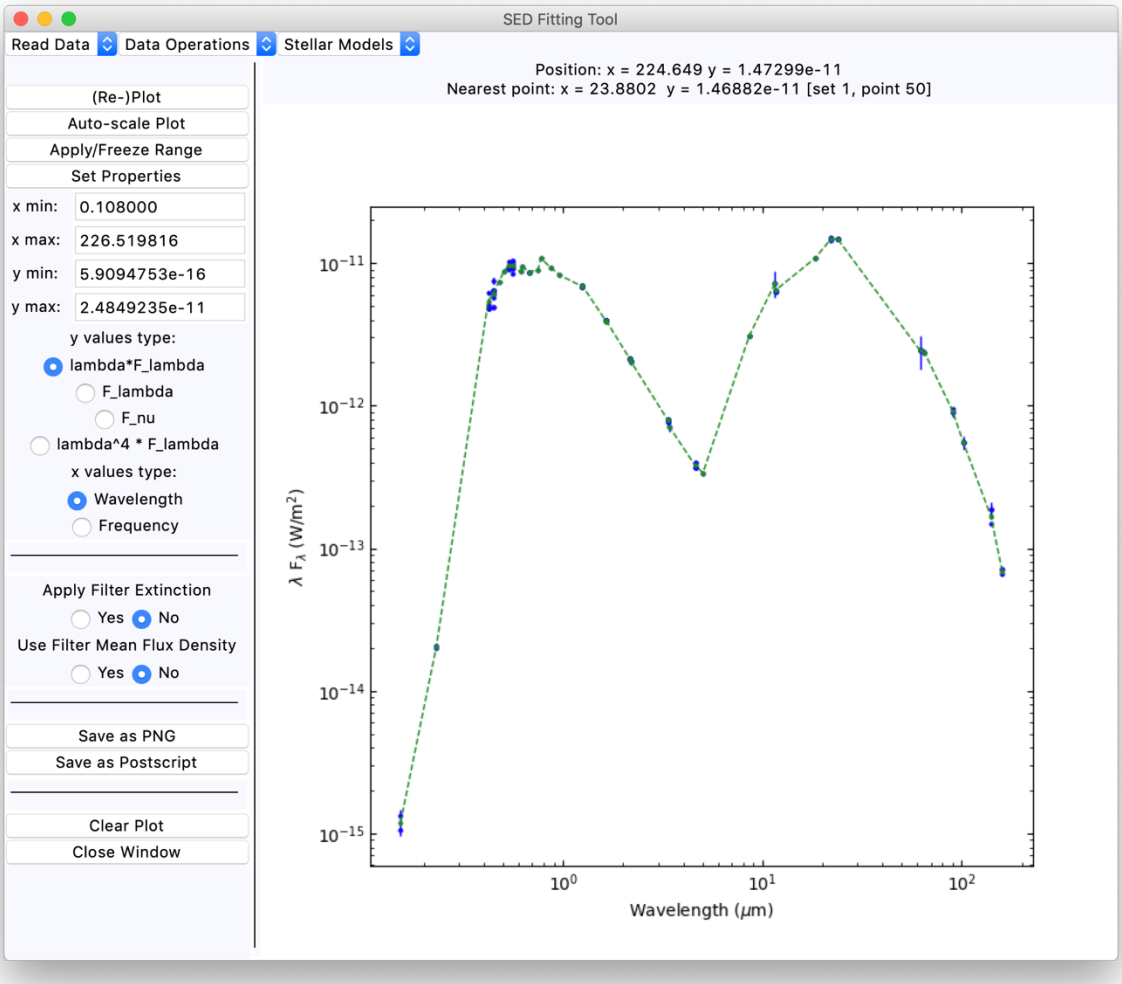

Figure 7: The HD 56126 SED plot with the mean values added (green points).

Figure 9 shows the initial appearance of the window when the Vizier photometry data for HD 56126 is plotted. The window has a text area at top and a control area below. The data set number to use for the calculation is set by the first menu at right below the text area. One can provide a parallax value in milli-arc-seconds or a distance value in kiloparsec in the entry window in the control area. An uncertainty value for the distance or parallax can also be given. The radio button in the fourth row of the control area allows one to select whether the values in

the entry areas are interpreted as parallax values or distance values. The GAIA data release 2 parallax value for HD 56126 is 0.4601±0.0526 milli-arc-seconds (hereafter, mas). Putting these values into the field and doing the calculation again with the "Recalculate" button gives the same flux value but a revised luminosity estimate. The calculation can be done with the averaged Vizier photometry values as well. These data values given a slightly different estimated flux. The output is accumulated in the text window as seen in Figure 10.

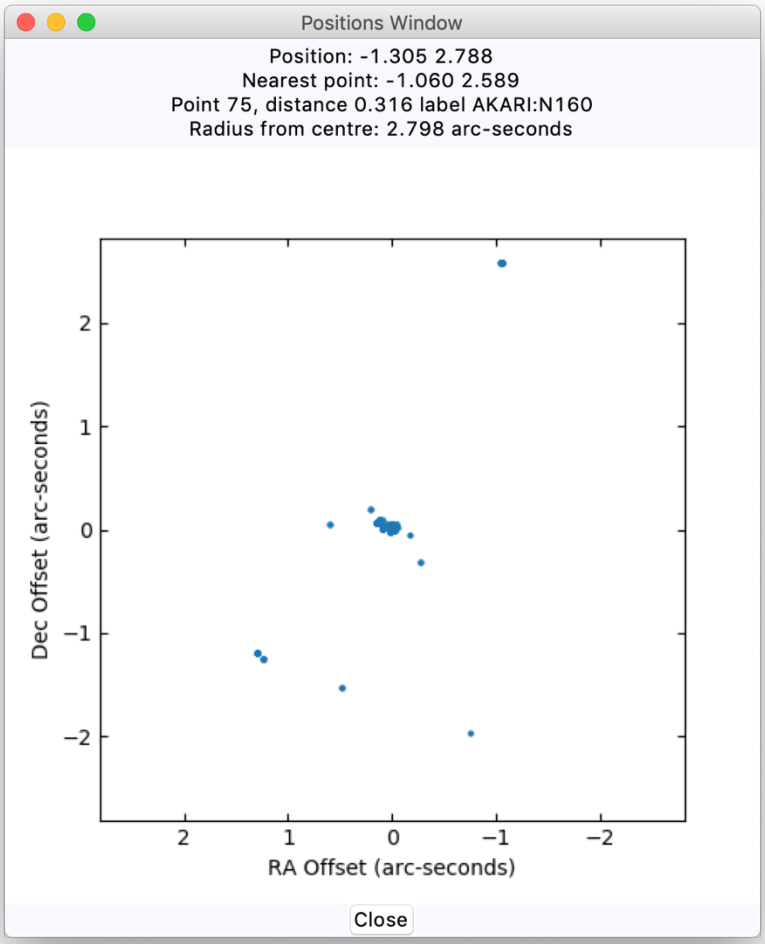

Figure 8: The position window provided by the tool. The right ascension and declination offsets are plotted in x and y, with the x axis inverted to put east at left.

When an uncertainty in the distance is given, the luminosity values include a range matching the distance or parallax uncertainty. The values are given with a fixed format that in this case gives too many significant figures. It is up to the user to round the values off as needed.

The luminosity range values given do not include uncertainties in the flux. It is difficult to determine the flux uncertainties in the case where the input photometry values do not have listed uncertainties. The flux uncertainties are going to be a least 2% for most types of input data from either optical or infrared photometry. As these wavelengths usually dominate the SED, accuracies of better than 2% are not obtainable in most cases. Further uncertainties come from

the linear interpolation used to get the flux and possible discrepancies between independent data sets, especially when these combine both ground-based (i.e. 2MASS or SDSS) and space-based (i.e. HST and GAIA) photometry. For objects with very accurate parallax values the uncertainties in the luminosity will be dominated by the uncertainties in the flux, but this is only the case for a small number of very nearby stars. In general, the uncertainties from the distances remain rather larger than the uncertainties from the flux calculation, assuming that any extinction correction that is needed has been applied properly to the input data.

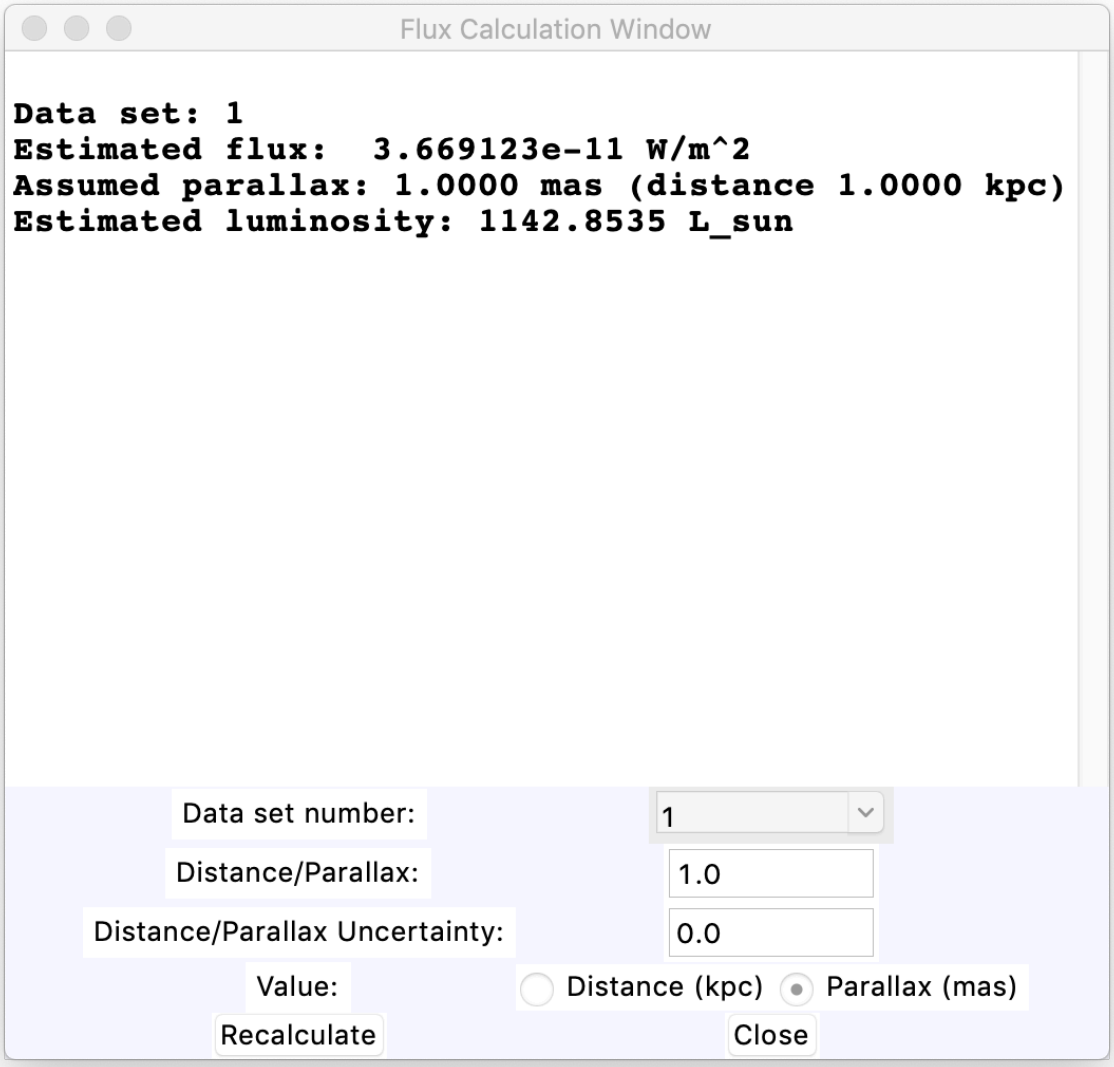

Figure 9: The flux calculation window as it initially appears for the HD 56126 data.

Reading in a Spectrum

One can also read in data values from an ascii spectrum file or FITS table file for the plot. In the ascii case input file needs to have one column of wavelengths and one column of flux

densities in the units that the code is able to handle. There may be other columns present that are not read as long as the column structure is uniform. The data values are read in via the `numpy.loadtxt` function. Columns are assumed to be separated by one or more spaces. Any lines with #, |, or \ symbols at the start are ignored. In the FITS table case, one needs to know the table heading values for the wavelength and the flux density. Again the units need to be in units that the code can convert.

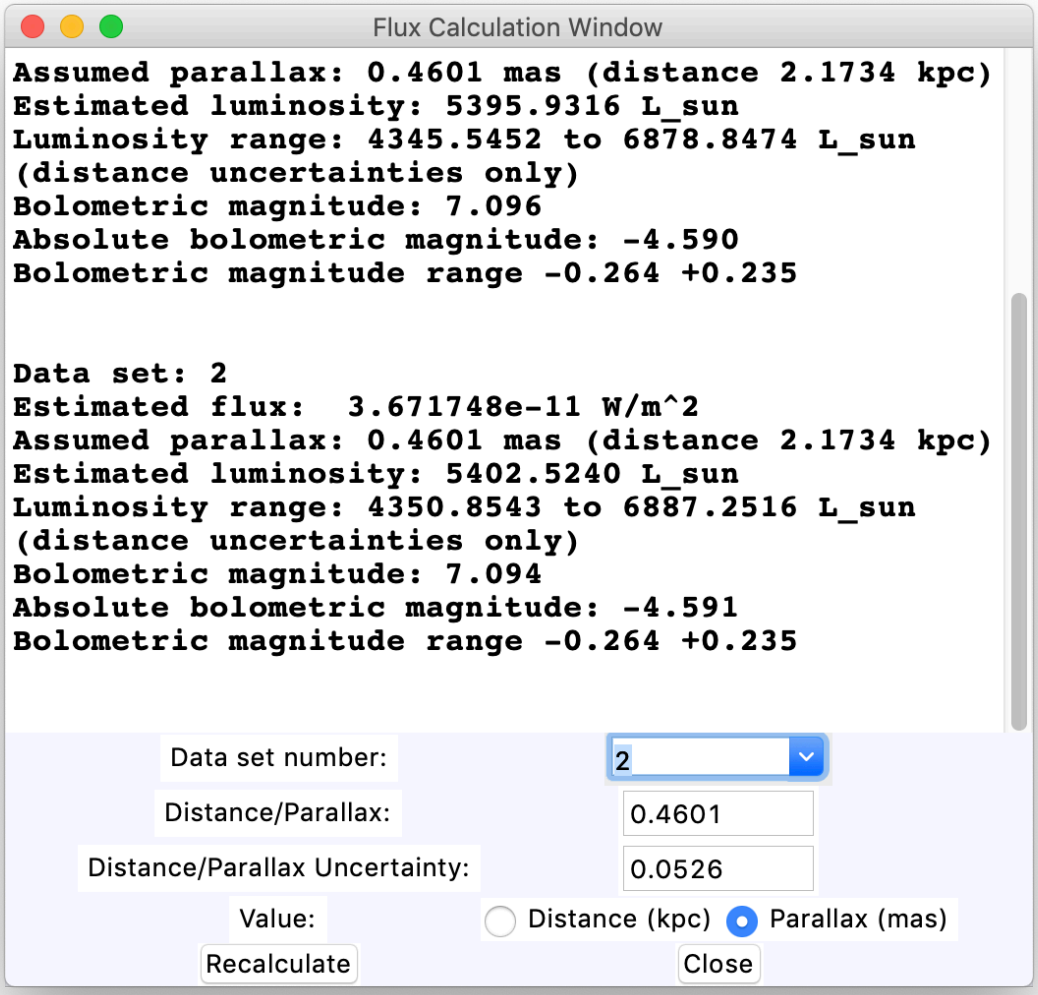

Figure 10: The results of the HD 56126 flux calculations with the GAIA DR2 parallax values.

This function is invoked via the "Read spectrum file" option in the "Read Data" menu. A new window is created with a button to select a file and a set of parameter fields. The initial appearance of the window is shown in Figure 11. Initially there is no file name listed and the parameters have default values. One needs to tell the code which columns have the x and y data, with column numbers starting at 1, and the units of the values. Only a small sub-set of possible units are supported. The x data points need to be one of the following: wavelengths in μm or

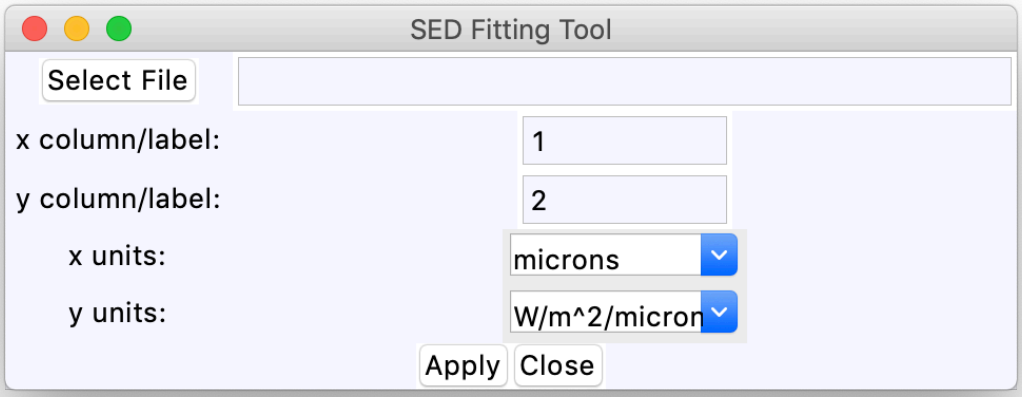

Figure 11: The initial appearance of the spectrum read-in window.

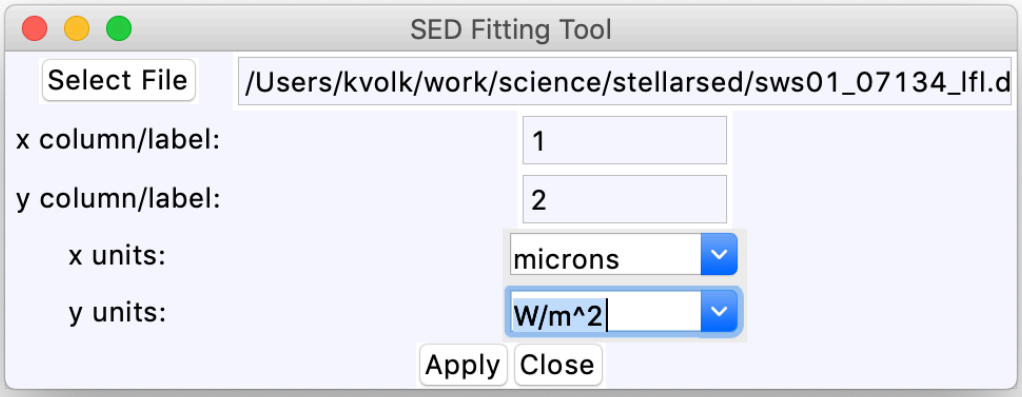

Figure 12: The spectrum read-in window with selections for an ISO spectrum for HD 56126.

Angstrom, or frequency in GHz or Hz. The y data points need to be one of the following: F_λ values in Watt/meter^2/μm or in erg/second/centimeter^2/Angstrom, F_ν values in either Watt/meter^2/Hz or Jansky, or λF_λ in units of Watt/meter^2. Only these limited sets of input values will be handled properly by the code. One can for example read in an Infrared Space Observatory (ISO) spectrum for HD 56156 from the file `sws01_07134_lfl.dat`. This file has wavelength values in μm and λF_λ values in units W/m^2 in the two columns. Figure 12 shows the spectrum reading window with the y units selection set for this file and the file name entered in the file entry field, assuming that the file is in the current directory from which the code is run. One can then press the "Apply" button to read in the spectrum. If the spectrum is read in

successfully, the SED plot appears as in Figure 13. The spectrum is shown as the green line—this example does not have the mean Vizier values as in some of the previous Figures. The spectrum is noisy and includes negative values in the wavelength range between about 4 and 6.5 μm that therefore are not shown in the log-log plot.

When a spectrum has been added to the plot one can integrate the flux density to get the flux for this part of the SED as with the Vizier photometry values. The integration is over the entire wavelength range of the spectrum. The integration uses the input values whether they are positive or negative. If the spectrum data are bad, the results will not be useful. The code does not check for negative flux density or flux values in making the flux and luminosity calculations.

The only difference in the case of a FITS table is that one needs to enter the label strings in the entry fields for "x column/label" and "y column/label". These might be "wavelength" and "flux" for example. If the file name ends in ".fits" the FITS table code is selected, otherwise the ascii file code is selected. The code is not able to look at a FITS table header and determine which column labels correspond to wavelengths and flux density values.

Changing the Data Set Point Properties

The code provides a limited capability to change the symbols used for the different data sets in the plot. This is accessed via the "Set Properties" button at left in the main window. A new window comes up that allows one to set some of the symbol and line properties for each individual data set on the plot. The window appearance is shown in Figure 14. One first needs to select which data set is being considered from the "Data Set Number" menu. The sets are numbers starting at 1 as usual, rather than being numbered from 0 as in the internal python code. One can set whether there is a line connecting the points, the symbol type, the line type, the colour, the line thickness, and the symbol size from the menus or entry fields. Once values are set the "Apply" button causes these values to be used in the main plot area. One can make a series of changes as needed. Each time the "Apply" button is activated the plot is re-drawn. The window is closed with the "Close" button at lower right.

The menu of symbols includes all the normal matplotlib symbols, referenced by name. The normal line types are also all included in the menu for that parameter. The colour selection is more limited. There are 12 colours available in total. If one wishes to add more colours to the menu these are stored in the `COLOUR_SET` global variable defined at the top of the `sed_plot_interface.py` file.

Plotting By Catalogue/Mission

The on-line Vizier SED plot has the data points grouped by the bibliographic source of the data values. In this tool that is not done. Rather, the code has an option to plot different Vizier data sets by the catalogue or mission from which the data are gathered. What is done is to take the filter labels in the Vizier data set and extract the first part of the label. This is the catalogue name or abbreviation, as with "2MASS" or "GAIA/DR2", or the mission/satellite name, as with "AKARI" or "DIRBE", or the instrument name, as with "ALPHAMBRA". The data points for each distinct sub-set of points can be plotted in a different colour. The flag for this is toggled by the "Toggle Vizier Filter Colours" menu item under "Data Operations". There are some limitations to this approach. For example, all the Johnson filters from UBV through the infrared are grouped together in this type of plot, irrespective of where the data values come from. This

may not always be what is needed since there are significant differences in the degree of standardization of the Johnson optical filters and the Johnson near-infrared and mid-infrared filters.

To aid in differentiating the different Vizier SED filter types, when the data set of the point nearest the cursor is a Vizier SED data set then the filter name is shown in the text area above the plot. For other types of data this is not put into the information field.

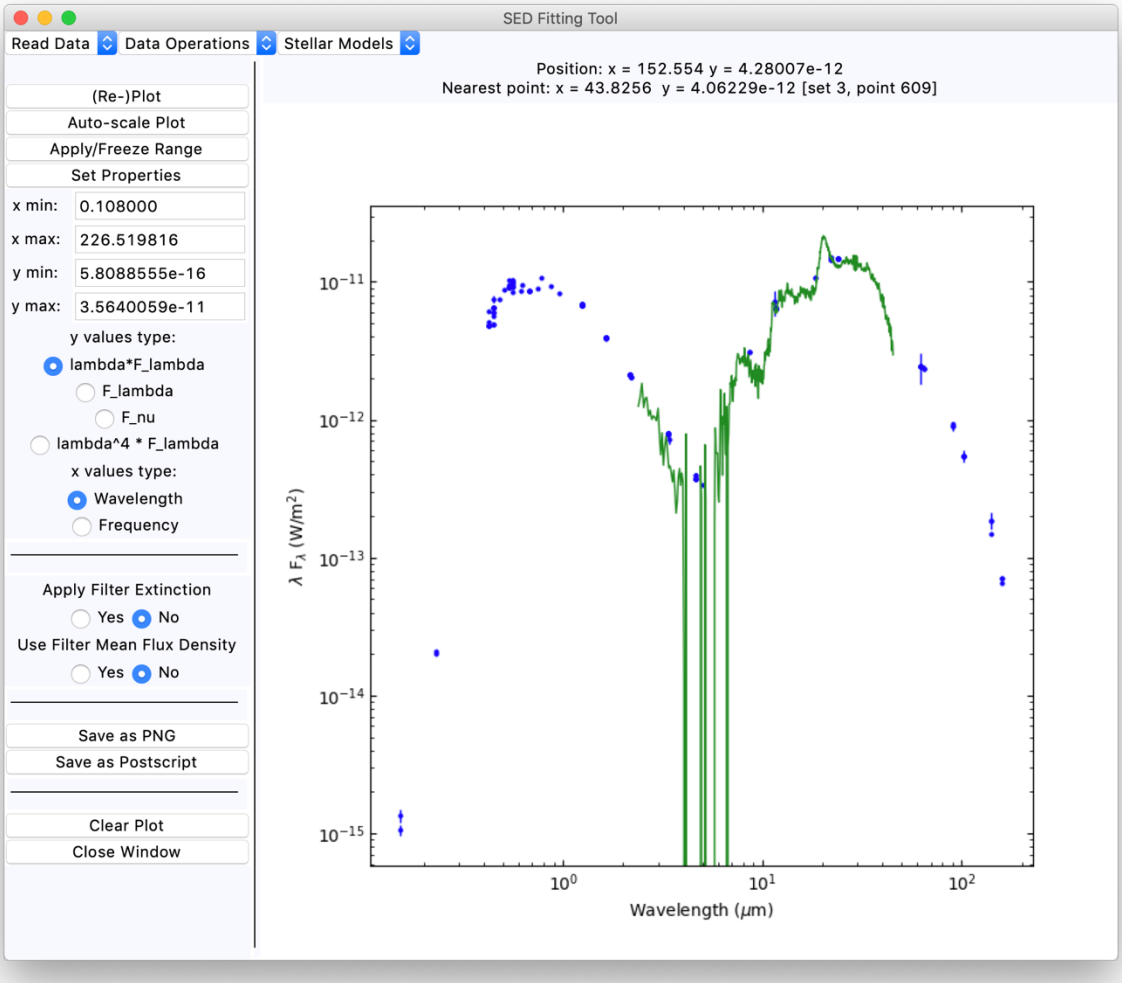

Figure 13: The SED plot for HD 56126 with the ISO spectrum.

If this menu item is selected, a pop-up window appears asking which sets are to have the flag toggled. One can enter a set number or enter "all" to toggle all the flags. If the flag is toggled to True then instead of a single colour for all the data points in a Vizier SED data set the code takes as many colours or symbols as needed from pre-defined lists to separate the different data sources that are listed. The user does not have control over these colours or symbols, When one toggles the flag back to False the colour and symbol selections revert to the normal values which can be set by the user. Note that the code assumes that the points when coloured are not connected by lines, and that this option can only be applied to the Vizier SED data sets. One can toggle the flag for other data sets, but it will have no affect on the plot. Figure 15 below shows

what the HD 56126 SED plot looks like with the filter colour option set.There are 24 colours and 20 symbols defined for use with this option. One should never exceed this combined number, but if so the colours/symbols are reused with a larger symbol size.

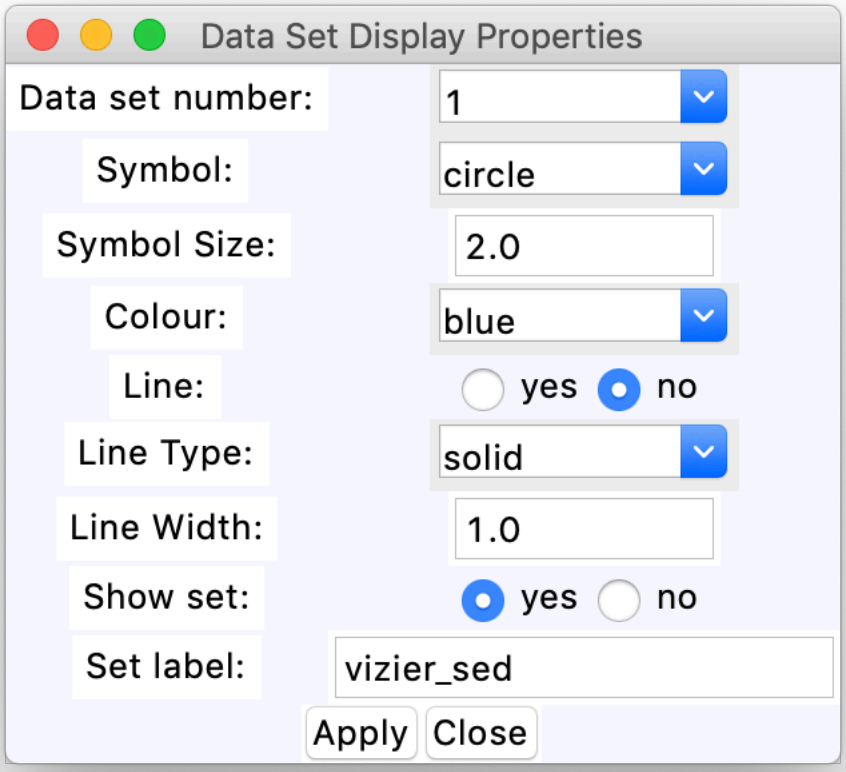

Figure 14: The set properties window.

In the Vizier SED files any assumed monochromatic points are not assigned a filter name, so the first element of the name string is mission. In the code these points are grouped together as "monochromatic" and have a common colour when this option is used.

Stellar Models

The code allows the user to read in a few types of stellar atmosphere model files for comparison with observed data. Once input the user can redden a spectral model for comparison with data or calculate the stellar flux from the model. The extinction part of the code will be discussed in the next section. The "Stellar Models" menu allows one to read in a stellar model and apply a scaling to it. When one selects the "Read Stellar Model" menu item a window comes up for the user to specify the file name to be read in and the model type. Figure 16 shows

how the window looks when initially brought up. One uses the "Select File" button to navigate to the file of interest. One needs to specify the model type with the radio buttons. Then clicking on "Apply" will direct the code to read in the model. An information message is posted to the text area at the top of the window noting whether this succeeded or not. Note that getting a file name from the "Select File" button does not automatically cause the code to try to read the file. That only happens when the "Select" button is activated.

For some of the input models one will generally wish to smooth the model from the original resolution to something more useful in the plots. The slider at the bottom of the window is provided for this purpose. Resolutions from 100 to 3000 in the slider are applied by the code. If the slider is at either end of its range, 90 or 3010, then the smoothing is not applied. The default is to not have any smoothing. Currently the smoothing is only applied to the Phoenix models.

There are a number of different stellar models available and the code does not allow all of these to be used. There are four types of models that the code can handle, as will now be described.

The Bohlin et al. (2017) "BOSZ" Models

The models are described in the reference Bohlin, R. C., Mészáros, S., Fleming, S. W., et al., 2017, AJ, 153, 234. A brief description is given at https://archive.stsci.edu/prepds/bosz/, and the model files are available from https://archive.stsci.edu/hlsp/bosz/search.php in either FITS table or ascii form. These are a very large set of stellar atmosphere models using the Kurucz Atlas9 code. The individual models cover from 0.1 to 32 μm. At the MAST site one can find these models at a range of instrumental broadening values from 200 to 300000. For the purposes here the lower resolution versions of these models should be used. As a number of filters at wavelengths beyond 30 μm are included in the Vizier SED plots, the code extrapolates the BOSZ models out to a wavelength of 1000 μm using a simple power-law of wavelength fit to points near the long wavelength limit of the input models. The last few points in the BOSZ models tend to be lower than an extrapolation from the adjacent shorter wavelength points would indicate. It is assumed here that this is a result of the smoothing and not real. In using the shorter wavelength points as a basis for the extrapolation the code may produce a dip in the spectrum around 32 μm that is not real. However, this should not affect the usage of the models, and the dip is useful for marking the boundary between the input values and the extrapolated values.

The code is able to read in the BOSZ models either in the ascii form (whether plain ascii or bzipped) or in the FITS table form, as can be downloaded from the MAST site. Two example files

amp00cp00op00t10000g40v20modrt0b200rs.asc.bz2
amp00cp00op00t10000g40v20modrt0b200rs.fits

are included with the code. These are roughly A0V stellar models, T = 10000 K and $\log_{10}(g)$ = 4.0 for the nominal solar abundances. If the file name ends in ".asc.bz2" the code will unzip the file, read in the values, and zip it up again.

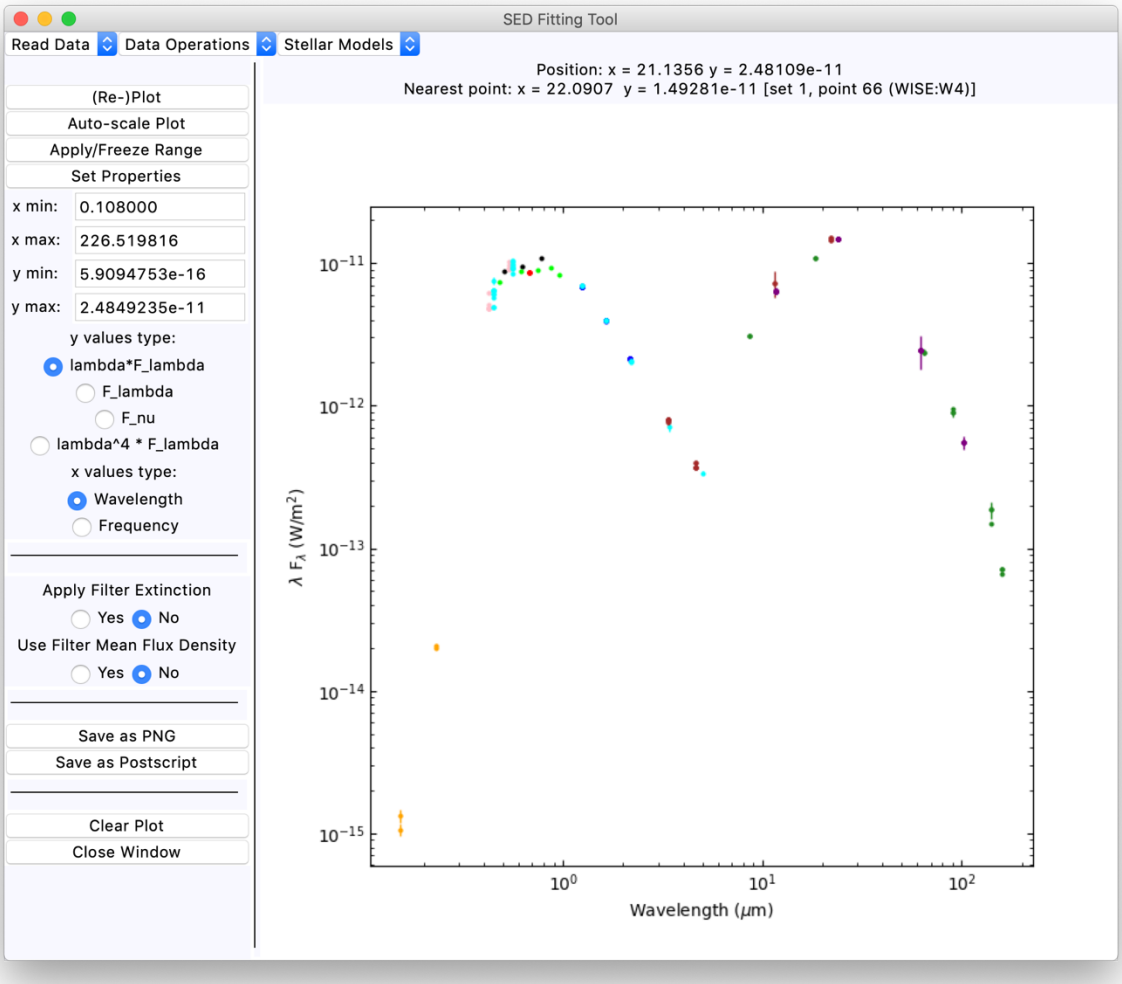

Figure 15: The appearance of the HD 5626 Vizier SED plot when the filter colours option is active.

The BOSZ models are renormalized by default to an F_λ value of 5 10^{-15} W/m^2/μm at a wavelength of 2.2 μm when first read in. The normalization can then be changed via other commands as will be described later. This normalization value is arbitrary, serving only to put the models roughly at the brightness range of a source with K magnitude around 7.3. If one wishes to change these default scaling values the global variables WAVELNORM and FLAMBDANORM need to be changed in `sed_plot_interface.py` and in `model_utilities.py`.

One should not use the high resolution BOSZ models in the code, both because of speed considerations and because the structure in these models is not shown well in the plots over a very wide wavelength range as will normally be the case in this code. Resolutions of up to 3000 or so will still be suitable for this program. The R=200 example files provided with the code were selected because they are small and still serve to test the code. The code itself does not check the resolution, so if one insists on reading in an R=300000 BOSZ model the code will do it and plot the result.

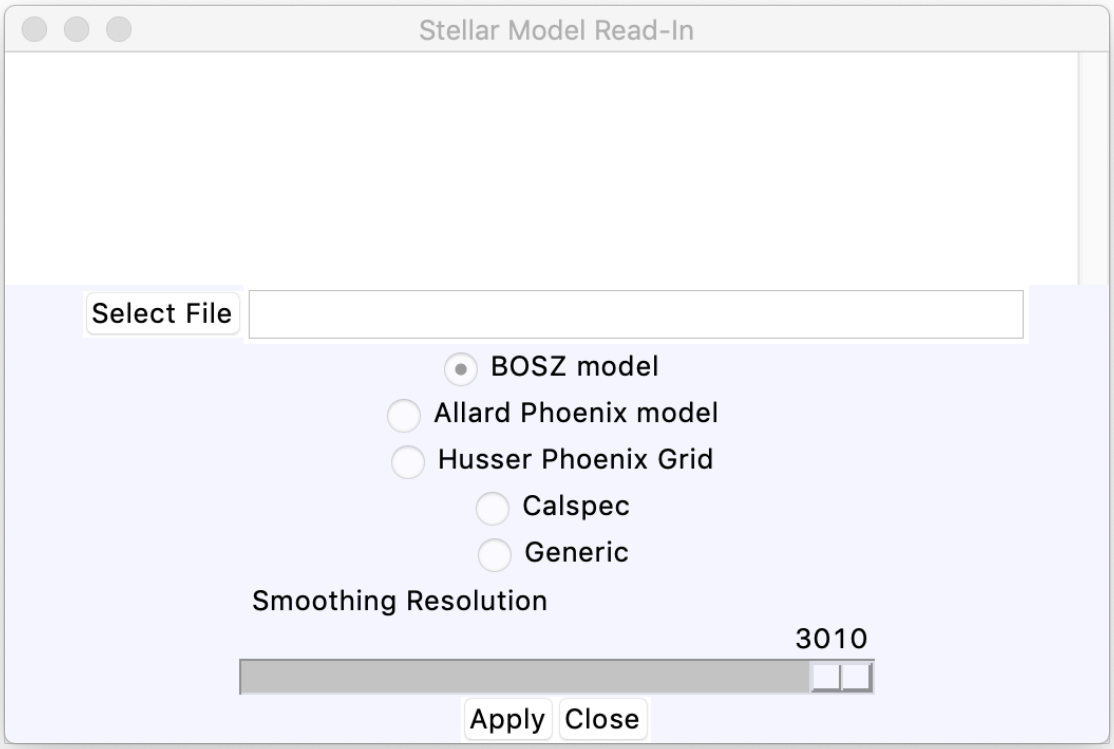

Figure 16: The window for reading in stellar model files.

The France Allard Phoenix Models

A large grid of models calculated with the Phoenix stellar model code is available at France Allard's web site http://perso.ens-lyon.fr/france.allard/, which are widely used for modelling low mass stars and sub-stellar objects in particular. Any of these files should be able to be read in by the code. Specifically the models with the '*.spec.gz" type files names or those with the "*.spec.7.xz" type file names both should be handled properly in the code. Note the files are assumed to be stored in the compressed forms using one of gzip, bzip2, or xz. The code will attempt to unzip the files, read them, and zip them up again. An example file

lte72-4.5-0.0.NextGen.spec.gz

is included with the code as a test case for this option. Any of the BT-Settl, NextGen or AMES models from France Allard should be useable with the code.

The NextGen models have wavelength coverage to 1000 μm, although at relatively low wavelength resolution beyond 82.062 μm. For these models additional points are defined using a power law interpolation between the original grid point, so that the spectra appear smooth at long wavelengths in the log-log plot used here. In the case of the later AMES models the

wavelength range goes to about 50 μm, and the spectrum is extrapolated to 300 μm with a Rayleigh-Jean power law. The BT-Settl models have wavelength coverage to almost 1000 μm and do not need any additional wavelengths for the plotting. In all these cases the extrapolations/interpolations that are done are approximations, and if one is vitally concerned with the detailed spectral shape at these long wavelengths one should supply models that do a proper sampling of the wavelengths and species of interet.

The models are renormalized by default to an F_λ value of 5 10^{-15} W/m^2/μm at a wavelength of 2.2 μm when first read in (as noted above these are set by the WAVELNORM and FLAMBDANORM global variables in the code). The normalization can then be changed via other commands.

These models are high resolution and the details of the spectrum at shorter wavelengths are lost at the resolution of the plots in the tool. To work around this, one can select a resolution with the slider and smooth the input spectrum to this resolution. Resolution values between 100 and 3000 cause the smoothing to be done, values of 90 and 3010 (the left and right limits of the slider) will cause the smoothing step to be skipped. At long wavelengths the smoothing, which uses linear interpolation between the input data points, causes the spectrum to deviate from a simple power-law because the sampling in wavelength is rather sparse. This makes small bumps on the SED for the model. If this spectral region is important for comparison with the data then the full resolution model should be used. This wavelength region is not expected to be vital for fitting observed data.

In some cases, the Phoenix models have zero values after the scaling to the target flux density at the normalization wavelength (nominally 2.2 μm), and in the smoothing calculation this can result in division by zero when there is a series of wavelength points with a value of zero. These errors are not important as the code sets the flux density for such wavelengths back to zero before the values are returned.

The Husser et al. (2013) Phoenix Models

Another grid of Phoenix models is described by Husser, T.-O., Wende-von Berg, S., Driezier, S., et al., 2013, A&A, 553, 6. These models can be obtained from http://phoenix.astro.physik.uni-goettingen.de/ and are FITS table files. The long wavelength limit of these models is only 5.5 μm, which is an issue for longer wavelength filters such as the WISE filters. Hence the models are extrapolated to longer wavelengths using a strict Raleigh-Jeans function at00 relatively low wavelength resolution. In those cases where longer wavelengths are of concern the other stellar model libraries are preferable.

For these models the wavelengths are stored separately from the individual stellar models. The code assumes that the wavelength file is present in the same directory as the input stellar model. The example files provided are

WAVE_PHOENIX-ACES-AGSS-COND-2011.fits
lte05700-5.00-0.0.PHOENIX-ACES-AGSS-COND-2011-HiRes.fits

the first one of which is the wavelength file. Note that the name of the wavelength file above is hard-coded in the python file `model_utilities.py`. As with the previous models, the input stellar model is normalized to F_λ=5 10^{-15} W/m^2/μm at wavelength 2.2 μm, or whatever the WLNORM and FLAMNORM global variables are set to in the code.

These models are high resolution and so the drawing in the tool can be slow. To work around this, one can select a resolution with the slider and smooth the input spectrum to this resolution. Resolution values between 100 and 3000 cause the smoothing to be done, values of 90 and 3010 (the left and right limits of the slider) will cause the smoothing step to be skipped. When smoothing is used, it causes a dip in the spectrum near the long wavelength limit. The extrapolation to longer wavelengths is done with the full resolution spectrum to avoid producing a discontinuity in the output spectrum. As a result, one generally sees a dip and recovery at about 5.5 microns in the model spectrum after the extrapolation and smoothing.

The CALSPEC Models

The individual stellar models of photometric standard stars held in the CALSPEC database https://archive.stsci.edu/hlsps/reference-atlases/cdbs/current_calspec/ can be read in by the code. These files are FITS tables. The code does not scale these models on input the way it does the other models, because the flux densities are within the range observed for stars. For those CALSPEC files that are BOSZ models, and any of the CALSPEC models with a maximum wavelength less than 32.1 μm, the long wavelength extrapolation is done in the same way as for generic BOSZ models. Those models with maximum wavelengths greater than 32 μm are not padded with extra wavelengths, as these normally extend to 300 μm wavelength or longer. As an example for use with the code, the CALSPEC model for Sirius is included. The file name is sirius_mod_003.fits.

Generic Models

This option uses the same interface that reads in a data spectrum. The only difference in usage is that the input file name is selected in the model read-in window, and this name is put into the name field in the spectrum window (Figure 11 above). Hence either a FITS tale file or an ascii file can be read in. One is able to use any of the input units defined for spectral data to read in a model. As with the other model types aside from the CALSPEC models, the input spectrum is scaled to an F_λ value of 5 10^{-15} W/m^2/μm at 2.2 μm after it is read in (set by the FLAMNORM and WLNORM global variables), unless the interpolated flux density is zero at this wavelength in which case no scaling is applied.

As the generic models are read in through the spectrum interface, the message about the wavelength and flux ranges one gets for the "normal" models is not produced in this instance. The file `blackbody_10000k.data` contains the flux density values for a 10000 K blackbody spectrum, with wavelengths in μm in column 1, F_λ values in W/m^2/μm in column 2, and F_ν values in Jansky in column 3; this can be used to test the generic model interface

Blackbody "Models"

One can also define blackbody spectra for the plot. This is done using the "Blackbody Spectrum" menu option under the "Stellar Models" menu. If selected a dialog window comes up asking for the temperature value in K, which must be larger than 4.0. If a suitable temperature is entered, the blackbody function is created using the wavelengths from the CALSPEC Sirius

model and the specified temperature. The spectrum is then normalized in the same way as the other input models and added to the plot.

Example of the Models

Figure 17 below shows four of the spectra from files included with the code, those files being

`amp00cp00op00t10000g40v20modrt0b200rs.asc.bz2`
`amp00cp00op00t10000g40v20modrt0b200rs.fits`

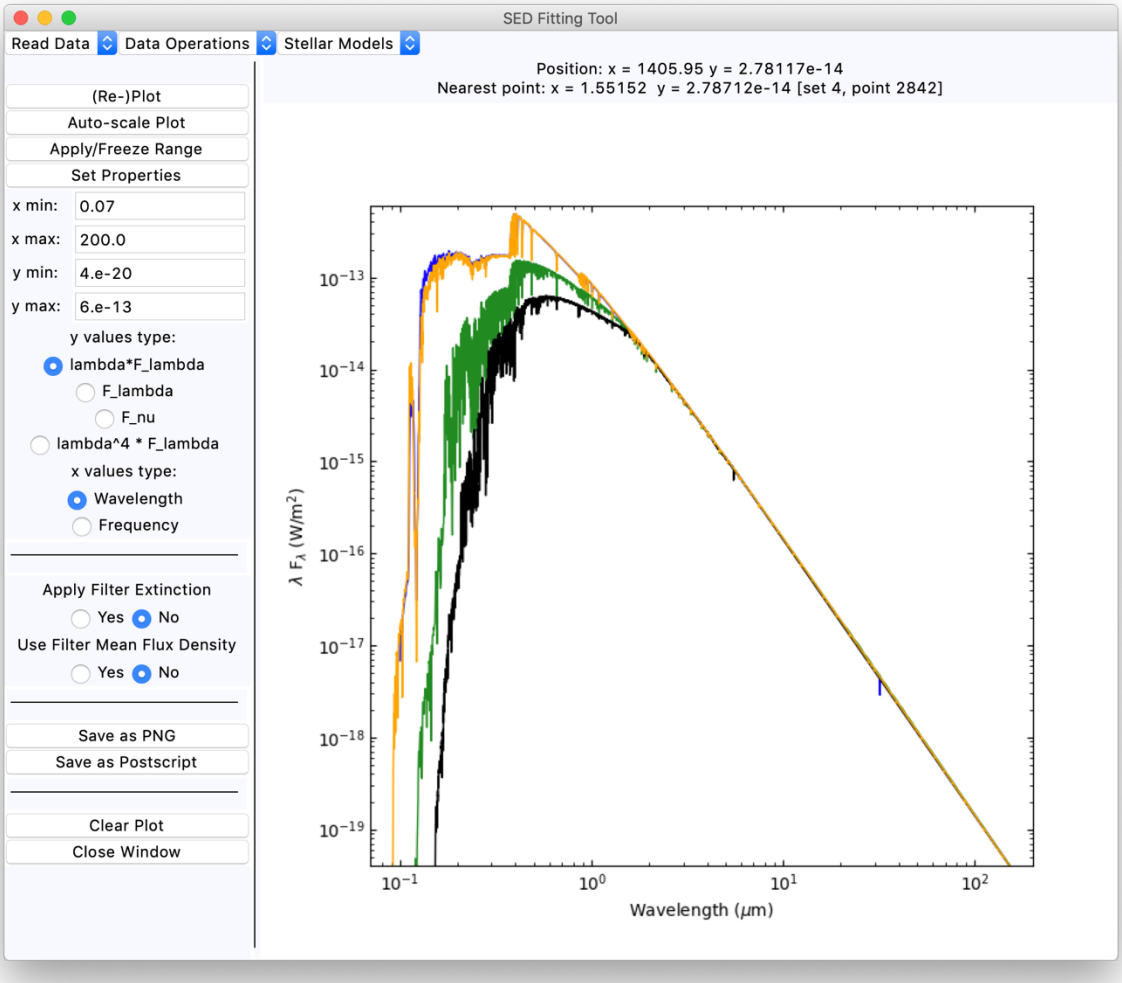

Figure 17: An example of the different stellar models available in the code. The phoenix models (green, Allard model; black, Husser et al. model) have been smoothed to resolution 3000. The BOSZ model plotted in blue. The orange curve shows the CALSPEC Sirius model, scaled down to match the others in the plot.

`blackbody_10000k.data`

```
lte72-4.5-0.0.NextGen.spec.gz
lte05700-5.00-0.0.PHOENIX-ACES-AGSS-COND-2011-HiRes.fits
WAVE_PHOENIX-ACES-AGSS-COND-2011.fits
sirius_mod_003.fits
```

The first two are BOSZ model files, the third file is an example for the generic model case, the fourth file is an Allard phoenix model file, the fifth and fifth files are from the Husser et al. phoenix grid model files, and the last file is the CALSPEC example file. In the Figure the Sirius spectrum has been scaled down to match the others. The plot shows the BOSZ model in blue, the Sirius model in orange, the Allard model in green, and the Husser et al. model in black. The blackbody example is not plotted here.

Scaling Model Spectra

Once a stellar model spectrum has been read in, it can be rescaled to match observed data values. This can be done in two ways. One can select the "Scale Model to Cursor" or the "Scale Model to Data Point" menu items. After this selection, clicking within the plot window will trigger the rescaling. In the first case, the (x,y) coordinates of the cursor when the button press event is received will be used to rescale the model. In the second case, the nearest point to the (x,y) coordinates is used for the scaling. The scaling is done by a linear interpolation of the model spectrum to the wavelength or frequency selected, and then scaling to the flux density selected. If a model has spectral lines near the wavelength that is selected, the scaling will depend on the line profile value with respect to the continuum. If the model spectrum has been smoothed this is less of a concern.

Note that if the point selected to match is beyond the wavelength range of the model that is being scaled then the flux density at the end point is used for the scaling, and that will produce an incorrect scaling of the model. The code for the scaling does not make an attempt to extrapolate values beyond the wavelength limits.

Least-Squares Matching of Models to Data

The code has an option to allow the user to carry out a least-squares fitting of a stellar model to a data set. This option is invoked by the "Find Best Fit Model" menu item in the "Stellar Models" menu. If one has only one data set and one stellar model in the plot, then these are selected for the calculation automatically. If one has multiple data sets or multiple models active in the plot then one is queried for the set numbers for the data set to fit and for the model to use in the fitting.

Once the data set and model are selected, the code will carry out a standard least-squares data fit of the model values to the data values. The default is to match the monochromatic stellar spectrum values to the data set flux densities at filter reference wavelength. This of course is going to produce a poor result if the input spectrum has spectral lines that affect the stellar model flux density at the reference wavelength of a given filter. If the input spectrum is smooth then there is in general only a small error in using the monochromatic flux density at the filter wavelength rather than carrying out an integration over the filter profile. For stellar spectra with lots of spectral lines matching the monochromatic flux density at full spectral resolution will

produce a bad result. When the stellar spectrum is smoothed the errors caused the line structure are corresponding reduced.

This difficultly does not generally apply to matching a stellar model to an observed spectrum, unless the object has significant spectral structure within each resolution element of the spectral data. If a low resolution spectrum is matched to a high resolution stellar model over wavelengths where many spectral lines are present the matching will again produce bad results in such a case.

Where photometry values are being matched the best method is to simulate the photometric observations by integrating the stellar spectrum over the filter response, with proper standardization of the flux density values if needed. While the code does not attempt this in the general case, there is an option in the code to carry out integrations over the filter profile for a sub-set of commonly used filters, and matching these to the data values. This can only be done when the filter names are given for the photometric data points, as is the case for the Vizier SED data. The radio button "Use Filter Men Flux Density" at left in the main window can be used to select this option.

The filter response functions for a sub-set of possible filters has been collected from the Spanish Virtual Observatory filter service at http://svo2.cab.inta-csic.es/theory/fps/ and placed in the `filter_subset` directory that is provided along with the code. The filters included are: the Johnson UBVIR filters, the Bessell UBVRI filters, the Stromgren ubvy filters, the GAIA DR2 filters, the SDSS ugriz and u′g′r′i′z′ filters, the 2MASS JHK$_s$ filters, the UKIDSS YZJHK filters, and the WISE W1 to W4 filters. When Vizier photometry data is used in the tool each point has a filter label. If the label for the data point matches the expected label for one of the above filters the code is able to calculate the mean flux density via integration. Assuming that the photometric observation is made using a photon-detecting instrument, a CCD or an HgCdTe infrared detector for example, then the observed signal is proportional to the photon weighted mean flux density

$$\langle F_\nu \rangle = \frac{\int \frac{1}{\nu} F_\nu \phi_\nu d\nu}{\int \frac{1}{\nu} \phi_\nu d\nu} \tag{1}$$

where ϕ_ν is the total (filter plus detector plus telescope) response. The wavelength equivalent of this is

$$\langle F_\lambda \rangle = \frac{\int \lambda F_\lambda \phi_\lambda d\lambda}{\int \lambda \phi_\lambda d\lambda} \tag{2}$$

and the mean F_λ value is related to the mean F_ν value in the usual way if the pivot wavelength is used in the equations. See the discussion from Bohlin, Gordon, and Trembley (2014) [PASP, 126, 711] section 2.2 concerning the photon-weighted mean flux density and its relation to observed quantities in photometry.

If the filter integration option is selected the mean flux density calculation (in F_λ units) is carried out for the filters that have profiles in the package, and these values are then directly compared to the Vizier photometry values for the fitting. Data points for filters other than the standard ones or for spectral points without an associated filter name are treated monochromatically as in the normal case. Note that is it not certain that the Vizier SED values

are calculated on a consistent basis to what is done in the code for this option: both the wavelengths reported by the Vizier SED service and those tabulated by the SVO filter service are not reproduced exactly by direct calculations on the SVO filter profiles. These small disagreements may not cause any systematic errors in the matching, but one cannot be certain of this.

To get the filter profiles the code looks in the directory defined by the $EXTINCTION_PATH environment variable. This is due to the code using these filter response files for extinction calculations as well as for the mean flux density calculations. See below for a discussion of the filter extinction option. If the $EXTINCTION_PATH environment variable is not defined, the path is assumed to be to the current directory.

The Least-Squares Algorithm

For either the monochromatic case or the filter integration case the program assembles the data values for the filter flux densities O_i and the corresponding model flux densities M_i. Only the unmasked points are used in the calculation. Once the two sets of values have been complied the scaling factor s is calculated from

$$s = \frac{\sum O_i/\sigma_i^2}{\sum M_i/\sigma_i^2} \tag{3}$$

where σ_i is the relative flux density uncertainty of photometry point O_i. Photometric or spectroscopic points with no listed uncertainty are assigned 5% relative uncertainty in the fitting. Once the scaling factor s has been calculated it is applied to the model data set.

The above formulation of the scaling value is dominated by the points of higher flux density, and so for a normal star the long wavelength points have low impact on the scaling. An alternative is to directly calculate the mean scaling factor

$$s' = \frac{\sum \frac{1}{\sigma_i^2}\frac{O_i}{M_i}}{\sum \frac{1}{\sigma_i^2}} \tag{4}$$

which weights the points by the relative deviations. This will give equal weight to the long wavelength and short wavelength photometry points.

The code allows one to change from using equation (3) to equation (4) or back again. There is an internal flag that determines which form is used in the fitting. The default is the regular least-squares calculation in equation (3). One can toggle the flag using the "Toggle Standard/Ratio Fit" item in the "Stellar Models" menu. When toggled an information window comes up stating either that the standard least squares calculation is to be used, equation (3), or that the ratio calculation will be used, equation (4).

A projected future revision is to allow the user to specify a set of stellar models and an extinction range, and then have the code find the best fit model from the least-squares data fitting. The "Find Best Fit Model" menu item under "Stellar Models" is intended for this purpose. Currently the code brings up the file selection window but the option does not carry out any more functions than this. Another projected revision would be to have a scaling option only

using the standard filter photometry and the photon-weighted mean flux density values, where such are available, excluding the monochromatic points.

Treatment of Interstellar Extinction

The code allows the user to apply interstellar extinction functions to a data set, either to de-redden observed data or to redden model spectra. There are several extinction laws available for use. Most of the extinction functions are available via the Python "extinction" package that needs to be installed using the command

```
pip install extinction
```

in order for the code to work. This package is described at the documentation page https://extinction.readthedocs.io/en/latest/. If this is not installed, the code will not be able to run and an error message will occur stating that the extinction package is not found. The different extinction functions will now be described.

Rieke and Lebofsky (1985)

The original extinction function in Rieke, G. H., and Lebofsky, M. J., 1985, ApJ, 288, 618 provides extinction estimates to 13 μm including the effect of the 10 μm silicate feature. The extinction function here is an extended version of the extinction curves in this paper. The author extended the Rieke and Lebofsky (1985) values to both longer and shorter wavelengths. At wavelengths longer than 5 μm adjustments were made to the extinction function based on the IRAS Low Resolution Spectra of silicate emission sources, plus an extrapolation of the underlying continuum extinction to much longer wavelengths. Extinction was set to zero at wavelengths above 300 μm.

The extinction function was also extended to the ultraviolet wavelengths using the standard extinction curve from Cardello, Clayton, and Mathis (1989) from the U band to 0.0912 μm. The result is a standard extinction curve normalized to a value of 1.0 at 5500 Å. For any specified A_V value in magnitudes the standard curve is multiplied by the input A_V value and interpolated to the set of input wavelengths. There is no way to vary the $A_V/E(B-V)$ value for this extinction curve. The value is nominally 3.08 but when simulated with the filter profiles the value is around 3.5.

The extinction values are read from an ascii input file named `extinction.values`, assumed to be either in a directory pointed to by the EXTINCTION_PATH environment variable or in the current directory in the case where this environment variable is not defined.

As this is the only one of the extinction laws included in the code that allows for the silicate extinction at long wavelengths, this is the default extinction function in the code. The other functions are preferable if one is dealing the ultraviolet spectra.

Cardello, Clayton, and Mathis (1989)

The original paper is Cardelli, J. A., Clayton, G. C., and Mathis, J. S., 1989, ApJ, 345, 245. A parameterized extinction function depending on the assumed A_V and $A_V/E(B-V)$ values

in magnitudes was derived from a set of extinction observations. The latter parameter will be denoted as R_V. That function is used here. The default R_V value is 3.1, but the value can be changed between 2 and 6 in the code. The input data in this paper covers an R_V range from 2.60 to 5.3. It is noted that there are variations in the UBV and ultraviolet extinction values where R_V is larger than 4, but that the extinction seems to be independent of the R_V value at wavelengths >0.9 μm.

O'Donnell (1994)

The extinction function from O'Donnell, J. E., 1994, ApJ, 422, 158 is selected by this option. The methodology in the paper is similar to that in the Cardelli, Clayton, and Mathis (1989) paper.

Calzetti et al. (2000)

The source is this extinction function is Calzetti, D., Armis, L., Bohlin, R. C., et al., 2000, ApJ. 533, 682. This function was derived from work on de-reddening the spectral energy distributions of starburst galaxies, and so is intended for circumstances where massive stars produce a strong radiation field to illuminate the interstellar dust and gas. As such, this extinction function is rather different from the others included with the code. Their nominal R_V value is 4.05±0.8, and so is provisionally higher than what is observed in the Galaxy.

Fitzpatrick (1999)

Fitzpatrick, E. L., 1999, PASP, 111, 63 gives a parameterization of the extinction curve, which agrees fairly closely with the values from Cardello, Clayton, and Mathis (1989) for optical wavelengths and over most of the infrared wavelengths considered but which deviates significantly in the ultraviolet from the earlier function. There are also some variations between this function and the earlier Cardello, Clayton, and Mathis results for wavelengths from 1 μm to about 4 μm when the R_V value is large.

Fitzpatrick and Massa (2007)

This final function is from Fitzpatrick, E. L., and Massa, D., 2007, ApJ, 663, 320. The function is specific to the "normal" R_V=3.1 case. This is an extension of the earlier Fitzpatrick (1999) extinction function.

Application of Extinction

The extinction function for data can be activated using the "De-redden Data" item under the "Data Operations" menu. The analogous function for model spectra is activated by the "Redden Model" menu item under the "Stellar Models" menu. When either item is activated an extinction control window comes up. The only difference in the two cases is whether the default function is to de-redden or to redden a set of data values, and which set value is the default. The appearance of the window is shown in Figure 18. The window has two sliders, one for the A_V

parameter and one for the R_V parameter. The A_V value is in magnitudes. The range of the A_V slider is from 0.0 to 20.0 with steps of 0.01 magnitudes. The range for R_V is 2.0 to 6.0 with steps of 0.01.

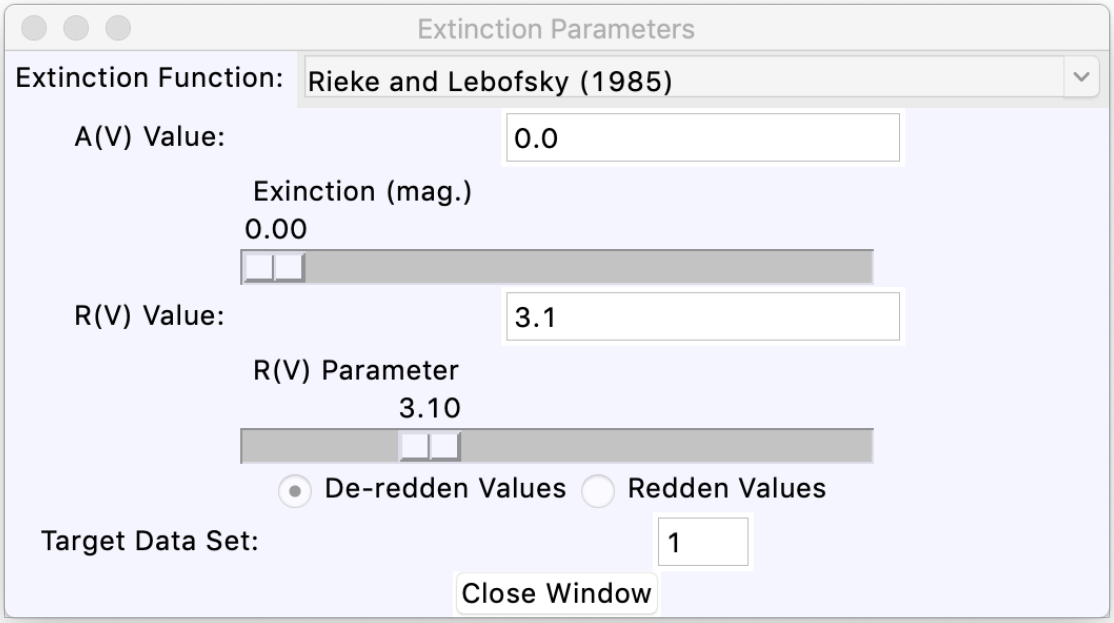

Figure 18: The extinction control window.

One needs to remember that the extinction functions are approximate in the sense that they are averages over many lines of sight. High precision extinction estimates for a given object require a lot of effort and a wide range of input data to work with. This code is not optimized for such a study.

Figure 19 below shows the appearance of the plot window with the two data sets for HD 56126 from Figure 13 with an applied de-reddening of 0.15. The new sets in black and in gold are the de-reddened photometry and ISO spectrum respectively.

The tool works one data set at a time. When it is called from the "De-redden Data" item the code sets the radio button to "De-redden Values". The code the looks at the data sets to see which is the first one that has observed values rather than being a stellar model and places that number into the "Target Data Set" field. The extinction values are applied when either of the sliders has a change in the value.

When a data set has extinction corrections applied initially, a new set is created with a source label such as "set 1 with extinction 0.10/3.10/1" or "model 1 with extinction 0.1/3.10/1" where the first number is the set index. Stellar models are read in and assigned a source label starting with 'stellar model'. When extinction is applied to any set with such a source label the extinction corrected set source label starts with "model", otherwise the source label starts with "set". The extinction parameters A_V, R_V, and index number for the extinction function are written to the source label. The number for the extinction function is 1 for the Rieke and Lebofsky (1989) function, and so on as in the menu items in the window. Subsequently, if the extinction calculation is repeated the results are put in whichever the data set has the expected

source label “set # with extinction” or “model # with extinction”, excluding the extinction parameters as these will vary. This is to prevent having multiple data sets for a range of extinction values in the plot. If one has multiple data sets from the same object and wishes to apply the same extinction to each one, the best course is to apply the extinction to one set and find a value that seems appropriate, and then apply the same parameters to any other data sets that are from the same object. This cannot be done automatically as only the Vizier input data values are required to include position information. After the target set number is changed one has to move one of the sliders to get the values applied, so one needs to tick the extinction value off by 0.01 magnitudes and back again, or something similar, to apply the extinction function. The same is true if the extinction function menu setting is changed; nothing happens until one of the sliders is moved.

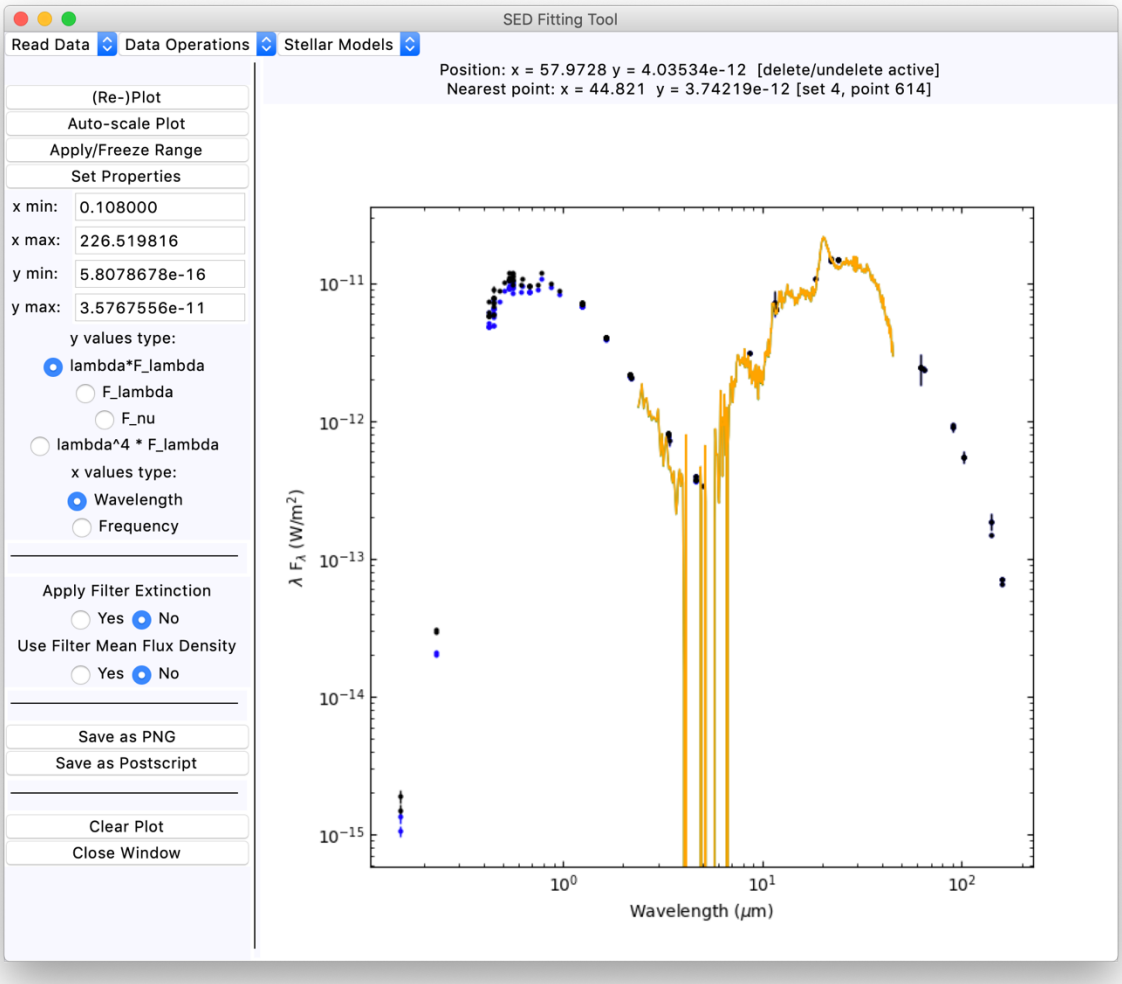

Figure 19: The main SED window plot for the HD 56126 data as in Figure 13, with the addition of de-reddening both data sets by E(B –V) of 0.15 magnitudes with the default extinction function.

One can switch between de-reddening and reddening the data with the radio button at any time, although any changes are only applied when one of the sliders is moved. The source labels for the extinction corrected data sets do not distinguish between reddening and de-reddening, but just gives the parameters that were used.

Filter Extinction and Monochromatic Extinction

The different extinction functions described above all produce a monochromatic extinction value at a specified wavelength. For reddening a stellar model or de-reddening spectroscopic data this is what is needed. When attempting to de-redden filter photometry the situation is more complex. In such cases the extinction is a function of both the filter profile and the spectrum of the source under consideration. One cannot produce a general extinction function for, say, the Johnson V filter that is accurate for all spectral types nor for each of the "versions" of the filter at different observatories.

In most cases when filter photometry values are de-reddened the monochromatic extinction is calculated at the filter wavelength and applied to the photometry values. This is an approximation, both because the filter wavelength may be assigned in different ways from one case to another and because the extinction will vary over the filter response. For narrower filters or cases where the object has a smooth spectrum over the filter bandpass the use of the monochromatic extinction will be fairly accurate. For wider filters, such as the GAIA G filter for example, this becomes a poor approximation.

For a small number of widely used filters the code is able to calculate the filter extinction explicitly via the formula

$$\Delta m = 2.5\log_{10}\left(\frac{\int \lambda F_\lambda \phi_\lambda \, \mathrm{d}\lambda}{\int \lambda \mathrm{F}_\lambda E_\lambda(A_V, R_V)\, \phi_\lambda \mathrm{d}\lambda}\right) \tag{5}$$

where λ is the wavelength, the filter response function is ϕ_λ, the ISM extinction function is $E(A_V, R_V)$, and F_λ is the object wavelength flux density. The extra factor of λ in the integrations allows for photon detecting instrumentation, as is normally the case for current optical and near-infrared detectors. The filter response function ϕ_λ should include the telescope response and the quantum efficiency of the detector. Hence, it should be the "total photon conversion efficiency" rather than just the filter response function. In practical terms, the filter response generally dominates the response and so a reasonably accurate result is obtained from using just the filter response.

The same set of filters as can be used for the photon weighted mean flux density calculations in matching stellar models to data sets, described previously, can be used for an explicit filter extinction calculation. When Vizier photometry data is used in the tool each point has a filter label. These filter labels are compared to the tags for the 41 specific filters for which profiles are provided, and if a match is found the code can use equation (5) above to calculate the effective extinction. The default spectral shape used is the Sirius model spectrum from the CALSPEC database read in from file `sirius_mod_003.fits`. This is a somewhat time-consuming process so the option to use filter extinction is controlled by a radio button in the area to the left of the plot region. The default is to not use filter extinction to save time in the code.

If the object of interest is of much later spectral type than A1V then the filter extinction estimates will become less accurate. One can override the default spectral shape assumed using

the “Set Standard Spectrum” button in the “Stellar Models” menu. One then can select any of the models currently used in the code as the standard spectral shape for the extinction calculation. This provides flexibility in which spectral shape is used for the extinction calculation, but it may slow down the code significantly if a finely sampled model spectrum is used. One needs to read in whatever model is to be used as the extinction template model, except for the default Sirius model, before it can be used for this purpose. Also note that once the template spectral shape has been changed from the default the code does not allow one to revert to the default case except by clearing the plot.

Comparison of the filter extinction values and the monochromatic extinction values at the filter wavelength show that one can get of order 10% deviations between the two calculations. This difference may be of little concern for low values of the extinction. It obviously is of more concern for objects subject to higher amounts of reddening. Further improvements in this aspect of the code (e.g. plotting the residuals between a reddened model and a set of data points to assess the quality of the extinction correction) may be implemented at a later date.

Other Functions

One can save the current plot as a PNG file or as a postscript file using the buttons at left “Save as PNG” and “Save as Postscript”. These cause the program to query for a file name wherein to save the current plot in the format requested.

There is some control of the properties of the plotting of the data sets using the “Set Properties” button on the main window. This brings up a small control window as seen in Figure 20. One can select a symbol, or have no symbol, and select a line type, or have no line joining the points. The window also allows changes to the symbol size and line thickness. Whatever parameters are set in the fields are applied only when the “Apply” button is clicked.

One can use a zoom function with the cursor by activating the “Cursor Zoom” menu item. When this function is enabled, one can define a zoom box by clicking on a position and holding the mouse button down as the cursor is moved. When the button is released, the initial point where the mouse button was pressed and the final point where the mouse button was released define the new plot limits. One can return to the normal plot limits using the “Auto-Scale Plot” button at left from the main plot area. The other way to control the plot limits is to set values in the range fields at left in the tool, and then use the “Apply/Freeze Range” button. The default rounding used in matplotlib is often less than ideal for the log-log plots here, so changing the range values to round numbers is often required to make the plot look better.

One can clear the current plot and all its associated data sets with the “Clear Plot” button at left from the display area. A pop-up verification window will appear to ask for confirmation of this, since it is irreversible. Similarly, the “Close Window” button queries for confirmation before the program is terminated. One can close the window with no such query using the button at the upper left corner of the window, should one tire of confirming that the window should indeed be closed.

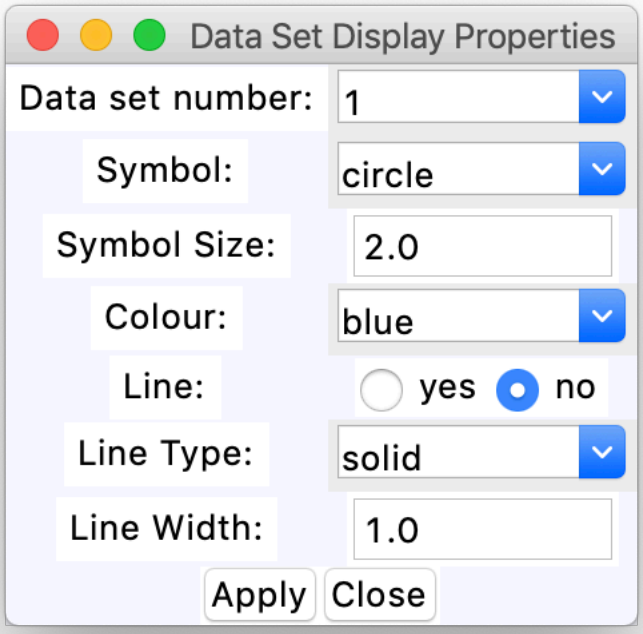

Figure 20: The window for setting symbol properties.

The Package Files

The full set of files for the SED display code is

```
WAVE_PHOENIX-ACES-AGSS-COND-2011.fits
amp00cp00op00t10000g40v20modrt0b200rs.asc.bz2
amp00cp00op00t10000g40v20modrt0b200rs.fits
blackbody_10000k.data
extinction.values
extinction_code.py
lte05700-5.00-0.0.PHOENIX-ACES-AGSS-COND-2011-HiRes.fits
lte72-4.5-0.0.NextGen.spec.gz
model_utilities.py
position_plot.py
read_vizier_sed_table.py
sed_display_and_fitting_code.docx
sed_plot_interface.py
sed_utilities.py
sirius_mod_004.fits
sws01_07134_lfl.dat
```

```
tkinter_utilities.py
vizier_votable_hd_56126_radius_5p0.vot
```

plus the files in the `filter_subset` directory which needs to be under the same directory that the above files are placed in. The contents of the subdirectory are

```
filter_2MASS_2MASS_H.vot
filter_2MASS_2MASS_J.vot
filter_2MASS_2MASS_Ks.vot
filter_GAIA_GAIA2r_G.vot
filter_GAIA_GAIA2r_Gbp.vot
filter_GAIA_GAIA2r_Grp.vot
filter_Generic_Bessell_B.vot
filter_Generic_Bessell_I.vot
filter_Generic_Bessell_R.vot
filter_Generic_Bessell_U.vot
filter_Generic_Bessell_V.vot
filter_Generic_Cousins_I.vot
filter_Generic_Cousins_R.vot
filter_Generic_Johnson_B.vot
filter_Generic_Johnson_I.vot
filter_Generic_Johnson_R.vot
filter_Generic_Johnson_U.vot
filter_Generic_Johnson_V.vot
filter_Generic_Stromgren_b.vot
filter_Generic_Stromgren_u.vot
filter_Generic_Stromgren_v.vot
filter_Generic_Stromgren_y.vot
filter_SLOAN_SDSS_g.vot
filter_SLOAN_SDSS_gprime_filter.vot
filter_SLOAN_SDSS_i.vot
filter_SLOAN_SDSS_iprime_filter.vot
filter_SLOAN_SDSS_r.vot
filter_SLOAN_SDSS_rprime_filter.vot
filter_SLOAN_SDSS_u.vot
filter_SLOAN_SDSS_uprime_filter.vot
filter_SLOAN_SDSS_z.vot
filter_SLOAN_SDSS_zprime_filter.vot
filter_UKIRT_UKIDSS_H.vot
filter_UKIRT_UKIDSS_J.vot
filter_UKIRT_UKIDSS_K.vot
filter_UKIRT_UKIDSS_Y.vot
filter_UKIRT_UKIDSS_Z.vot
filter_WISE_WISE_W1.vot
filter_WISE_WISE_W2.vot
filter_WISE_WISE_W3.vot
filter_WISE_WISE_W4.vot
```

The other requirement to run the code from directories other than the one where all the above files is located is to define the `$EXTINCTION_PATH` environment variable to point to the directory where the `extinction.values` file is. One could in principle place the filter profile sub-directory, the file `extinction.values` and the file `sirius_mod_004.fits` in some other directory and define the `$EXTINCTION_PATH` environment variable to point there.

To run this code from arbitrary directories one would need to add the directory to the `$PYTHON_PATH` environment variable. The code currently is not organized as a package.