The Rutgers Pipeline for Reducing SALT Supernovae Spectra

Viraj Pandya[*]

May 27, 2013

Contents

[*]Rutgers Supernova Group. E-mail: viraj.pandya@gmail.com.

1 Preface

The Rutgers Pipeline for Reducing SALT Supernovae Spectra was written with the goal of helping members of the Rutgers Supernova Group reduce supernovae spectra in a matter of minutes rather than hours or days. This allows one to experiment with particular steps of the reduction process and witness very quickly the effects of those experiments on the final product.

Although this user manual explains how to use PyRAF and PySALT, users are still encouraged to read the original, official manuals [1, 3, 4] as well as [2] for a wonderfully lucid overview of the process of astronomical spectroscopy.

2 Pipeline Preliminaries

This section is home to an overview concerning the nature of the data that will be operated on, and the way the pipeline will undertake those operations.

2.1 Data and Headers

In what follows, we shall assume that the data are images and that they are stored in FITS files[1]. Generally, the FITS filetype allows one to create a file with a list of *extensions*. Each extension in the list can contain what is referred to as a "Header-Data Unit" (HDU), and the first extension (usually extension 0) is referred to as the "primary HDU". The header of an extension contains keywords, and values for those keywords; the data of an extension is essentially the image, and it is recorded as an array of values corresponding to each pixel in the image[2].

The Rutgers SALT Pipeline operates on data that has already undergone preliminary reductions with the official SALT pipeline: PySALT [1]. Astronomers at the observatory take care of, for example, bias-subtractions and dark-current-subtractions for the raw data using PySALT. This results in FITS files with a certain structure:

1. Extension 0 contains the "primary header," but absolutely no data.
2. Extension 1, also named the 'SCI' extension, contains a header as well as the data (the image or spectrum).

In the process of reducing, extracting, and calibrating data, astronomers use a program known as IRAF, or its Python-equivalent: PyRAF. These will be introduced in a future subsection, but the important thing to note is that the tasks performed by IRAF/PyRAF do not produce FITS files with the same structure as PySALT-processed files. FITS files produced by IRAF/PyRAF store everything in the primary extension (0), i.e., there is only one header, and the data is stored in extension 0 (not extension 1).

The entire header of a single image can be pulled up by typing the following at the PyRAF prompt: **imhead *filename.fits*[Extension Number] long+**. A partial listing of user-specified keywords and values of one or a set of images can be pulled up by using **hselect**. For **hselect** and other tasks that can be run on multiple files, it is useful to know that the asterisk (*) character can be used as a wildcard: **.fits* will run the task on all fits files in the current directory, whereas *m*.fits* will run the task on all fits files in the current directory whose names start with "m".

[1]See `http://heasarc.gsfc.nasa.gov/docs/heasarc/fits_overview.html` for more information on the FITS filetype.

[2]Sometimes, the data is not an array of pixel values, but rather a table of information. We shall not concern ourselves with that possibility here, but the idea is similar to what was just described.

2.2 Data Structures

The Rutgers SALT pipeline makes good use of the Python data structures known as *dictionaries*[3]. Dictionaries can be thought of as lists where each element is of the form *keyword:value*. The "values" of dictionary elements can be looked up by simply calling the keyword in that dictionary.

There are several dictionaries that are created by the pipeline to store the filenames of images. Since there need to be interactions between certain images – for example, a science image might need to be divided by a flat-field image during the flat-fielding process – the keywords used in each dictionary must be linked in some way to the keywords used in another dictionary. It is for this reason that each dictionary's keywords are the grating-angles. This allows the image filenames to be indexed by their grating-angle, and so when images of a different type need to be linked up, their filenames can simply be looked up by calling the same grating-angle keyword in the two dictionaries.

The dictionaries are *global* dictionaries in that they exist in the pipeline outside of the functions, and any changes to a dictionary by some particular pipeline function will also change the dictionary for other pipeline functions. In other words, the dictionaries are the same for all pipeline functions.

2.3 Python Functions

Python is the language in which the pipeline has been coded. In particular, each step of the reduction process is separated into its own Python function. Furthermore, each PyRAF/PySALT task has its own function as well in which the parameters of the task are set. In general, the "top-level reduction-step" functions call the PyRAF/PySALT "run" functions.

In Python, functions can have arguments. The PyRAF/PySALT "run" functions have arguments which do not generally have default values. This means that these functions cannot be called without specifying arguments, but since it is the "top-level" functions that call the "run" functions, the arguments are always specified. All of the "top-level" functions have default values specified for their arguments, so the user doesn't have to specify anything to run the function.

However, if individual reduction steps are being done with the pipeline as opposed to the mostly-automated full-reduction process, then the user will be prompted for filenames, or sometimes, parameter values before the relevant PyRAF/PySALT task can be run. What the user inputs at the prompt will then be fed to the appropriate pipeline functions to get the job done.

2.4 PyRAF and PySALT Tasks

It has been mentioned several times now that astronomers use IRAF, or its Python-equivalent PyRAF, and the SALT pipeline, PySALT to do their data reductions, extractions, and calibrations. The Rutgers SALT pipeline imports and calls PyRAF and PySALT tasks. Most PyRAF tasks are run non-interactively because the pipeline feeds them the appropriate parameters, while others require input from the user to ensure that the correct data is being extracted or calibrated. The same goes for the PySALT tasks that the pipeline calls.

PyRAF and PySALT each have their own required keystrokes and sequences of steps to follow. Although this is all explained in the relevant subsections of Section 3, pressing **?** while in a PyRAF or PySALT window will inactivate that window and activate the aschere terminal window with the current task's help file. You can scroll through by pressing **d** and return to the task window by

[3]See `http://docs.python.org/2/tutorial/datastructures.html` for the official documentation on Python dictionaries.

pressing **q**. These task help files can be very insightful, as is expected, so users of the Rutgers SALT pipeline are encouraged to read through them at least once.

2.5 Output Filenames

Since the Rutgers SALT pipeline has the option to be mostly-automated, it needs to follow a naming convention for the output files. Here is the general idea behind the naming convention for functions that produce new files. The first three letters of the filename are self-explanatory: "arc", "sci", "std", or "flt". The next five characters indicate the grating-angle; they are of the form "00.00".

The next three characters indicate what step of the reduction process was run on the image. These are relatively self-explanatory, but since arc emission lines are identified twice and thus wavelength solutions applied twice, the naming convention for wavelength-calibrated images can be ambiguous. The application of the first wavelength solution by the PySALT task *specrectify* results in images with the letters "wav" in them, and the application of the second wavelength solution by the PyRAF task *dispcor* results in images with the letters "dsp" in them[4].

The last three numbers of the image filename indicate the unique number which concludes the "mbxgp" image names. This is helpful for identifying which original images correspond to pipeline-processed images without pulling up header information. Some images, notably arc lamp spectra, have additional characters beyond these three numbers which indicate additional self-explanatory information. Since, in principle, each science and standard star image must have its own arc lamp image for wavelength calibration, the additional characters at the end of arc image filenames indicate whether it is associated with a science or standard star image.

There are some functions that produce auxiliary files, and these files may not follow the naming convention described above. For example, during the flux calibration process, the PyRAF task *sensfunc* produces sensitivity function images. The first *four* letters of these sensitivity function image filenames are "sens". In another case, the PyRAF task *standard* results in text files without extensions and which begin with the three letters *std* – not to be confused with any standard star image filenames. The PySALT task *specidentify* produces text files with .db extensions, and whose filenames start with these *five* letters: "arcid".

3 Detailed Procedure for Using the Pipeline

This section goes through each step of the full reduction process in great detail. It can be thought of as a guide for the user.

3.1 Starting the Pipeline

You must first make sure of the following things:

- You are in your working directory, i.e., a directory with a copy of the files from the product directory. These files are the product files provided by the SALT observatory, and have an "mbxgp" prefix.
- The pipeline python file (file with a .py extension) is in this working directory.

[4]In fact, images with "wav" in their name are two-dimensional, i.e., the spectrum has not been extracted yet. Images with "dsp" in their name are one-dimensional (spectra).

- Your aschere account has either automatically set or you have manually set the command: **source /usr/local/astro64/login.csh**.

You now want to start the pipeline by typing, at the command line, **python pipeline.py** where pipeline.py is the name of the pipeline python file. The pipeline should now be running, and it will ask you what you wish to do. You have several options.

Option 0 will clone all the files in your current directory to a new subdirectory and switch to that new subdirectory: this will become your working directory. This option isn't necessary if you are already in a working directory.

Option 1 will begin the mostly-automated full reduction process. This will run through every single step of the algorithm of the pipeline going from the original product data to a final, combined spectrum of the object.

Option 2 leads to a submenu from which you can choose to run individual pipeline tasks. For example, you might wish to do nothing but combine and normalize those flat-field images in your working directory that have the same grating angle (GR-ANGLE) parameter. Most options in this submenu will require you to input image filenames and perhaps parameter values for the PyRAF task you wish to run.

Option 3 will exit the pipeline and return you to the command line.

The rest of this section assumes that you have chosen Option 1, i.e., to do a complete reduction of the product data. This section will also show images from the reduction process (using the pipeline) for a Type Ia supernova known as sn2012fr.

3.2 Sorting the Images

After choosing Option 1 at the pipeline's main menu, the pipeline doesn't yet know which of the images in the directory are flat-fields, science, standard star, or arc lamp images. It must look up the required information in each image's header, and place that image's filename into the appropriate dictionary. This will be done automatically, and at the end, four dictionaries will be printed on the screen: *flats, sciences, standards, arcs.*

There is generally only one science, standard star, and arc lamp image corresponding to some particular grating-angle (GR-ANGLE keyword), but there are multiple flat-field images (usually five). Thus, whereas the *sciences, standards*, and *arcs* dictionaries will have one string (a filename) corresponding to each keyword (GR-ANGLE), the *flats* dictionary will have a list of strings (filenames) corresponding to each GR-ANGLE.

3.3 Combining Flat-field Images

Each list of flat-field images with the same grating-angle must be combined into one flat-field image. This step is done automatically using the PyRAF task *flatcombine*. Each resulting combined flat-field image's filename (a string) is added to the dictionary *combflats* indexed by the same GR-ANGLE keyword.

3.4 Normalizing Combined Flat-fields

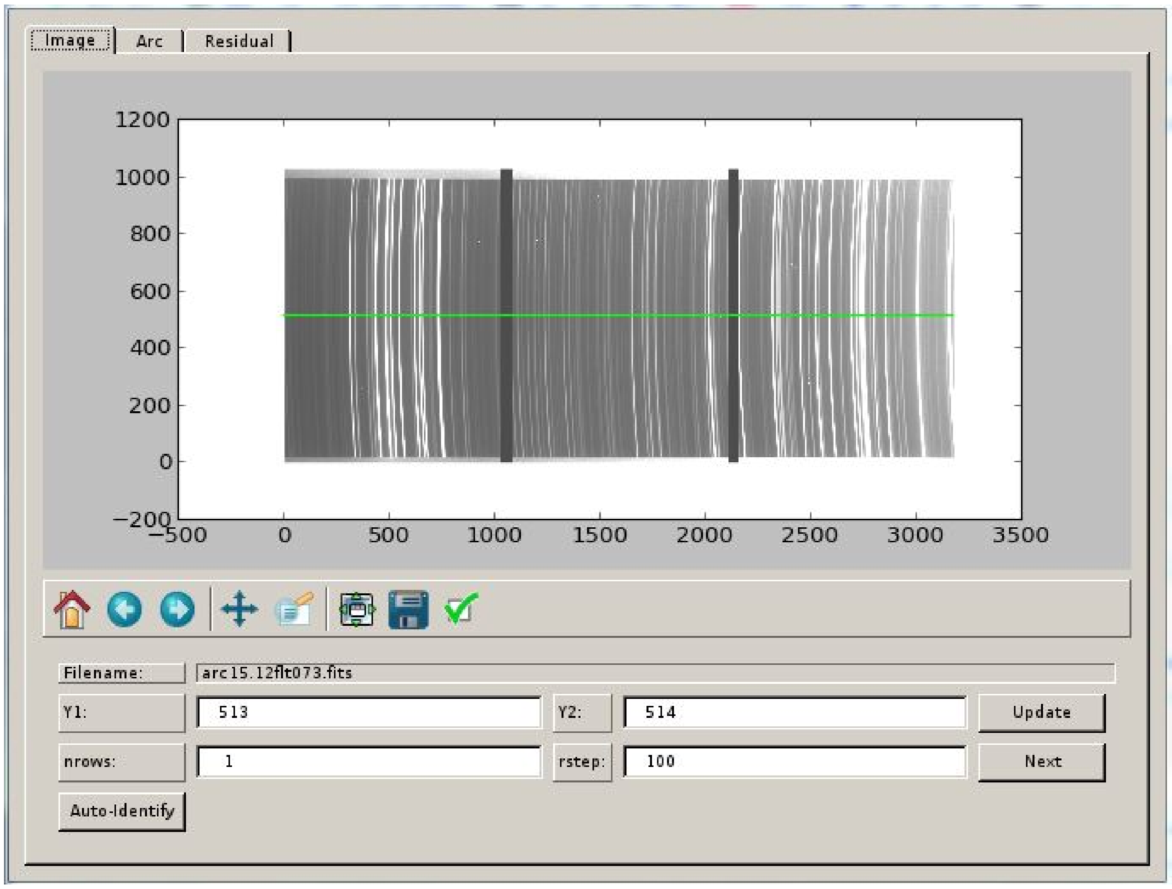
The science, standard star, and arc lamp images must be divided not by the combined flat-field images, but rather by normalized flat-field images. This step is done automatically by the PyRAF task *response*: it divides each combined flat-field image by itself. The pipeline then adds the resulting normalized flat-field image's filename to the dictionary *normflats* indexed by the same GR-ANGLE keyword as before.

3.5 Applying Flat-fields

We can now divide the original science, standard star, and arc lamp images by the normalized flat-field images. This is done automatically by the PyRAF task *imarith*: it simply performs an arithmetic operation (division in this case) using a dividend image and a divisor image. The divisor image is one of the normalized flat-field images, and the dividend image is either a science, standard star, or arc lamp image with the same GR-ANGLE. The resulting output (quotient) image's filename is added by the pipeline to one of the following dictionaries based on the dividend image type: *flatsciences, flatstandards, flatarcs.*

3.6 Identifying Arc Lamp Emission Lines

This is the first step that requires assistance from the user. It involves identifying the emission lines in the flat-fielded two-dimensional arc lamp image, and makes use of the PySALT task *specidentify*. This step will be re-run for each flat-fielded arc lamp image filename in the dictionary *flatarcs*, and it will result in a text file containing the wavelength solution for that particular GR-ANGLE; this text file (.db file) will be added to a dictionary *wavesols* automatically by the pipeline.

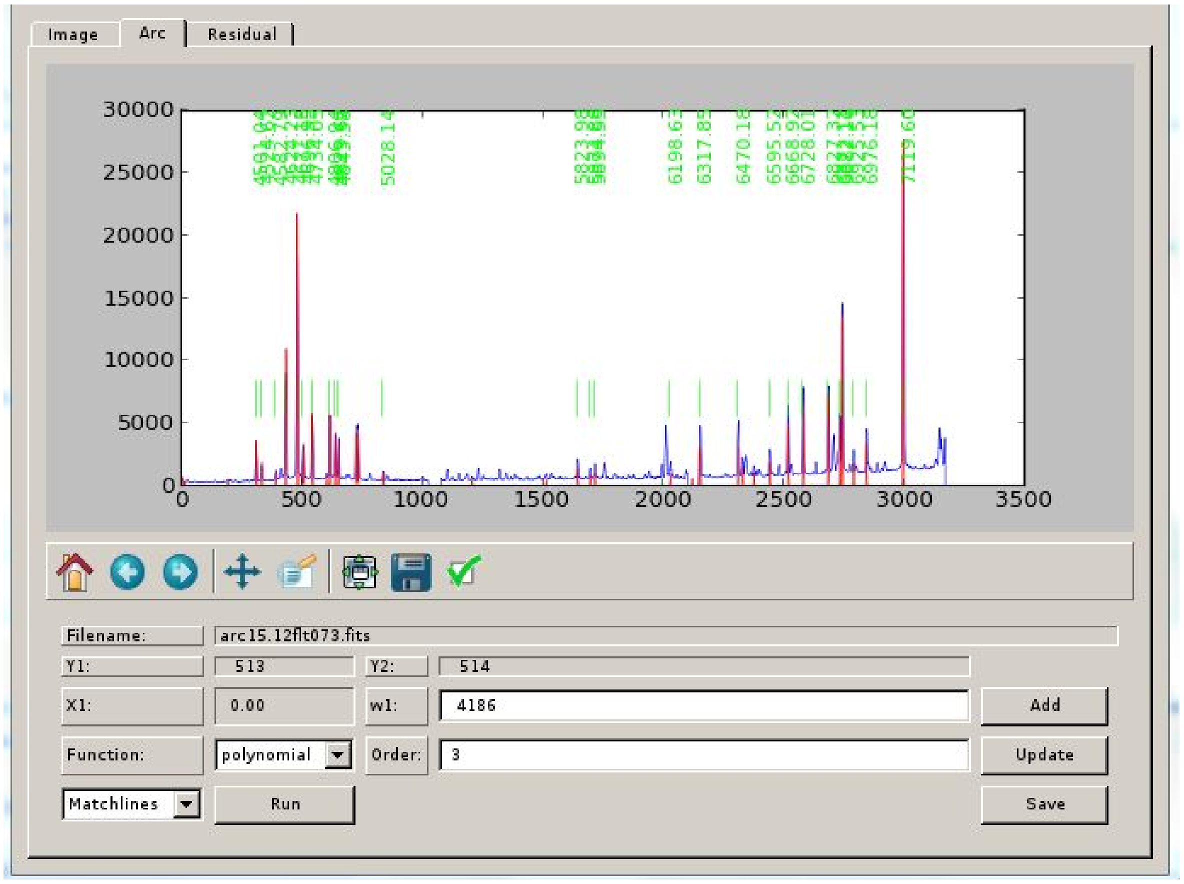


Filename:	arc15.12flt073.fits			
Y1:	513	Y2:	514	Update
nrows:	1	rstep:	100	Next

Figure 1: The "Image" tab of the PySALT task, *specidentify*.

What you see in 1 is the “Image” tab of the *specidentify* pop-up window. It shows the two-dimensional flat-fielded arc lamp image as well as a task-added green row. Your goal is to identify the emission lines at this middle (default) row, three rows above it, and three rows below it for a total of seven identifications.

To begin the identification process, click the “Arc” tab at the top of this same window. That should lead you to the actual one-dimensional spectrum of the data on that particular row. We want to transform the pixel number corresponding to each column on that row to a wavelength number (in Angstroms). To do that, first, read in the line list by pressing **a**, and then fit a zero-point solution by pressing **z**. Show the emission line wavelengths by pressing **b**, and press **f** to fit the template spectrum to the actual spectrum. You should see something like 2.

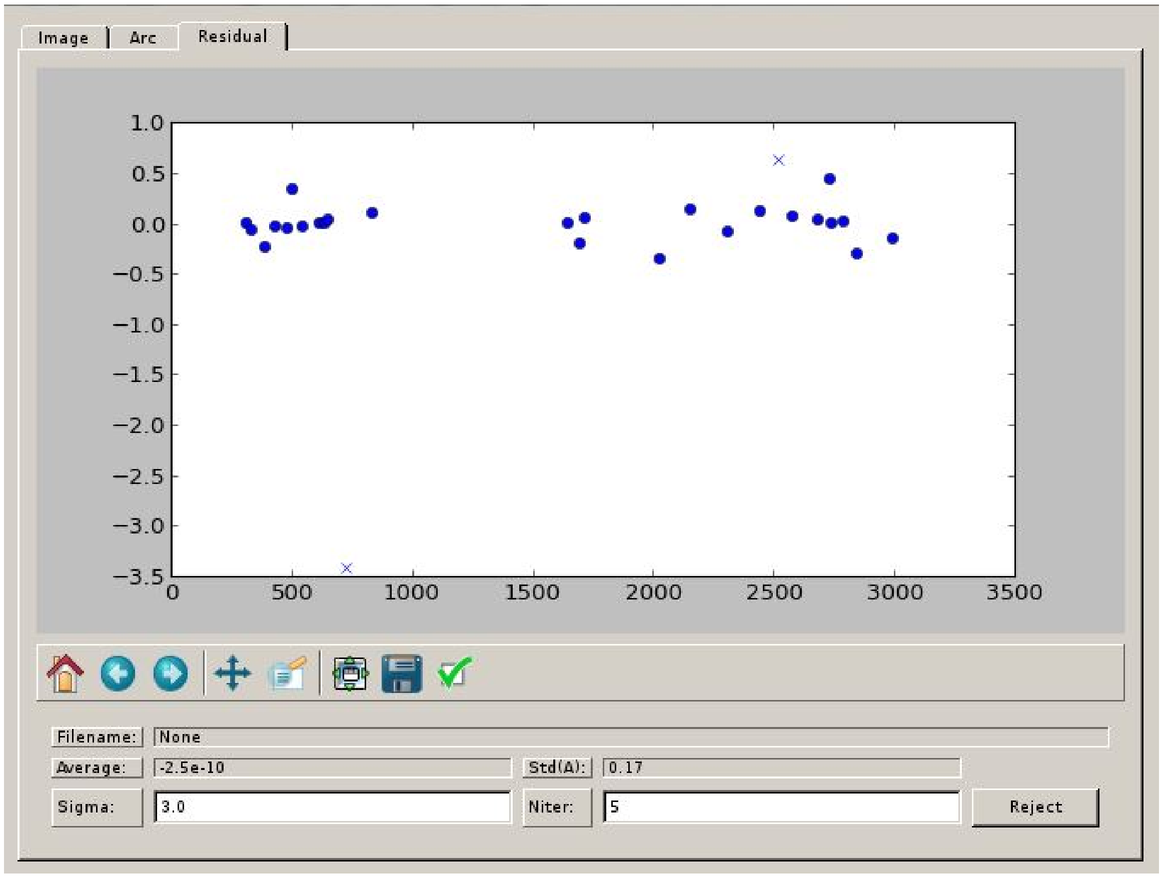


Filename: arc15.12flt073.fits
Y1: 513
Y2: 514
X1: 0.00
w1: 4186
Function: polynomial
Order: 3
Matchlines

Figure 2: The “Arc” tab of the PySALT task, *specidentify* after pressing the following sequence of keys: **a,z,b,f**.

This is a preliminary wavelength solution, but there are probably emission lines here which don’t have a very good fit. To fix that, click the “Residual” tab at the top of the same window to lead you to 3. The goal is to reach an RMS between 0.20 and 0.30. Delete any outliers (lines) by hovering over the circle and pressing **d**. Then, re-fit the wavelength solution by pressing **f** and look at the RMS. Once you have reached a suitable RMS, return to the “Arc” tab and hit the **Save** button near the bottom-right. Back in the aschere terminal window, you should have been notified that this particular row’s fit was written to a text file.

If you still have more rows to identify, go back to the “Image” tab, and either hit **Next** to go up (or down) one row by **rstep** pixels. If you have reached the third row above the middle row, and you want to begin your descent, simply change **Y1** and **Y2** to the first row below the middle



Filename: None
Average: -2.5e-10
Std(A): 0.17
Sigma: 3.0
Niter: 5

Figure 3: The "Residual" tab of the PySALT task, *specidentify*.

row, and **rstep** to its negative equivalent.

Once you have identified wavelengths at seven rows, you can manually close the *specidentify* window. Either a new one will pop up for the next flat-fielded arc image filename in *flatarcs*, or the pipeline will proceed to apply the wavelength solutions to the appropriate images.

3.7 Rectifying Science, Standard Star, and Arc Images

This step calls the PySALT task *specrectify* to apply the wavelength solutions stored in the newly created .db files to the appropriate images. It is completely automated. The resulting images are two-dimensional flat-fielded images whose x-axis values are no longer pixel numbers, but rather wavelengths. The resulting image's filenames are added to one of the following dictionaries based on the original type: *wavearcs, wavesciences, wavestandards.*

3.8 Subtracting the Background

SALT is a ground-based observatory, and so it is inevitable that the data it records will be contaminated by emission lines from the sky. These emission lines appear as vertical lines in the two-dimensional images. They can be removed by interactively choosing the background level to be subtracted using the PyRAF task *imred.generic.background.* When this task is run, a window resembling 4 should appear. You want to set two background sample regions: one to the left of your science spectrum, and one to the right of your science spectrum.

It is important to note that these are *sample* regions because this is all being done on a particular ensemble of columns (perpendicular to the dispersion axis). Ideally, with a bright science spectrum, you should see relatively low fluxes everywhere except for the relatively huge and wide spectrum: the width of this spectrum corresponds, of course, to the number of rows that it spans in the two-dimensional image.

In order to prevent unnecessary spikes in the interactive background fitting window due to sky lines and cosmic rays, the two-dimensional image will first be opened up in the program **ds9**. When it opens up, there are several things you need to do and ideally remember or *write down*:

1. Find the center row of your *science* spectrum. This can be very easy if the SALT astronomers mentioned that no other sources were on the slit during the spectroscopy process. However, if there are other sources, then you should use the acquisition images and make the required calculations (if any) to locate your spectrum. Generally, it can be distinguished from other spectra by looking at the distance (in arcseconds) between it and a bright source on the slit in the acquisition image. (Spectra generally span several rows, so just find the central row.)

2. Is your science spectrum relatively bright throughout the image, or does it appear very faint – perhaps even fading into the background at times?

3. Find a column in the two-dimensional image that is away from sky lines, and that doesn't contain any cosmic rays. This will ensure that the background subtraction will be a smooth process.

After you are finished inspecting the two-dimensional image and have committed the above three things to memory or print, close the **ds9** window. The background selection window resembling 4 should appear. If you chose a column without any cosmic rays and far away from sky lines, then this should be a relatively easy process. First, choose a background sample region to the left of your spectrum[5] by pressing **s** a bit further out from the left base of the spectrum. You'll need to hit **s** again to define the background region: continue onwards to the left of where you first pressed **s** until you get close to the edge of the entire spectrum (where it makes a nearly vertical descent), and hit **s**. Do the same to select a background region to the right of your spectrum.

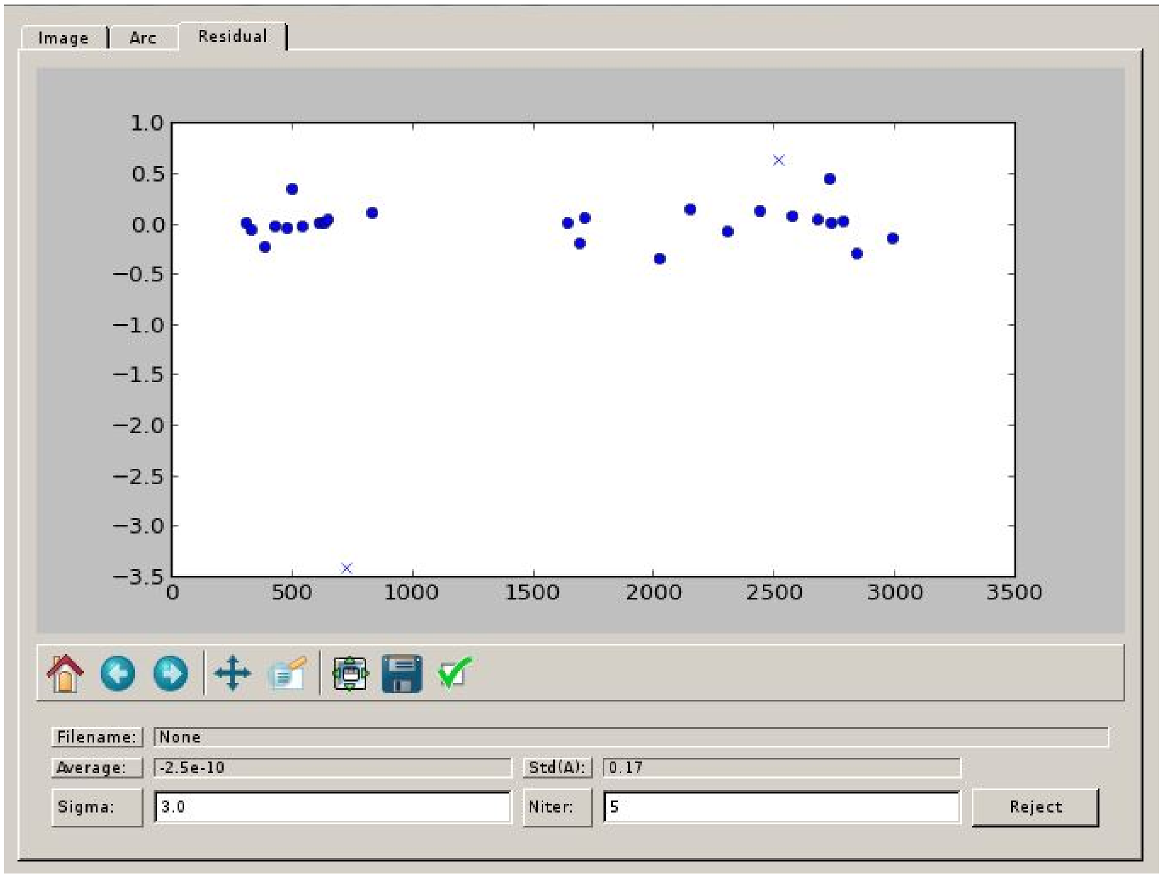
Then, hit **f** to fit the background level. Ideally, you want the background to fit the rest of the spectrum well, to not do anything very crazy in terms of fit within or beyond the spectrum, and to be at the base level of your science spectrum in particular (as opposed to going through it halfway up towards the peak). If you need to increase the order, type **:order 2** and increase the order number again if you need to (generally, you want to fit low-order background functions).

If you chose a column with cosmic rays, then you just want to make sure that you don't choose any background sample regions that include the bright sources. Otherwise, when you try to fit the background, you won't get an appropriate function.

You can delete background sample regions by pressing **z**. You can zoom in by pressing **w** – after you do this, you need to define the boundaries of your zoomed-in window. PyRAF wants you to first define the bottom-left corner by pressing **e**, and then the top-right corner by pressing **e** again. To zoom out, press **w** and then **a**.

Since background-subtraction will only be done for the science images, the resulting background-subtracted science image filenames will be added to the dictionary *backgroundsciences*.

[5]If your spectrum is very faint, i.e., a "bump" compared to the rest of the spectrum, then just choose background sample regions around the brightest spectrum source. The goal is to not include any bright sources as part of the background because they are not, of course, part of the background.



Filename: None
Average: -2.5e-10
Std(A): 0.17
Sigma: 3.0
Niter: 5

Figure 4: Interactively setting the background region to be subtracted from the image.

3.9 Extracting Science, Standard Star, and Arc Images

Until now, we have been working with two-dimensional images, but we wish to eventually have one-dimensional spectra. It is this step that will result in one-dimensional spectra using the PyRAF tasks *apall* and *apsum*. This step is relatively simple if the supernova spectrum is quite bright and thus easily visible along some particular ensemble of rows in the two-dimensional science images. However, if it is a very faint line that perhaps even disappears from time to time into the background, then we shall have to improvise our spectral extraction procedure.

The spectra of standard stars and arc lamps are very easy to extract with *apall* and *apsum*, respectively, and so the user will not extract them manually. The pipeline extracts them automatically and adds the resulting spectra's filenames to one of the appropriate dictionaries: *extractedstandards* or *extractedarcs*.

Supernovae, on the other hand, are the main objects of concern, and their spectra can vary in brightness depending on a number of factors. It is important to extract the correct spectrum, and so it is necessary to inspect the two-dimensional image, look for the science spectrum, and find out whether it is bright or faint. If the pipeline already consulted the user about this information during the background subtraction (previous) step, then it will not do so again; otherwise, if the extraction is being done as an individual reduction step, the pipeline will open the **ds9** program, and the user should consult the acquisition images provided by SALT to find the science spectrum perhaps among many others. Once this is done, the user can close the **ds9** window manually and return to the pipeline prompt.

Since the user has now identified the supernova spectrum, remembered or written down the central row number of the spectrum, and decided whether it looks faint or bright, the pipeline will ask the user whether the spectrum looked faint or bright. This question is asked because the PyRAF task *apall* is run with a different set of parameters depending on the brightness of the supernova spectrum. The pipeline will ask the user to enter 0 if the spectrum looked faint, or 1 if it looked bright. Upon entering the appropriate number, the extraction procedure can go one of two ways. Both shall now be described.

3.9.1 Bright Spectrum

If the user indicated that the spectrum was bright by entering 1 at the prompt, then the pipeline will set the optimal extraction parameters and run *apall*. A window resembling 5 should appear. First, hover over any spectra that are marked for extraction and press **d** to delete those marks. Then, find your spectrum – it should be relatively easy to find since its bright, although there might be a few other bright ones so be careful to choose the correct spectrum – and press **m** to mark it for extraction. You will probably have to extend the length of the extraction region: use **l** to set the lower or left limit, and **u** to set the upper or right limit.

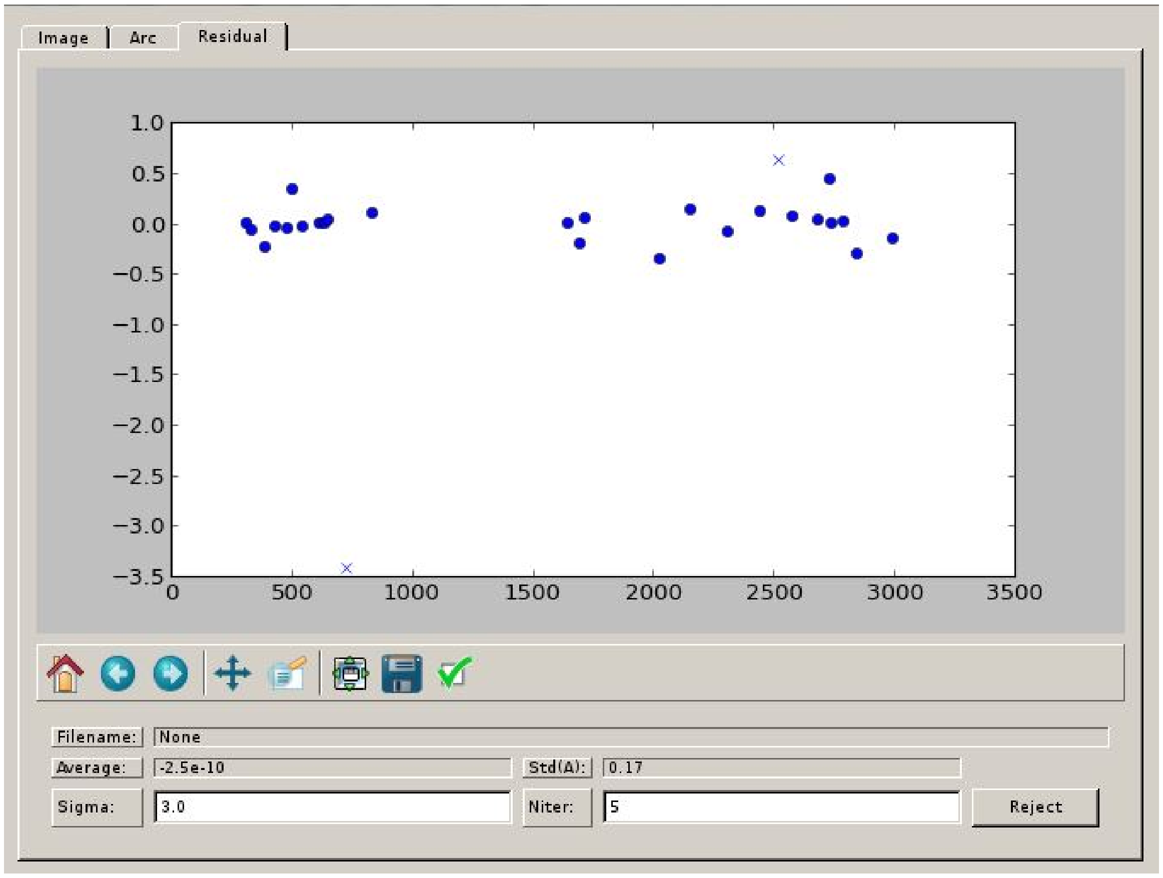
Next, hit **b** to bring up the background selection window. If you have already done the background subtraction (previous step), then hit **z** to delete all background sample regions. Otherwise, use the previous step's instructions to select a background sample to the left and to the right of your scince spectrum: press **s** twice to define each sample region, **z** to delete sample regions, **f** to fit or refit the background level, and **:order 2** (or another number) to increase the order of the background fitting function. When you're done, hit **q** twice to go to the extraction fit window. Say yes to the questions that follow.

You should check that the range of rows (the y-axis) isn't very high: your supernova spectrum probably isn't so slanted in the two-dimensional image that it spans hundreds of rows. Generally, the supernova spectrum might be slanted from left to right in the two-dimensional image, and this slant should only span a couple of rows. The numbers on the y-axis should also be centered around and/or include the actual central row of the supernova spectrum that you identified in the two-dimensional image. Otherwise, the correct spectrum wasn't marked for extraction in the previous step, and you will have to re-run that step.

The order of the fit might have to be increased; type **:order 5** or a lower/higher number if needed. If the spectrum only spans one or a few rows, the order doesn't need to be high. You can delete outlier data points by pressing **d**, and undelete them by pressing **u**. You should refit the extraction function after making changes by pressing **f**. Once you're done, hit **q** and answer 'yes' to anything to write and create the extracted spectrum file.

3.9.2 Faint Spectrum

If, on the other hand, the user indicated that the spectrum was faint by entering 0 at the prompt, then the pipeline will run the PyRAF task *apall* with a different set of parameters, and a window resembling 6 should appear. The supernova spectrum will probably not be visible, and may even be lost in the noise. The user will now need to use the information learned earlier from inspecting the two-dimensional image in **ds9**: press **w** to call the zoom function, define the lower-left corner of the zoomed-in window by pressing **e** to the bottom-left of the central row of the *supernova* (not entire window!) spectrum, and define the top-right corner of the zoomed-in window by pressing **e** to the top-right of the central row of the spectrum. A tiny bump should be visible at the supernova spectrum location. Press **l** just to the left of the left-base of the supernova spectrum, and then



Filename: None
Average: -2.5e-10
Std(A): 0.17
Sigma: 3.0
Niter: 5

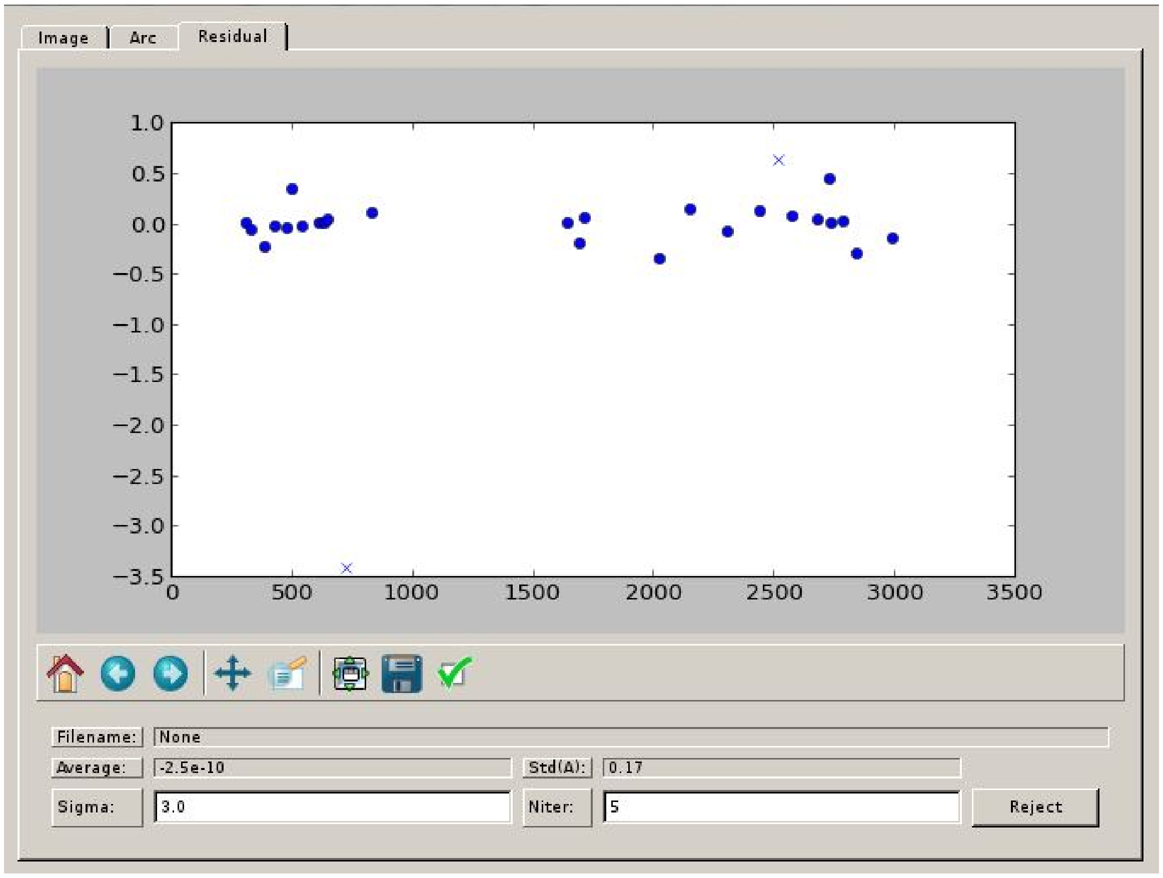
Figure 5: Selecting a bright spectrum for extraction.

press **u** just to the right of the right-base of the supernova spectrum. This defines your supernova spectrum rows, or width for the extraction.

If the background was already subtracted by the previous step, there is no need to fit the background again. Thus, hit **b**, and use **z** to delete any background sample regions. Otherwise, define them properly as outlined in the previous step's subsection; you still want to choose background sample regions around your supernova spectrum. When you're done, hit **q** twice to go to the extraction fit window. Say yes to the questions that follow.

You should check that the range of rows (the y-axis) isn't very high: your supernova spectrum probably isn't so slanted in the two-dimensional image that it spans hundreds of rows. Generally, the supernova spectrum might be slanted from left to right in the two-dimensional image, and this slant should only span a couple of rows. The numbers on the y-axis should also be centered around and/or include the actual central row of the supernova spectrum that you identified in the two-dimensional image. Otherwise, the correct spectrum wasn't marked for extraction in the previous step, and you will have to re-run that step.

The order of the fit might have to be increased; type **:order 5** or a lower/higher number if needed. If the spectrum only spans one or a few rows, the order doesn't need to be high. You can delete outlier data points by pressing **d**, and undelete them by pressing **u**. You should refit the extraction function after making changes by pressing **f**. Once you're done, hit **q** and answer 'yes' to anything to write and create the extracted spectrum file.



Filename: None
Average: -2.5e-10
Std(A): 0.17
Sigma: 3.0
Niter: 5

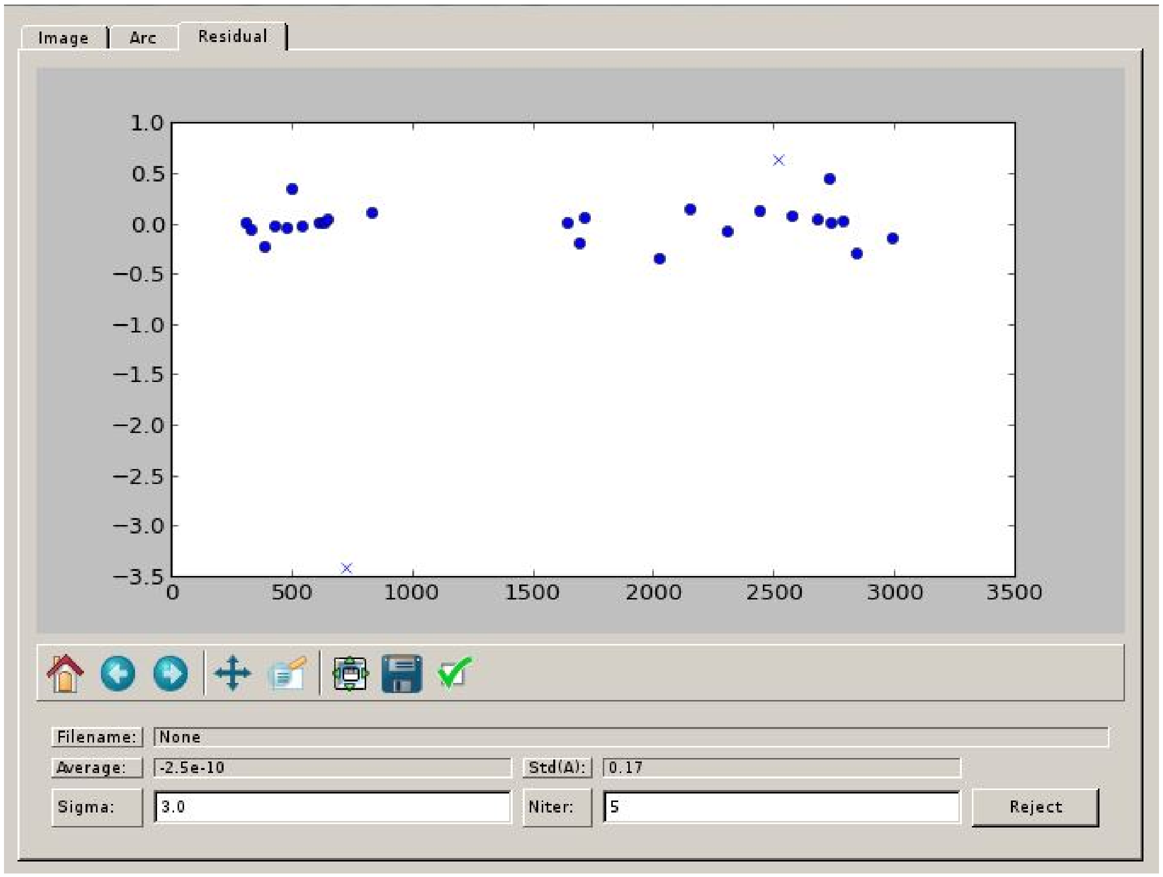
Figure 6: Selecting a faint spectrum for extraction.

3.10 Re-Identifying Arc Lamp Emission Lines

Now that we have one-dimensional spectra instead of two-dimensional images, we can use the PyRAF task *identify* to test the first wavelength solution. The arc lamp spectra will be opened in a window resembling 7. Simply hit **l** to read in the linelist, and then press **f** to see the fit. Since the spectra are already wavelength-calibrated (the x-axis is in terms of wavelengths and not pixels), the initial wavelength solution shouldn't have a very high RMS; the y-axis range for the errors should span only a few Angstroms. Feel free to delete any outliers in the fit window by pressing **d**, undelete any data points (lines) by pressing **u**, and refit the new wavelength solution by pressing **f**. Hit **q** when done to return to the spectrum, and **q** again to exit *identify*. Answer 'yes' when asked if you want to save the new wavelength solution. There are no text files produced by *identify*; instead, the task (and pipeline) links up the identified arc image with the appropriate science and standard star images by adding keywords to the headers of the latter two images.

3.11 Applying Second Wavelength Solution to Spectra

After running *identify*, the pipeline linked the identified arc image to its corresponding (same grating-angle) science and standard star images so that the PyRAF task *dispcor* can apply the wavelength solution. This step is done non-interactively and results in dispersion-corrected (same as wavelength-calibrated) images which are added to one of the appropriate directories: *dispsciences* or *dispstandards*.



Filename: None
Average: -2.5e-10
Std(A): 0.17
Sigma: 3.0
Niter: 5

Figure 7: Creating the second wavelength solution with the PyRAF task *identify*.

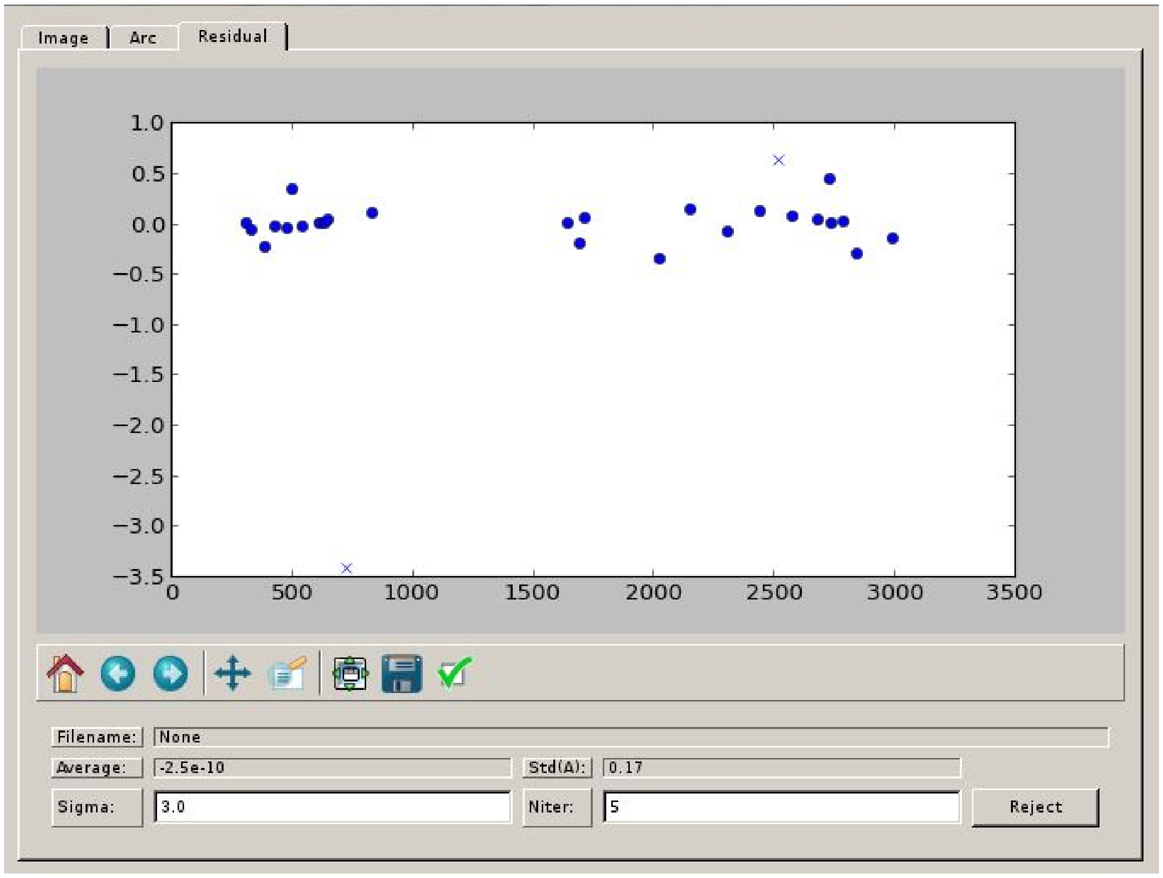
3.12 Flux Calibrating Science Spectra

The last major step of the reduction process involves transforming the y-axis of the spectra from pixel brightness values ("counts") to real flux values, i.e., erg/cm^2/s/$\AA$. During this part, the pipeline will ask you to input the name of the standard star. The pipeline will actually print out the name, but this is an abbrevation that SALT uses; the abbreviation that PyRAF uses is slightly different. Enter **?** when prompted for the star name, look through the list and find the PyRAF abbreviation that corresponds to the SALT abbreviation[6], and then enter that abbreviation. It is pretty straightforward, and in fact, the pipeline even channels its "inner-Gandalf" when giving you instructions about how to proceed.

After that, the PyRAF task *standard* opens up a screen resembling 8. The goal is to add about fifteen bandpasses on the standard star's spectrum using **a** twice to define them, but to not place any bandpasses over the two chip gaps or telluric absorption lines (very big dips in the spectrum). Use **d** to delete bandpasses, and press **q** when done. This task will produce a text file added to the dictionary *stdfiles* – the text file contains data about the standard star, in particular a mapping from counts to real flux values.

Next, the task *sensfunc* will use the data created by the task *standard* to produce a sensitivity function in the form of an image. A screen resembling 9 should pop open. You may delete any outliers using **d** and then pressing **p** to delete that single data point (as opposed to all of the data points corresponding to the bandpasses chosen by *standard* on the standard star's spectrum). Press

[6]For example, SALT uses 'HILT600' whereas PyRAF uses 'h600'.



Filename: None
Average: -2.5e-10
Std(A): 0.17
Sigma: 3.0
Niter: 5

Figure 8: Manually selecting bandpasses for flux calibration using the PyRAF task *standard*.

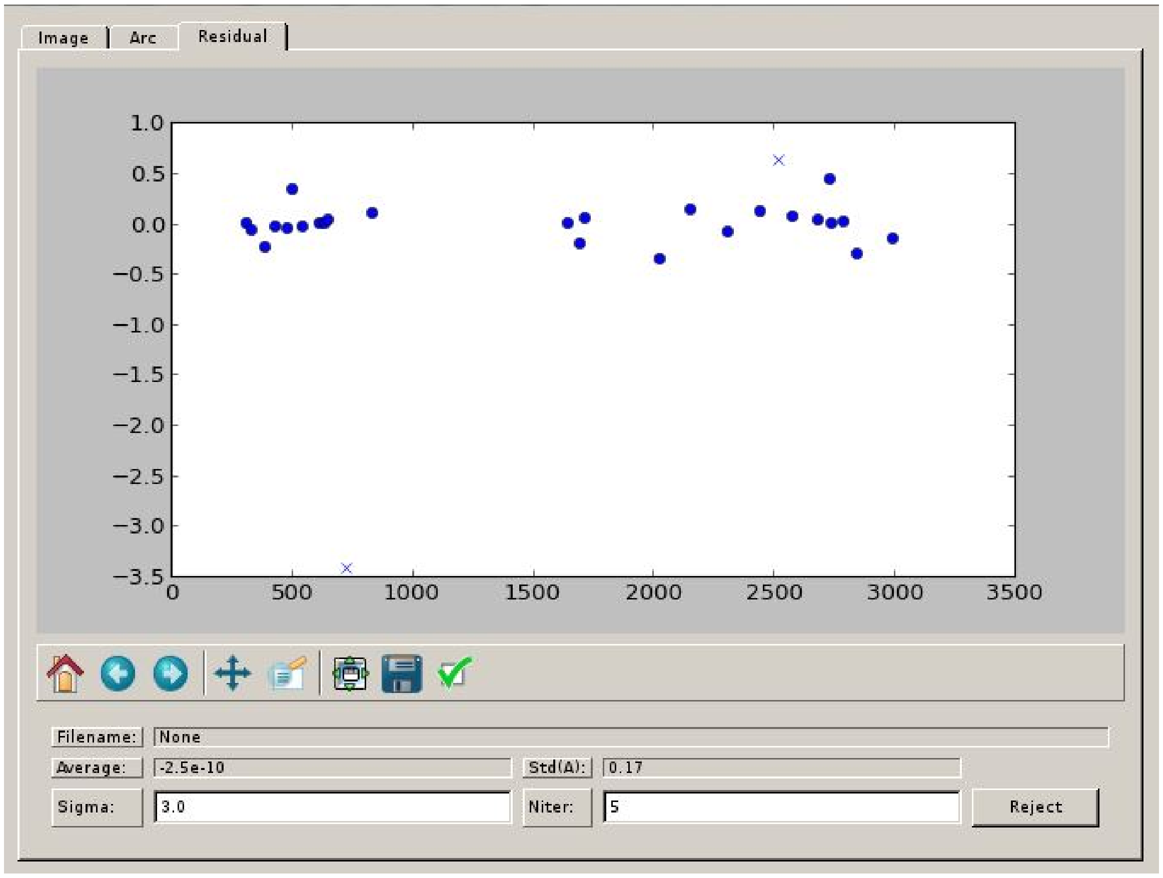
g instead of **f** to refit the sensitivity function if necessary. Generally, the *sensfunc* task produces a satisfactory sensitivity function image: see the fit of the points in the top screen, and the spectrum in the bottom-right screen. Hit **q** after verifying that the fit is good. Don't change the order of the function; order six seems to be necessary here. The task will produce a sensitivity function image added to the *sensfuncs* dictionary.

The task *calibrate* applies the flux-calibration sensitivity function non-interactively, and adds the resulting flux-calibrated image filenames to the dictionary *fluxsciences*.

3.13 Combining Science Spectra

The only thing remaining is to combine all the different spectra at the different wavelength (grating-angle) settings with the PyRAF task *scombine*. This is done automatically and non-interactively, but then the pipeline opens the final combined spectrum using the PyRAF task *splot* so the user can see it. Furthermore, the user is encouraged to get rid of any remaining cosmic ray spikes in the image (quite obvious from the flux of the spike relative to the rest of the spectrum) by connecting the left and right bases of the cosmic ray spike: hit **x** twice to connect the bases. Press **i** when done and enter the image's filename to overwrite the old combined spectrum (or write a new filename to create a new spectrum).

That is it! That's the end of the spectral reduction process.



Filename: None
Average: -2.5e-10
Std(A): 0.17
Sigma: 3.0
Niter: 5

Figure 9: Verifying the sensitivity function created by the PyRAF task *sensfunc*.

4 Limitations of and Extensions to the Pipeline

This section addresses bugs in the current version of the pipeline and examines where (and perhaps how) the pipeline could be improved in the future.

4.1 Known Bugs

As of June 2, 2013:

1.

4.2 Future Work

As of June 2, 2013:

1. Incorporate lacosmicx to remove cosmic rays.

Acknowledgements

The Rutgers SALT pipeline was written, debugged, and augmented during Spring 2013 and Summer 2013 with the help of Professor Saurabh Jha, Curtis McCully, Thomas Holoien, Yssavo Camacho, and Brandon Patel.

References

[1] S. Crawford, M. Still, P. Schellart, et al. PySALT: The SALT Science Pipeline. 2010. `http://www.salt.ac.za/fileadmin/files/science/Publications/7737-82-PySALT.pdf`.

[2] P. Massey and M. Hanson. Astronomical Spectroscopy. 2011. `http://arxiv.org/abs/1010.5270`.

[3] P. Massey, F. Valdes, and J. Barnes. A User's Guide to Reducing Slit Spectra with IRAF. 1992. `http://iraf.net/irafdocs/spect/`.

[4] F. Valdes. Guide to the Slit Spectra Reduction Task DOSLIT. 1993. `http://iraf.net/irafdocs/doslit/`.