A User's Guide to Multislit Spectroscopic Reductions with IRAF

E. Ellingson

November 1, 1989

Abstract

This document is intended to guide you through the basic principles of the reduction of multislit spectroscopic data using IRAF. It discusses the procedures necessary to take your data from the telescope to wavelength calibrated, one-dimensional spectra ready for whatever further analysis you choose. Though the primary emphasis here will be on reducing slitlet spectra taken with the KPNO 4-m telescope and cryogenic camera, many of the procedures are also useful for reducing other types of slit data, aperture "hole" data and multi-fiber data.

Contents

1 Introduction

This document is intended to guide you through the basic principles of the reduction of multislit spectroscopic data using IRAF. It discusses the procedures necessary to take your data from the telescope to wavelength calibrated, one-dimensional spectra ready for whatever further analysis you choose. It assumes that you are using IRAF Version 2.8.

If you are a new IRAF user it is recommended that you first read the document "A User's Introduction to the IRAF Command Language" by Shames and Tody, which can be found in Volume 1A of the 4 blue binders that compose the IRAF documentation. Simultaneous graphics and image display of your multislit data is extremely useful in the reduction process; if you are reducing your data on a Sun workstation the manual "A Quick Look at IRAF on the Tucson Sun Network" will help in getting you set up to take advantage of the workstation's capabilities.

It is assumed that you are familiar with some of the basic IRAF routines, for example examining and changing a task's parameter file using **epar**, using **implot** to graphically examine a two-dimensional image, and displaying a two-dimensional image. If you are new to IRAF, after you read the " User's Introduction" you should play around a little and learn how to do these things.

The reduction of multislit data, especially of faint or extended objects, can be difficult, and the reduction parameters will probably require a lot of "tweaking", depending on the quality of your data, and the results you wish to obtain. Because each data set will have its individual problems and area of concern, this guide presents the general reduction procedures, along with specific recommendations for how to deal with different sorts of data. There is no substitute for really understanding your data and the reduction procedure, however, and the user should proceed with caution and thought.

2 Calibration Frames

The following list contains the type of calibration images you may need in reducing a typical night of multislit observations:

bias frames. These are zero second integration exposures obtained with the same pre-flash (if any) you are using on your program objects. It is a good idea to take a set of 10-20 bias frames each night.

quartz exposures. Exposures of quartz lamps are used to normalize the response of the CCD + aperture mask system. This normalization comes in two parts: the high spatial frequency pixel-to-pixel response of the CCD, and the relatively uniform pattern of illumination of slits (the "slit function"). Because of flexure in the instrumentation, it is probably best to take quartz exposures before and after each exposure of your program objects, and of course you will need at least one for each aperture mask you are using. Multi-fiber and aperture "hole" data do not need quartz calibrations, but dome flats (see below) are necessary for correcting for the overall response of the fiber or hole.

dome or sky flats. The illumination provided by the quartz lamps is not identical to that of the sky, and in some cases, short exposures of a dome flat are needed to correct the slit function. These may not be necessary in some cases, but, as in the bias calibration frames, it is better to take them than not. They are crucial in the reduction of data where the sky background is not determined from the same slit (e.g. multi-fiber data, aperture "holes"). The dome flats may also be augmented by sky flats; the illumination pattern provided by the dome flats is also not identical to the sky illumination, though it is usually adequate.

comparison frames. These short exposures of arc lamps are necessary to determine the wavelength scale of each slit. They should ideally be taken before and after each exposure, especially if velocity information is important to you. Some instruments (notably the KPNO NESSIE multi-fiber spectrograph) also require exposures of a second lamp, to determine zero-point shifts in the wavelength solution.

Note that flux calibration is not usually attempted with multislit data, and hence no standard star observations are necessary.

3 Overview

The general procedure used in reducing raw data from the telescope to wavelength calibrated one-dimensional spectra can be divided into three parts: instrumental normalization, extraction of a one-dimensional spectrum from each slit, and wavelength calibration.

The instrumental calibration process consists of the subtraction of the CCD bias using the bias frames and the overscan region of each frame, and the removal of the CCD non-uniformities using the quartz frames.

The extraction of the one-dimensional spectrum starts by defining the properties of each slit. These properties are the position of the object in the slit, the spatial size of the object, and the "curvature" of the object position as a function of dispersion (sometimes called S-curvature). In some cases, background subtraction from adjacent pixels within the slit is required and these regions also need to be defined. The signal within the defined aperture is then summed for each dispersion line, and the background for that line is subtracted. The net is your extracted spectrum.

Wavelength calibration requires first that you create similar extracted spectra for each slit of each comparison frame. The dispersion correction (a.k.a. the wavelength calibration, or wavelength solution) is determined for each slit using one of the comparison exposures. These are then used to determine the (very slightly different) dispersion corrections for each of the other comparison frames. The proper combination of these dispersion solutions is decided for each object exposure (i.e. use the average of the comparison taken before the exposure and the one taken after). Finally, each individual object spectrum is wavelength corrected, and rebinned.

Figure 1: Parameters for **rfits**

Because extracting many individual spectra from each CCD frame (not to mention all of the comparisons that you need) creates a frightening number of individual files, the multispec image format has been created to keep the number of files under control and to simplify the bookkeeping. The image format consists of a two-dimensional image where each line is an individual spectrum from each of the slits. This greatly simplies the matter of summing exposures taken through the same aperture mask, comparing and plotting spectra etc. Finally, if you wish, these multispec format files can be disassembled to "normal" one-dimensional spectra for further analysis.

4 Getting Started

4.1 Reading In Your Data From Tape

The first step is to transfer your CCD frames into "IRAF images". These will consist of IRAF "header files" with the extension ".imh", and IRAF "pixel files" with the extension ".pix". The latter will be found in your image directory (**show imdir** will show you where this is), while the headers will reside in whatever directory you happen to be in when you create the pictures. The header files contain the information about where the pixel data resides on disk, and when you refer to an image by its name without the extension, the ".imh" is assumed.

The first step in reading in your tape is to find an available tape drive, and figure out what its IRAF name is. Usually the names are **mta**, **mtb**, etc., but if you are on a network you may also need to refer to the machine that the tape drive is attached to, such as **orion!mta**, **tucana!mta**, and so on. To allocate the tape drive type **all tucana!mta** or somesuch. Next, put the tape on (you might want to remove the write-ring first!) and fiddle with the tape drive until the tape is on-line and at the load-point. If your tape is in FITS format we will use **rfits** in the **dataio** package, so type **dataio** and then **lpar rfits** to see what the parameters are. Modify the parameters using **epar rfits** until they resemble those shown in Figure 1.

Typing **rfits tucana!mta 1-999 multi > headers &** will put the task in the background (freeing up the terminal so you can do useful things like read your e-mail and play computer games) and write the header information into a file called "headers". You can check its progress by typing **tail headers**. If you lose track of any task you have set running in the background, typing **jobs** will tell you whether it is still running, and for how long. When the tape is read, all the files on your tape will be copied into files with names like **multi001, multi002, multi003, multi004.....** If you wish to read in data from more than one tape, then you can **rewind tucana!mta** when you are done, exchange the second tape for the first (note that this way you don't run the risk of losing your tape drive!), and then execute **rfits** again this time inserting the current picture number for the "offset" parameter. When you are done, do a **deallocate tucana!mta** to release the tape drive, and a **bye** to get rid of the "dataio" package. If your tape is in "CAMERA" format rather than "FITS" format, then you will need to load **noao** and **mtlocal** and use **rcamera**; the syntax is similar to that of **rfits**.

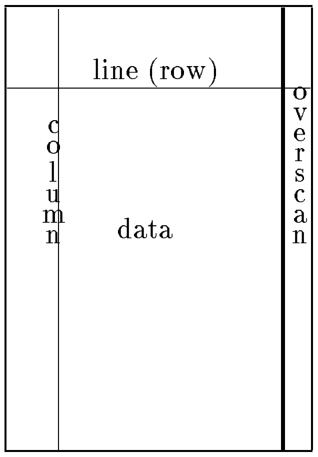
5 Bias Subtraction

The first step in reducing any CCD data is subtracting the bias. This "bias" is a pedestal level of several hundred ADUs which is added to the output signal of the CCD when it is read out. The absolute level of the bias is dependent on telescope position, chip temperature etc., so we need to determine the overall bias level for each frame individually. This is done by determining the level of the "overscan" region, extra pixels with no charge from the exposure. The level of the overscan for each exposure is used to determine the average bias level, and is subtracted from each exposure. The overscan is also subtracted from the bias frame, leaving the pixel-to-pixel bias pattern. This bias frame is also subtracted from each of the the data frames.

Subtracting the bias from multislit frames is no different than doing it for any other sort of CCD data, so here we can use the standard IRAF reductions package. This guide will step your through the procedures necessary for most multispec data, but for a more comprehensive description, you will find "The User's Guide to CCD Reductions with IRAF" very useful. First, load this package by typing **imred** and then **ccdred**. The task you'll be using here is called **ccdproc**. This task in its entirety is extremely complicated, but luckily we only want to subtract a bias frame, and therefore don't have to worry about most of its subtleties. Type **setinstrument**, enter "cryocam" for instrument type, then **CNTRL Z** twice to exit. More on this later.

Now, look at your data and find your bias frames. Do you need to combine several individual frames to create a single bias (one per night, usually)? If so, use **epar** to look at the task **zerocombine**. ("Zero" is the IRAF translation of "bias.") This task will combine your biases together in a reasonable fashion. You might want to type **zerocombine multi001, multi002,multi003,multi004** and let it go, but if you have more biases than you'd like to type, try this trick: first type **files multi* > biases**. An alternate command for those who have the correct type of exposure (i.e. zero, object) in the image header would be **ccdlist multi* ccdtype=zero name+ > biases**. This creates a text file called "biases" which contains a list of images that start with “multi.” Edit this file (**edit biases**), deleting lines until all that is left are the names of files that really *are* bias frames (remember- you can check your file “headers” for this). Then type **zerocombine @biases** (using the default combine option set to “maxreject”) and it will use the contents of the file “biases” as an input list, creating a bias frame called “Zero”. Remember this way of using “ @ ” – it will be useful later. If you have more than one night’s worth of data, you need to be careful to change the output bias image name for each night. or possibly create a new directory (**mkdir nite1** etc.) for each night, move your images to the appropriate directory (**imrename @nite1images nite1/**) and do the entire bias subtraction process separately for each night.

Let’s take a look at the data now. Use **implot** to look at one of your quartz frames. Using **Z** to zoom, or one of the windowing commands (type **?** for a list of commands if you’re lost here), find the overscan region of the frame. This is usually a strip 32 pixels wide on the righthand side of the frame. You can tell where it is because the ADU level drops a little bit there, especially if you have pre-flashed the chip or have a problem with scattered light.



Using the **C** key (you may have to hit it twice), note the left and right extents of the overscan region and write them down. Also note the left and rightmost extent of your spectra (not each individual spectrum– just the “wasted” area to either side of them all). To save diskspace, we will be trimming the images down to their minimum sizes. Remember that each of your aperture masks will have a different pattern of slits and possibly different trim sizes! Look at a quartz from each of them and decide whether you want to trim them all to the same size, or trim the data from each aperture mask separately. If you see that the area between the slits is not flat, but rather has a noticeable shape, you may have a problem with scattered light outside of your apertures. This can be subtracted a little later in this process, so remember it, and do not trim your frames heavily (do trim away the overscan region). Some data will already have the bias section entered as a keyword in the image header. Look for it by typing **imhead multi001 long+**. In that case, if you decide you want to trust it (it is never a bad idea to check, however), you can leave the parameter **biassec** to be “image,” and it will read the overscan section from the image header.

Now it's time to delve into the parameters of the ccdred package. Again, type **setinstrument**. It will again query you for an instrument and you should type **cryocam** again. You will be confronted by a complicated-looking parameter list. Don't panic— just type **CNTRL Z** and look instead at the second "page" of parameters. These are the ones that control what the task **ccdproc** actually does.

We only want to subtract a bias here, so go down the parameter list and change the parameters so it *only* uses the overscan and the bias frame to correct the images. Figure 2 shows how it should look. Be sure to call the zero frame the right name if you are using data from different nights. Note the format for entering the overscan region and the trim size of the frames. It's not a bad idea to trim a few pixels off the top and bottom of the frame, as the data are usually a little weird there. Remember that if you want to trim data from different aperture masks to different sizes that you will have to run this task several times, changing the trim section parameter for each.

When you are satisfied with the parameters, a **CNTRL Z** will get you out, and

```
cl> ccdproc multi*.imh
```

will subtract the bias from all frames starting with "multi". Once again, remember that if you want to change parameters for objects of different aperture mask, or taken on different nights, you must run ccdproc several times with appropriate changes in the parameters and the input list. Using " @filen" for the input list, as described above, will be useful here. Ccdproc also reduces the data in place, so be very sure that you know what you are doing before you let it go. Changing the parameter **noproc** to "yes" will allow you to make a dry run and make sure it's going to work. Note that ccdproc is clever enough to realize that your bias frame needs to be overscan-corrected before it does anything else.

After your data have been bias-subtracted, use **implot** to take a look at a frame and make sure it's all right. While you are here, look carefully at the areas of the frame *in between* the apertures. Is it flat, or is there a significant amount of scattered light? This is not often a problem with cryocam data, but if you are concerned, you should read the section below on **Scattered Light** before proceeding with the normalization of your data.

6 Normalization

Normalization of multislit data is a tricky process because of the strange and extreme shapes caused by light falling through multiple slits onto one CCD frame. In general, there are two types of normalization that need to be done. One is high spatial frequency pixel-to-pixel response variations in the chip, and the other is relatively low spatial frequency illumination patterns caused by the slit mask (the slit function). We can get them both out, to some extent, by using high signal-to-noise exposures of a quartz lamp.

The quartz lamp is useful in that it has a smooth, well-behaved spectral shape that we can easily fit and divide out, leaving only the pixel-to-pixel variations. The *relatively* uniform

Figure 2: Parameters of **ccdproc**.

Figure 3: Headers for Dataset Used in these Examples

illumination of the lamp can also be used to correct for the slit function, though if you are concerned, you may want to use dome or sky flats for this purpose as well (this is *crucial* for multi-fiber data).

Before you do anything else, you will need to tell IRAF which way your spectra are oriented (i.e. which dimension is the dispersion axis). To do this type

```
cl> setdisp *.imh dispax=2
```

if your data are aligned with the spectra lying along columns, and **dispax=1** if they lie along rows. Now you will probably want to move your data (if you haven't already done so) into separate directories for each aperture mask. Use **mkdir mask1** etc. to create the directories, and **imrename @mask1list mask1/** to move the proper frames there. Next, you will need to load the packages **imred** and **msred**. The **msred** package contains the rest of the tasks you will need to become familiar with in this reduction. Type **help** and just take a look for a moment at what it contains in order to get oriented. From here on, I will be drawing examples from a 4-m cryocam+multislit dataset which consists of two exposures, along with three quartzes and three comparison exposures bracketing them in time. The names and headers for these exposures are in Figure 3.

6.1 Normalizing Multislit Data

At this time, the method of choice in normalizing multislit data first defines the positions and width of each slit, and traces the "curvature" of the quartz spectrum along the dispersion direction. A one-dimensional spectrum is determined from the average of the data in the aperture, and is fit with a smooth function to follow the shape of the quartz spectrum. This fit is divided into each line (or column) in the entire aperture, leaving only the pixel-to-pixel response. Because the entire slit is divided by a single function taken from the average of the aperture, this response also represents the relative illumination at different parts of the slit- the slit function. This process is repeated for each different aperture, and the output frame is used as a flat field.

In order to combat the problems caused by flexure, I hope you have taken quartzes (and comparisons!) before and after each long exposure. In that case, you will want to create an average quartz frame to use as the normalization for each exposure.

cl> flatcombine multi017,multi020 output=qava combine=average exp+ subs-

cl> flatcombine multi020,multi023 output=qavb combine=average exp+ subs-

This will combine the appropriate quartzes to normalize the two exposures, "a" and "b". If you're suddenly lost here, look again carefully at Figure 3. The frames are scaled by exposure time, in case you changed your flat exposure time during the night. Use the combine option "avsigclip" for three or more input frames to get rid of spurious pixel values.

If you looked at the "help" for the msred package, you will have guessed that we want to start with the task **apnormalize**. Look at the parameter file first with **epar apnorm**. The yes-no parameters will tell you roughly how the task works. First you find and define the aperture positions, then trace them (that means fit the S-curvature), redefine the normalization apertures, if you want, and finally fit the quartz shape out of the normalization. So let's do all of this for the first averaged quartz frame:

cl> apnorm qava flata

No, you don't want the task to bother finding the apertures for you, and yes you do want to edit the apertures. You are now in the basic aperture editing (apedit) mode, which is common to all of msred "ap" tasks. You will need to be familiar with it, so take your time now and play a little bit with setting the parameters. If you type **:show** (all : commands require a carriage return) you will be shown the apedit parameters. You can reset them by typing, for example **:nsum 50**, or **:line 100**. Single-character capital IMPLOT keys also work in this mode for zooming, windowing etc. The first thing you should make sure of is that the width of your slits is properly set. Use the **C** key to get the cursor positions that mark the edges of a typical slit, and then use **:width** to make sure that the width is set to a reasonable value. This is important if the trace is to work right. Because of defocussing across the spectrum (a major problem with some instruments!) it is also important that you check different lines in the frame and decide where you want to optimize the process.

After you've done that, you're ready to start defining apertures. The following single-stroke keys are useful in defining the position of the center of the slit, and the proper slit width.

m - defines and centers an aperture nearest the cursor

+ - moves cursor to next aperture

- - moves cursor back one aperture

. - moves cursor to nearest aperture

c - fits the center of an aperture

d - deletes an aperture

o - re-orders aperture numbers

l - sets the lower (lefthand) limit of an aperture

u - sets the upper (righthand limit) of an aperture

y - sets the limits of an aperture to where the horizontal cursor crosses the edge of the aperture

a - "ALL" toggle- does subsequent procedures on all apertures

You will want to place the aperture so the center is roughly in the center of the slit, and the limits are about halfway down the sides (Figure 4). Again, the **:width** should be set to about this width as well, and you should be careful to check the effects of defocussing across the chip. If your aperture mask has "setup" holes in it, ignore them. When you are done, type **q** to proceed. Yes, you want to trace, and you want to trace interactively. This may take a moment, so it's a good time to stretch your legs.

When the curvature tracing has been done, answer "yes" to all queries (or "YES" if you know you will want to do that operation for all apertures). You will be shown the trace and apnorm's first guess at a fit to it (Figure 5). This fit will be the S-curvature assigned to that particular aperture. At this point you are in another general IRAF mode with which you will have to become comfortable - ICFIT, which stands for interactive curve-fitting mode. When in this mode, your job is to set the fitting parameters so that the result is a fit that you like. Useful commands are:

:fun spline3 - set the fitting function to a third-order spline. Other options for the fitting function can be found by hitting **?**.

:or 5 - change the order of the fitting function

:nit 1 - iterate the fit once, throwing out deviant points (see rejection threshholds below)

Figure 4: Quartz Apertures

:high - high rejection threshhold (in sigma of fit)

:low - low rejection threshhold (in sigma of fit)

f - refit the curve using the new parameters

d - delete individual data points

u - un-delete individual data points

s - set the sample range (will ask you to move the cursor and hit **s** again).

t - reset the sample range to all points

q - quit, the fit is as good as I can get it.

Other commands can be found by typing **?**.

If your quartzes have good signal, the trace fit is usually pretty straightforward, showing a curve of order 3 to 7 wandering gently across a range of not more than a pixel or two. If your trace is very erratic, wanders across many pixels, or seems to make an abrupt jump to another position, check and see if the signal in that quartz spectrum is low for some reason, or if the trace has somehow jumped into a neighboring aperture. In the first case, check the parameter **nsum**; increasing it may improve the curve. In the second case, tweaking the **:width** parameter in the apedit mode can sometimes help. If you really can't get it to work,

Figure 5: Quartz Trace Fitting

a straight line passing through the center of the aperture is probably still not too horrible an approximation (though you should check and see if you can improve on it by looking at that particular aperture on other quartz frames).

After the initial tracing of the apertures you will be asked if you want to normalize the apertures - answer "yes". Then you will be asked if you want to edit or mark different apertures sizes for the normalization apertures - usually this is not the case. Type "no" and proceed to the fit the normalization.

After a short wait, you will find yourself again in ICFIT mode (Figure 6). This time you will want to fit the general shape of the quartz spectrum. The order to which you should fit your quartz spectra depends on how your quartzes were taken, and what you want your final data to look like. Fitting a first order curve to your quartzes will allow you to remove most of the instrumental effects of the telescope and spectrograph, leaving the quartz spectrum itself as a residual in your data. If you have used an additional filter in taking your quartzes, however, or if for some reason you object to introducing this residual shape into your data, you might want to fit a higher order curve, say 5th or 6th order. Be sure not to fit too high an order curve, since you want to be sure not to fit out any of the fringes which often occur at the red end of the spectrum. Since multislit data are not usually flux calibrated, the choice of what sort of low-frequency shape you want your spectra to have is yours.

For a closer look at the data and fit, you can use the capital letter cursor commands to zoom and window the plots. As this is an even more straightforward fit, after fitting the first couple interactively and making sure everything is in order, you can answer "NO" to whether

Figure 6: Fitting the Quartz Spectrum

you want to do an interactive fit, and it will do the rest of the apertures on its own. Note that this is not recommended with the trace fits, as they are more likely to have problems.

After the normalization fits are done, be sure to save the aperture data to the database (it will query you about this.) When you emerge from apnorm, look in the magically created directory "database" (type **dir database** and you will find two files containing the aperture definitions for your averaged quartz and for the last frame which you "apedited" - in this case the files will be the same). If you want to go back and redo any of the "ap" tasks, this database keeps you from having to redefine everything all over again. It also allows you to use the aperture definitions of one frame to be used as a reference for another frame.

Now take a look at the newly-created flat with **implot flata**. You should be able to see both the illumination slit function and the pixel-to-pixel variations. Note that the flat is set to unity in between the apertures. You may also want to set areas with low quartz signal to unity, so that you don't add noise to your data by dividing by it. The parameter **threshhold** in apnorm sets the minimum level of the normalization fit, below which the output flat is set to unity.

To create a flat for the "b" exposure, type

```
cl> apnorm qavb flatb ref=qava
```

Do not recenter immediately. This will set you up with the apertures for **qavb** exactly the same as **qava**, which will save you some typing. You may, however, have to recenter anyway and check the widths of the apertures, and you will definitely have to retrace them. Proceed as before to create a second flat.

Finally, normalize your data frames:

```
cl> imdiv multi019 flata obja

cl> imdiv multi022 flatb objb
```

It is important now that you take a good look at your object frames to get acquainted with them. Display the two-dimensional images if you can, and use **implot**. Note which apertures have strong object spectra within the slit, and where in the slit it lies (i.e. is the spectrum in aperture 6, say, off to the the left of the slit? That might be useful to remember later.) Check again the typical width of the entire slit, and this time also note the typical width of the object within the slits. Look at the effects of defocussing— you will probably lose some spectral coverage because of it, so get an idea how bad it is in different parts of the chip. How flat is the background level on either side of your spectrum? Are you satisfied with the normalization?

Because of flexure, illumination problems, and the general problem of fitting sharp edges, apnorm sometimes creates very noticeable spikes on the edges of the apertures (Figure 7). Using sky or dome flats to determine the slit function may help alleviate the part of the problem due to non-uniform illumination, but still may not get rid of it completely. If you get these spikes, you will have to decide whether you can live with them or not. If your slits are very short and you are strapped for background (as are all of us who are greedy to observe as many objects as possible with one mask!), it may make fitting an accurate background level difficult. Remember, however, that these spikes are not "covering" good background area, but rather lie in areas where the slit is falling away sharply, so you're not losing anything useful.

If you decide that you would rather not have any illumination correction at all (that is the culprit in forming those spikes), there is an alternative procedure which will produce a flat which consists of only the pixel-to-pixel variations. Load the package **imred.generic** and **epar flat1d**. This task fits a curve to each individual line or column of the frame and divides by the fit. Because it does not divide by a "single" fit, as apnorm does, the fit conforms to the shape of the spectrum even on the steep sides of the apertures. This means you don't get spikes, but you also don't normalize the slit function.

Edit the parameters of flat1d so that it will fit a 1-6 order function (similar to the normalization fit of apnorm). If you set dispax=2, then set axis=2 as well. **Minflat** plays the same role as **threshhold** does in **apnorm**.

```
cl> flat1d qava flata inter=no &
```

If you would rather look at a few of the fits interactively, keep **inter=yes** and don't put it in the background. The task with prompt you for a line number- choose something in the

Figure 7: Normalized Object Apertures

middle of the frame. You are now in ICFIT, and can proceed as usual. After fitting that line, type **q** and you will be prompted for another line which you might like to fit. Please don't wear yourself out by trying to look at every line, but see if you can find examples on the edges of the slits to illustrate why **apnorm** causes problems by dividing the entire aperture by a single quartz spectrum. When you are done, type **q** instead of a line number, and the task will take the last fitting parameters you set and will fit every line in the image. When it is done, divide your data by this new flat and see how different it is.

6.2 Scattered Light

Another process which you might need to consider is the subtraction of scattered light outside of the slits. This is not the "overflow" from each individual slit (there is currently no way to deal with that properly) but rather a large-scale pattern of additive light which you may wish to subtract. If the signal between and outside of your slits is not flat, but rather shows broad variations, you should use the task **apscatter** to remove it. This is rarely (if ever) a problem with cryocam data, and most users will not need to worry about it. If you do have significant scattered light, however, you should be sure to remove it *before* you do the normalization. If you have already flattened your data with **apnorm**, this means going back to your unflattened data, subtracting the scattered light from all of your frames, and recalculating the flat fields. Consider your first pass through **apnorm** as an educational foray which will make subsequent reductions all the more simple.

Apscatter is organized very similarly to **apnorm**. It allows you to define and trace the apertures (if you have not already done so), and then fit the scattered light and subtract it. Because the scattered light is a two-dimensional function, however, it does this fit in two stages- first between the slits, and then along the dispersion direction.

If, for example, one of your object frames seems to warrant this correction, start with:

cl> apscatter multi019 ref=qava

This will put you in apedit mode with the apertures defined as they were in your quartz frame. Do recenter the apertures, but don't trace them. You do want to subtract the scattered light, as well as smooth and fit it interactively. The next plot you will face shows the slit profiles, and the task's first attempt at fitting a smooth curve to the scattered light. This is ICFIT mode, and you can proceed as before. When you have set the fitting parameters, type **q**. You will be prompted to quit for good, to examine another line, or to change the buffer. The buffer is the number of pixels from the edge of an aperture that the task will ignore in the fit. Try setting it to a few pixels and experiment. Also check different lines to make sure that defocussing is not causing you problems. Remember that this entire process is only to set the fitting parameters and buffer size.

When you are done with fitting the scattered light in that direction, you are asked to fit it in the dispersion direction as well. Again, this is ICFIT mode, and the normal options apply. When you have set the fitting parameters in that direction, **apscatter** will calculate the smooth scattered light image and subtract it from your data. If you are curious or careful, the scattered light pattern can be output into the image name specified by **scatter** (use **epar** to set it.

7 Extraction

Now that instrumental effects are removed from your data, it is time to work on getting some useful information out of it. In almost all multislit applications, you will want to extract a one-dimensional spectrum from each slit, and in many cases subtract a local background from it. This can all be done within the **msred** package using the "ap" tasks. The basic procedure should sound familiar to you: 1) define the aperture centers, widths and curvatures, 2) set the background regions, 3) sum the data within the aperture, subtracting the background if necessary. You have already had a taste of these tasks with **apnorm**, so the rest should come easily.

There are several different ways to use these tasks as they are all interconnected. I prefer to enter the "apextract" tasks through the task **apedit** and access the other tasks while within it. First, though, we should take a look at the parameters of each of the tasks.

Start with **epar apedit**. These parameters deal with finding and centering the apertures correctly. You may want to change the **width** parameter to reflect the width (FWHM) of the *object* within the slit, rather than the width of the entire slit. These parameters can also be changed while running the task.

The parameters of interest in **apdefault** deal with the background subtraction default parameters. Here it is a good idea to set the default background limits to reasonable values- a slit 21 pixels wide with an object in the center seven pixels across will have a background set at [-10:-4,4:10], for example. The background fitting parameters deserve some thought. If you anticipate having many pixels' worth of background to fit in each slit, you can breathe easily, but for many slitlet observations, the background subtraction is the limiting factor in the quality of the data. If you have the "spikes" referred to above in the discussion of **apnorm** or if you think you will have to push your background close to the edge of your slit, it might help to set the rejection parameters to lower values (try 1.5-2 to start) and **nit=1**. The narrower rejection ranges will help keep your background determination from being affected if the background region happens to slip off the edge of the aperture, onto the object, or on one of the "spikes". Again, unless you have many pixels to fit in your background, you will want **order=1** and **nav=1** as well. If you want to experiment while you are looking at the data, these can also be set interactively.

The parameters of **apsum** show you what the task will ultimately do to your data. They can also be set interactively later, so take a look now to see what your options are, and we'll come back to them.

7.1 Defining the Apertures

Actually, you have already done most of this when you defined the apertures for your quartzes, but there are a few little changes that must be made before we can extract the spectra.

We begin with the first exposure but you should always start with the best of your exposures to get a feel for the data. If you can have the frame simultaneously displayed on your terminal (i.e. on your Sun workstation), do so.

```
cl> apedit obja ref=qava
```

We use the apertures we've already defined for qava as a starting point. Don't bother to recenter (it's always best to be distrustful of such things). This task puts you in the now-familiar apedit mode (Figure 8). Check over the aperture definitions, recentering them this time on the objects in the slits. If you have difficulty finding the objects, set **:num** to a higher number, summing more dispersion lines. Also move **:line** around the frame, looking at the defocussing and making sure that the aperture center stays on the object. If necessary, you can consult your displayed image of the frame to see where the objects are in the slits.

The widths of the apertures should be set (using **y**, **l** or **u** as before) according to whether you need to get as much signal as you can from the object, regardless of adding extra noise from summing data in the wings, or whether you want only to sum over the higher signal-to-noise peak.

In most cases, it is a good idea to keep the curvature traces the same as for the quartzes. The quartzes should have much better signal to noise than objects in the slit, and the traces should be identical. The traces are defined relative to the position of the object, so they will work correctly even if you have moved the aperture position. If for some reason you really do

Figure 8: Apertures for the Objects

wish to trace the objects themselves, be sure that the parameter **:width** is set appropriately. If you have had to set **:nsum** to a higher number in order to get good signal on the objects within the slits, you will have to change a couple of the **aptrace** parameters. Type **:aptrace** to edit the parameter list, and change both **nsum** and **step** to whatever you chose **nsum** in **apedit** to be. These numbers control the number of lines summed to make a datapoint in the trace fit, and the spacing of these points. In general, these values should be the same as the number of lines you have to sum to get a good object profile. Typing **CNTRL Z** will get you back into the apedit mode; then type **t**. This will trace and fit the aperture in the same way that **apnorm** does.

7.2 Background Subtraction for Multislits

Now that you have defined the apertures' positions, widths and curvatures, you will also need to define the background level that you will eventually wish to subtract. While still in apedit mode, move the cursor to the first aperture and type **b**. This puts you into an ICFIT mode to fit the background (Figure 9). Using the **w** and **e** keys will let you set the lower left and upper right edges of a viewing window. Undoubtedly, the sample areas will be wrong, so type **t** to clear them, and reset them using **s**. Be sure that while within the apedit mode you have set **nsum** to a number large enough to get good signal on the object profile, so you can see what you're doing. It is also a good, though depressing, idea to look at other different lines across the frame. Again, this will help you gauge the effects of defocussing. Typing **f** will refit

Figure 9: Background Subtraction Fit

the background level. Play with it until it's as good as you can get. Additional commands are available by typing **?**. Exit back into apedit mode by typing **q**. Repeat this for each aperture. Remember that at this point you are only defining the aperture and background parameters, so you can go back and re-do them as many times as you need.

7.3 Extracting the Spectra

The **apsum** parameters are set by typing **:apsum** in the apedit mode. Here you can choose whether you want to subtract an adjacent background or not, and whether you want the background fit to the ICFIT parameters you have just set, or just have the area marked as background averaged. Figure 10 shows the parameter list set up so that a fitted background is subtracted, and the data are summed using profile weighting.

There are two options for weighting the data you are summing, profile weighting and variance weighting. It is advisable to avoid variance weighting and stick with profile weighting for now.

After you have set up the apsum parameters, exit back to the apedit mode by typing **CNTRL Z**. To extract all of the spectra at one time, type **a e**. You will be asked for an output file name, and possibly a sky file name (we will be using **neta, netb** and **skya, skyb** in this example) . These files are two-dimensional IRAF images in multislit format, and will have a **.ms** appended to their names. These files have one line per aperture spectrum, i.e. neta.ms has ten lines, one for each aperture, and **implot**ting line 1 will plot the extracted

Figure 10: Parameter List for Apsum

spectrum from aperture 1. One problem with this part of the task is that if your output image already exists, it will not ask you for another output name, but will rather throw your newly extracted spectra into the ether and you will have to extract it all over again.

There is a parameter called **skyextract** used when subtracting a background, that outputs this background spectrum into a separate output file so that you can go back and reexamine it. This option is worth playing with. Once output, the background can be added back into the data, and then a smoothed background subtracted. Smoothing the background and resubtracting it does seem to decrease the noise in the output spectrum, but also broadens strong emission lines in the background spectrum, creating large residuals in the wings of such lines. Such an experiment might be done in the following way:

1) Extract your data setting **skyextract = yes**, outputting the sky spectra into **skya.ms**.

2) Add the sky back into the extracted spectra:

```
cl> imarith neta.ms + skya.ms noskysub.ms
```

3) Smooth the sky spectra. **Boxcar**, in the package **images** executes a running average box smooth on a two-dimensional image. In order to smooth the multispec format image, you must remember to set the y-dimension box size equal to unity. Otherwise the spectra will be smoothed into one another. The x-dimension box size can be set approximately equal to a resolution element of your spectra.

```
cl> box skya.ms xbox=4 ybox=1 skysmooth.ms
```

4) Subtract new, smoothed sky:

```
cl> imarith noskysub.ms - skysmooth.ms newneta.ms
```

Again, as the quality of the background subtraction is often the limiting factor in the quality of the reduced data (especially in wavelength regions where the night sky lines are strong and closely-spaced), experimenting a little with different methods is probably worthwhile.

Repeat this entire process for each of your object frames, using the appropriate quartz frame as a reference. Different exposures through the same masks may have slightly different centers due to flexure, or if you "tweaked" the telescope position between exposures. Seeing or focus changes may change the width of the object features and your background definitions. The traces will also change from exposure to exposure but, remember, you already caught that when you traced the appropriate quartz.

8 Wavelength Calibration

The wavelength calibration of multi-object spectra at this point is, besides some changes in bookkeeping, identical with that of normal one-dimensional spectra. In this guide, it is assumed that you have taken one or several comparison spectra which can be used as is to calibrate your data.

8.1 Creating the Calibration Spectra

It is very important that the comparison spectra used to calibrate each object spectra be extracted in exactly the same way as the object spectra. In our example, comparison spectra were taken before and after each exposure. We want to use two comparisons to determine the wavelength calibration for each each of two exposures, so we need to create four sets of comparison spectra.

This can be done quite easily using **apedit** the same way we used it before. First, though, **epar apsum** and change **background** to none, as comparisons, of course, have no background. Then:

```
cl> apedit multi018 ref=obja output=compa1
```

Do NOT recenter, do not change a thing! Instead, type **a e** and extract the comparison spectra using identical apertures and traces as the object spectrum. In this example, the output .ms frame will be called **compa1**. The **a** tells us that it is extracted like exposure **a**, and the **1** denotes that it is the first of the two comparisons. If you wish to look at each comparison spectrum, OK, but you might want to answer "NO" when it asks you if you'd like to review each of the comparison spectra. Don't bother to save the apertures to the database– they are the same as for **obja**.

Next, do the same for the other comparison you want to use for calibrating **obja**.

```
cl> apedit multi021 ref=obja output=compa2
```

Next, do the same for the comparisons you wish to use for the second exposure:

```
cl> apedit multi021 ref=objb output=compb1

cl> apedit multi024 ref=objb output=compb2
```

Note that **compa2** and **compb1** are derived from the same frame, but with different aperture definitions and traces. If you have a lot of frames, you can do these in the background.

```
cl> apsum multi024 ref=objb out=compb2 rec- tra- back=none interac- &
```

It is usually a good idea, however, to spot check a few of the spectra to make sure that nothing has gone awry. When you have created all of the comparison **.ms** images that you need, pat yourself on the back. You have finished with the apextract tasks.

8.2 Determining the Wavelength Calibrations

First, display a two-dimensional comparison frame (e.g. **multi018**) and identify which aperture is which (don't forget that you have ignored any set-up holes). Then, by looking at the emission lines, write down the aperture numbers in order, starting with the one that is shifted the furthest towards the bottom of the frame. This is the order in which you will want to determine the wavelength calibrations; you want the minimum shift in wavelength between one aperture and the next.

The task **identify** is a complex task which allows you to identify emission lines in your comparison spectra, and then fit a smooth function of wavelength versus pixel number. This fit is the dispersion correction, also called wavelength calibration. The fit parameters are then stored in a database and subsequent tasks use the fit to correct your spectra to the proper wavelengths. If you have used **identify** before, rest assured (or be warned) that the version you will use now is identical to the task in the **onedspec** package, etc. If you have not used it before, it is probably worthwhile to get a hardcopy of the help pages for the task (**help identify | lprint**) and keep them on hand.

Start by using the editor to create an "@" file called **identin** which contains a list of the image lines in your first comparison **.ms** file. Write down the lines in order of their relative shift (i.e. the order you wrote down after examining the two-dimensional comparison frame). For example, it might look like this:

```
compa1.ms[*,8]
compa1.ms[*,9]
compa1.ms[*,10]
compa1.ms[*,6]
compa1.ms[*,4]
compa1.ms[*,2]
compa1.ms[*,1]
compa1.ms[*,3]
compa1.ms[*,7]
compa1.ms[*,5]
```

Then type:

```
cl> identify @ident
```

You will be shown a plot of the first comparison spectrum in your list (in this case, the 8th aperture). You now need to identify a few of the brightest lines, (at least three, though four or five well-spaced lines would be better). Mark them by putting the cursor close to their

Figure 11: He-Ne-Ar Calibration Spectrum

peaks, and type **m**. You will be prompted to enter the wavelengths. If you make a mistake, **d** will delete a line, and **u** will let you change the assigned wavelength. When you have a few lines marked, type **l**. Identify will now fit a straight line to your data (not a bad first approximation), and, using a table of comparison lamp values, attempt to identify as many other lines if your spectrum as it can. The default wavelength tables are for He-Ne-Ar lamps. If you are using another type of arc lamp, you will have to change the identify parameter **coordlist** to something more appropriate. Check through the standard options by **page onedstds$README** and see if any of the lists are appropriate for your data. Then you can change the parameter **coordlist** to match. If you don't think any of the standard options will work for your instrument, it is best to ask for the help of someone who knows what other wavelength tables may be available.

Hopefully, while you've read this, identify has been able to mark a bunch of new lines for you (Figure 11). Now you need to check and make sure that it did it all right, and see if the dispersion correction fit is good. Typing **f** will get you into ICFIT and display the residual (data - fit) versus pixel number (Figure 12). If the residuals look like they follow some higher order function, you can increase the order with **:or**. It is usually not necessary or advisable to increase the order to more than about 4. Typing **f** again will re-fit the data with the new parameters. Notice that the standard deviation of the residuals is in the graphics header.

At this point, you need to fiddle around with the data until you get a wavelength solution which you like. A few useful keystrokes are:

Figure 12: Fitting the Wavelength Solution

d - delete a point

u - undelete a point

c - move cursor to the nearest line (for identifying problem lines)

l - display the non-linear part of the fit (Figure 13)

j - display the fit residuals

q - exit from ICFIT mode

? - more information on ICFIT commands

Note that if you delete a line, it is removed from the line list for all subsequent fits (that means the rest of your apertures, as well). If you want to get it back, you must start over again with **l**.

It is also very useful to examine the individual lines, in order to weed out blends, low signal peaks, and misidentifications. Get out of ICFIT mode (type **q**) and you'll be shown the comparison spectrum again. I usually like to step through each of the lines in "zoom" mode to make sure that there are no problems. Some useful commands in this mode are:

+ - move cursor to next line

Figure 13: Non-linear Component of the Fit

- - move cursor to previous line

d - delete a line

m - mark a line

u - change the wavelength assignment of a line

z - zoom

p - un-zoom (pan)

Be sure that you have enough lines, and that the lines are spaced throughout the entire spectrum. Fitting a high-order function and not having enough lines at the endpoints to "tie it down" will cause you problems. Also keep an eye out for a "solution not monotonic" message which may flash across the bottom of your graphics window. These are conditions you should be aware of if you're being careful anyway, but the error messages are fleeting. When you're satisfied with the wavelength solution, type **q** and write it all to the database.

Now, because you started **identify** with an "@" file, you will immediately be shown the next spectrum in your list— surmounted by the fit you have just determined. Now the fits to individual apertures should be very similar except for a zero-point shift due to the shift in the slit positions on the mask. In many cases, you can save a lot of time by simply trying this. Find a strong, isolated line in the center of your spectrum, put the cursor on it and type **s**. You will be prompted for its wavelength, and then the task will shift the solution so it lines up with the new data. Proceed as you did before, typing **f** to calculate a new fit. Do be cautious- sometimes rather subtle problems can creep in at this point. The non-linear portion of your fit should look very similar to that of your first aperture and the RMS of the residuals should also be similar.

Shifting the solution like this works most of the time when the shift is not large, but it can sometimes get confused and misidentify lines and really cause a mess. When that happens, get out of ICFIT mode, and type **i** to reinitialize and delete all of the lines, and start from scratch. This can be a long procedure if you have many apertures on one frame. Hang in there- you only have to do it once for each aperture.

Once you have wavelength solutions for one exposure through each of the apertures, it is quite simple to use these solutions to calibrate the rest of the comparison spectra. The task **msreidentify** uses the database information for each of the apertures to fit the wavelength solution to each subsequent exposure. This only works if you have extracted each comparison such that the apertures are in the same order in the **.ms** frame, but you should be doing that anyway.

```
cl> msreidentify compa1.ms compa2.ms

cl> msreidentify compa2.ms compb1.ms

cl> msreidentify compb1.ms compb2.ms
```

You should now have solutions for all of your comparison spectra entered in your database.

At this point, again, caution is advised. **msreidentify** is pretty good at refitting the wavelength solutions properly, but it can occasionally make big mistakes. It is advisable to go back into **identify** and at least glance at each of the solutions to make sure that there are no gross errors (when errors occur in **msreidentify** they are usually very obvious).

8.3 Calibrating the Spectra

Before you can apply these hard-won wavelength solutions to your data, you need to specify which comparisons you want to use and in what combination. The task **refspectra** is a complicated task which does a very simple thing- it goes into the header of your object frame and creates two new header keywords: REFSPEC1 and REFSPEC2. These keywords contain the names of the comparison spectra you want to use, and their relative weights. If you only have one comparison spectrum, it only creates REFSPEC1.

Choosing the relative weights can be a complicated business, and for that I refer you to the help pages for refspectra. The most likely choices will be either to average two comparisons, or to do a linear interpolation with respect to some parameter, say UT. In our current example, averaging is sufficient since the comparison exposures are very short compared to the object exposures, and the comparisons were taken immediately before and after each object exposure. So:

```
cl> refspec neta.ms refer=compa* select=average
```

Refspec confirms that the average of the comparison spectra compa1 and compa2 will be used. (Now do you see why we chose the names of the comparison spectra like that?) Likewise:

```
cl> refspec netb.ms refer=compb* select=average
```

If you would prefer to use the interpolation option, you must be sure that the UT keyword in your image header denotes the *middle* of the exposure, not the beginning (as KPNO headers do!). If you try to interpolate using the beginning of the exposures, the first comparison, since it usually has a much shorter exposure time than the object exposure, will be weighted much too heavily. Again, if you want to use one of the more complicated options, you should carefully read the documentation on **refspectra**. Also see the task **setairmass**. If all of this is too daunting, yet another option is to use the task **imhedit** to manually create the keywords REFSPEC1 and REFSPEC2, bypassing **refspectra** entirely.

You are now finally ready to assign the wavelength calibrations to your object data. The task which does this for multispec data is called **msdispcor**. Look at the parameters for this task using **epar** (Figure 14) and think a moment about the final form you want your data to take. In most cases, you will want all of the apertures to have the same number of angstroms per pixel, but different starting wavelengths. You might also want all of the different exposures of a specific aperture to have the same starting wavelength, to make combining them easier. If you are planning to cross-correlate your data later, you may also want to bin your data logarithmically. There are also options for summing or averaging all of the apertures together, or forcing them all to have exactly the same dispersion correction. This last can make subsequent subtraction and cleaning of sky emission and residuals much easier, and should be kept in mind. You can always rebin the data, but you really want to minimize the number of times you do it.

In our example, we want all of the spectra to have the same angstroms per pixel, but each individual aperture will have its own starting wavelength. We also want the two exposures, to have the same starting wavelength for each of the apertures. To do the first, we set the parameter **dw** equal to the wavelength/pixel you want. If you are not sure what you would like, do a dry run of **msdispcor** by setting the parameter **listonly** to yes and setting the other parameters so that the wavelength dispersion is chosen freely:

```
cl> msdispcor neta.ms test dw=INDEF same- glob- list+
```

This will show you the default wavelength solutions that it has chosen for each of the apertures. In this example, a value of 3.2 angstroms per pixel was chosen. Set **dw** to this number to force all spectra to have this value.

To force **msdispcor** to make each aperture have the same solution (w1 and dw) in each

Figure 14: Msdispcor Parameters

exposure, set the parameter **global** to yes. If you want all apertures to have the exactly the same wavelength correction, set the parameter **samedisp** to yes as well.

Then:

```
cl> msdispcor neta.ms,netb.ms fina.ms,finb.ms
```

Your data are now dispersion-corrected. You are almost done!

9 Combining, Cleaning and Examining Data

This section suggests a few of the options available to you in IRAF for combining your multislit data in different ways, cleaning them and for plotting and displaying your data. Since you have made it this far, you are now an IRAF data-reductions expert, and should be able to figure out these tasks without a lot more fuss. As always, it is a good idea to **epar** a task before running it, and the help pages are all online.

Combining data from several exposures is very simple if the spectra are in multispec format, and you have been careful to make sure that the starting wavelength and the wavelength-per-pixel is the same for each individual aperture. You can simply **imsum**, **imarith** or **imcombine** the two-dimensional frames and combine each of the apertures simultaneously. These tasks all have different options and capabilities, so you should have a look at them before deciding which you would like to use.

There are a few useful tasks which might help in cleaning your data of cosmic rays and the residuals from poor background subtraction. The task **lineclean** in the **images** package can be used for both. This task fits each line of the multispec format spectra with a smooth curve, and replaces any deviant data points by the fit value. To remove cosmic rays, try fitting your data with a very high (say 200-250) order function. Most cosmic ray events (depending on the type of CCD used) and bad pixels are single-pixel events, whereas true spectral features are resolved. Choosing the order of the fit to be 1/2 or 1/3 of the number of pixels in the spectrum will help in weeding out these single-pixel events. Play around with different spectra and with setting the high and low rejection parameters until you find a combination that you feel safe with (after all, you don't want to reject any real emission lines). Care should be used with this process– even if an entire line is not rejected, it can occasionally change the shape of sharp-but-still-resolved emission features. If your data have been rebinned or smoothed, cosmic rays may be broadened too much to be rejected by this method. To avoid this, cleaning the data before the wavelength calibration is performed is suggested. It is recommended that you always do this process interactively, as well.

If your background subtraction has left you with unsightly night sky emission line residuals, **lineclean** can be used to remove some of them as well. This is much simpler if your data are binned so that all of the apertures have the same starting wavelength. Run **lineclean** on your multispec format image, and take a look at the line which has the worst-looking residuals. Set the fit sample (type **t** to reinitialize it, and then **s**) so that it includes only pixels in and surrounding these bad areas. You will have to experiment to see how wide these areas should be, but they usually do not have to be too much wider than the actual residual. Fit these areas to a low-order curve; the worst of the night sky residuals will be discarded. Note that **lineclean** does not read the wavelength solution parameters from the image header, so you might want to note at which pixels you expect the prominent night sky lines to cause you problems. Once you have set the sample areas the way you want them, the task will use the same ones for each aperture– that is why having the same wavelength correction for each aperture is important here.

Plotting your data is also possible with a plethora of tasks. The **onedspec** task **splot** is a very powerful spectrum display and edit task, which allows you to do spectrum arithmetic, smoothing, pixel editing, measure line positions and equivalent widths, and lots more. Again, you can use this task on multi-spec format frames— it will prompt you for a line number. There is also a plotting task for multispec format frames called **specplot**. This task plots all of the spectra on a single page, allowing you to easily compare them. If you wish to convert multispec format files into individual **onedspec** images, you can use the task **msselect**. To create a one-dimensional spectrum of the fifth aperture of the multispec format image **fina.ms**:

```
cl> msselect fina ap5 5
```

A Outline of Multislit Reductions

I. Getting Started

1. Read in data from tape using **rfits**, **rcamera**.

II. Bias Subtraction

1. Combine bias frames with **zerocombine**
2. Subtract biases and trim images using **ccdproc**
3. Inspect frames for scattered light. If necessary, subtract using **apscatter**.

III. Normalization

1. Set dispersion axis using **setdisp**
2. Organize data into separate directories using **imrename, imcopy**
3. Combine quartzes using **flatcombine**
4. Create normalization flats using **apnorm**: define, trace and normalize the quartz spectra.
5. Flatten data using **imdivide**
6. Inspect data using **implot**
7. Alternative option: **flat1d**

IV. Extraction

1. Check all **apextract** parameters
2. Use quartz apertures as references in **apedit**
3. Fit background regions
4. Extract spectra
5. Alternative: extract sky as separate spectra, smooth and subtract from data

V. Wavelength Calibration

1. Extract comparison spectra, making sure to use the same aperture and trace definitions as your object spectra
2. Order spectra with respect to starting wavelength
3. Calibrate first aperture in list using **identify**
4. Calibrate remaining apertures within identify using the **s** (shift) key and checking each solution
5. Calibrate remaining comparison exposures using **reidentify**
6. Assign comparisons to each data frame using **refspectra**
7. Calculate dispersion solution for each aperture using **msdispcor**

VI. Combining, Cleaning and Examining Data

1. Combine data using **imcombine**
2. Clean data using **lineclean**
3. Plot data using **splot** or **specplot**
4. Extract data to **onedspec** format using **msselect**