A User's Guide to Reducing Slit Spectra with IRAF

Phil Massey Frank Valdes Jeannette Barnes

15 April 1992

Contents

1 Introduction

Spectroscopy is one of the fundamental tools of the astronomer, and the proliferation of linear, digital detectors on spectrographs has greatly increased the astronomer's ability to perform quantitative analysis. However, with this increased ability has come increased demands upon the tools used for reducing these data, both in the sophistication of the algorithms we've come to expect, and in the ease with which these tools can be used. It's easier to reduce CCD spectra than it was photographic plates, but our expectations have also risen: we want to really *use* the known noise characteristics of our detectors to make a mathematically optimal extraction of our spectra so that we can achieve the maximum signal-to-noise possible from our precious observations. Furthermore, while we might have been willing to spend days digitizing spectrograms on a microphotometer and doing the calibration necessary to extract useful spectra from photographic plates, we now want to see our two-dimensional CCD images fully reduced to dispersion-corrected, fluxed spectrophotometry before the telescope has finished moving to the next object.

IRAF provides powerful reduction tools for accomplishing this task. This manual is intended to guide you through the steps of extracting and calibrating spectra. We will start out by considering the simpliest case imaginable: a single star on the slit. In the appendices, we will consider increasingly complex cases: the extension to several stars along the slit (Sec. A) and the generalization to the "long-slit" case (Sec. B). However, this manual does not deal with the multi-object fiber situation; there are excellent routines for handling these data, and the user is referred to Frank Valdes' *Guide to the Multifiber Reduction Task DOFIBERS* or *Guide to the HYDRA Reduction Task DOHYDRA* for example. Nor does this manual touch upon the issue of cross-dispersed data; the user is referred to the Tololo cookbook *Echelle Reductions with IRAF* by Mario Hamuy and Lisa Wells. The present guide is intended for use with IRAF V2.10; if you are something older, we encourage you to obtain a modern version by contacting the IRAF group (email iraf@noao.edu).

Additional resources you may wish to review are:

- *A Beginner's Guide to Using IRAF* by Jeannette Barnes
- *A Quick Look at Sun/IRAF on the Tucson Sun Network* by Jeannette Barnes
- *A User's Guide to CCD Reductions with IRAF* by Phil Massey

We assume that by the time you get to this manual you have removed the "detector signature" from your data—that you have removed any DC-offset, removed the pixel-to-pixel gain variations using flat-fields, etc, e.g., that your data is now linear and zero counts equal zero light. The manual "A User's Guide to CCD Reductions with IRAF" explains how to do this with CCD data, and this manual is intended to pick up where that one left off. However, for convenience a quick summary of these steps is provided in Sec. D.

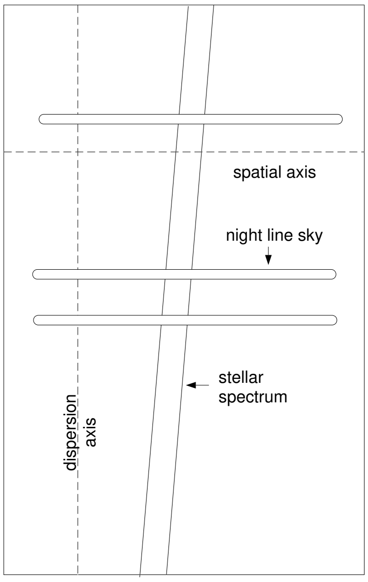

Figure 1: A stylized version of a "perfect" stellar spectrum.

2 Extraction and Calibration Overview

The steps involved in extracting a stellar spectrum in IRAF are conceptionally straightforward. We show a "schematic" spectrum in Fig. 1. The dispersion axis is along columns in this example, and the spatial axis is along rows. (Rows are called "lines" in IRAF.) Extraction of this spectrum could be broken down into the following steps:

- **Find the spectrum.** This may be done manually by examining a cut along the spatial axis and indicating the appropriate peak with a cursor, or it can be done automatically if the appropriate spectrum is the strongest peak present.

- **Define the extraction window and the background windows.** In practice this is done by specifying the size of the extraction window in terms of the number of pixels to the left of the center of the spatial profile, and the number of pixels to the right of the center of the profile. Similarly the background region is defined in terms of a region to the left and right of the center of profile. One can then examine these regions displayed upon a cut along the spatial axis, and alter them if need be.

- **Trace the center of spatial profile as a function of the dispersion axis.** Even though we assume that the spatial axis is exactly along a row or column (as discussed below), the spectrum will *not* be exactly perpendicular to the spatial axis (i.e., the stellar spectrum is not exactly parallel to what we are taking as the theoretical dispersion axis.) Instead, the exact center of the spatial profile will shift slightly with location along the dispersion axis. There are least three reasons for this: (a) the camera optics introduce distortions which will be be worse along the longer (dispersion) axis, (b) gratings do not sit exactly square in their cells, and (c) differential atmospheric refraction will cause the blue end of the spectrum to be shifted along the slit closer to the zenith than that of the red end of the spectrum. The latter effect suggests that we can expect the angle made by the spectrum to the dispersion axis will differ, often significantly, from one exposure to another.

- **Sum the spectrum within the extraction window, subtracting sky.** At each point along the dispersion axis, the data within the extraction aperture (centered spatially based upon the value that the trace is at that point) is summed, and the sky background is subtracted.

There is an important if subtle point in this last step: the spatial axis is assumed to be exactly along a row (or column), and *not* perpendicular to the trace. Thus if you want good sky subtraction, you must either have set up the spectrograph (usually by rotating the dewar) in such a way that a comparison line is well lined up along a line or column, or else the frame should be geometrically transformed after the fact to make this true. (Procedures for this transformation are given in Sec. B, but if you can avoid this step by being careful in your set-up do so, as additional interpolation steps are never going to improve your data.)

At this point you have extracted a one-dimensional spectrum from your two-dimension image, and can do the following calibration steps:

- **Wavelength Calibration.** You doubtlessly would like to see your spectrum with a wavelength scale. This procedure can be broken into the following steps:

  1. Extract a one-dimension spectrum from the appropriate comparison exposure using the identical aperture and trace used for the object spectrum you are planning to calibrate.
  2. Determine the dispersion solution for this comparison spectrum. This can be done interactively the first time, and the solution used as a starting point to determine the dispersion solution for other comparison exposures.
  3. If a second comparison exposure will be used for providing the wavelength calibration for the object spectrum (such as interpolating in time between two comparison exposures, or using the average solution of comparison exposures that were taken immediately before and after the object exposure), repeat steps (1)-(2) for the second exposure.
  4. Using the dispersion solution(s), put the object spectrum on a linear wavelength scale by interpolating to a constant delta wavelength per pixel. (There are a

variety of options available here, including linearizing the spectrum into log wavelength, or not linearizing it at all but instead invoking the new "world-coordinate system" of IRAF.)

- **Flux Calibration vs. Normalization.** For many users, leaving the wavelength-calibrated spectra in terms of integrated number of counts is sufficient. However, one is then usually left with a smooth residual wavelength-dependent signature due to the effects of the grating blaze and change in detector sensitivity over the wavelength region observed. There are two things you might choose to do, depending upon your scientific needs:
  - **Flux Calibration.** If you have observed suitable spectrophotometric standard stars, it is possible to transform your data to either absolute or relative flux units.
  - **Normalization.** It is often useful to normalize the object spectrum so the continuum level is unity. This is easily performed by fitting a smooth function through the continuum and dividing the spectrum by this fit.

3 Doing It One Step At A Time

3.1 Introduction to the Extraction and Calibration Routines

The IRAF routines for doing spectral extraction and calibration are loosely referred to as the **apextract** package, but there are three areas where these routines have been "customized" for extraction of data obtained with slit spectrographs in simple (non-cross-dispersed) mode. These three packages are all within the **noao imred** package, and are called **kpnoslit**, **ctioslit**, and **kpnocoude**. The differences between these are slight (such as the defaults used for the comparison line table), and the algorithms, routine names, and parameters are identical—it is only a few of the default values which differ, and which one you use does not really matter—pick one that seems appropriate and be happy.

As each step of the extraction process is completed, an entry is made in a local subdirectory **database** in a file called **ap***imagename*. It is through these database entries that one part of the extraction process communicates with the next. After the extraction process for a particular spectrum is complete, this database entry will contain the specifics as to the size and location of the extraction aperture, the size and location of the background windows, and the function and coefficients of the trace fit. This allows the astronomer to extract a comparison spectrum, for instance, using the identical parameters as were used in extracting the object spectrum.

The routine used for identifying the comparison lines and performing the dispersion solution make a separate entry in the **database** directory in a file called **id***compimagename*.

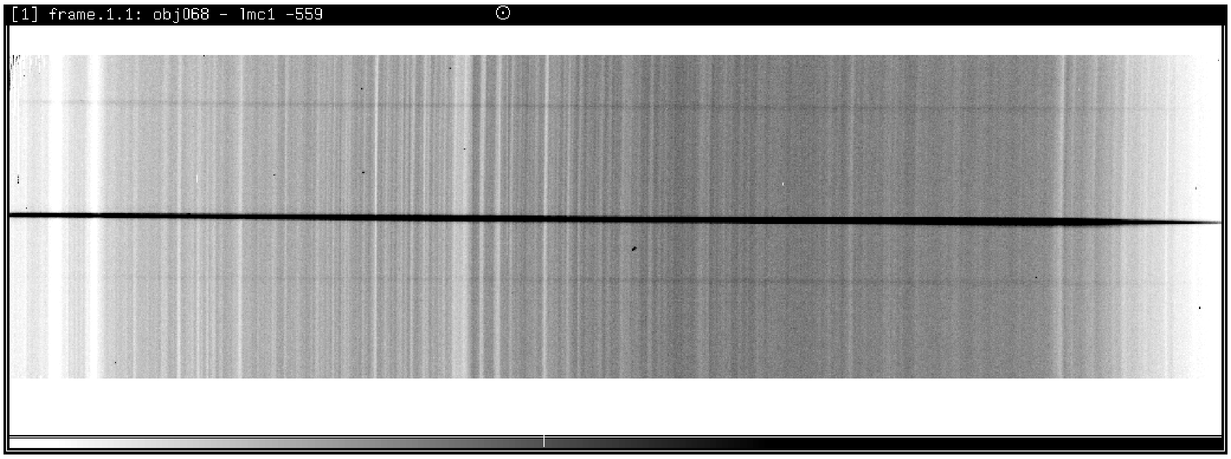

Figure 2: A stellar spectrum of a well-exposed star in the LMC. The wavelength coverage is $\lambda\lambda 3800 - 5000$, nearly that of the classical MK classification region. Severe vignetting occurs on the red (right) side of the spectrum. Note the strong solar absorption spectrum of the night sky due to bright moon-light: CaII H+K are on the left.

This makes it easy to juggle what dispersion solution should go with what spectrum; it also makes it easy to create a dispersion solution for a similar comparison spectrum using the previous one as a starting point.

The routines for doing flux calibration or normalization to the continuum are found in each of these same three packages.

3.2 Characterizing Your Data

For the purposes of illustrating the procedure for extracting and calibrating a single stellar spectrum, we will be using spectra obtained with the CTIO 4m and the newly implemented 1200×400 Reticon CCD. The dispersion axis goes along rows, and the spatial axis along columns. Fig. 2 shows a sample exposure. This data is rotated 90° to the schematic example shown back in Fig. 1, and the background sky spectrum is dominated by a solar absorption spectrum rather than strong night-sky emission lines we sketched in Fig. 1.

The first step in reducing your data is to characterize it. What is a reasonable width for the extraction window? Where do you want the sky windows to be located?

Using

implot *imagename*

make a cut perpendicular to the dispersion axis. (A **:c 500** will plot column 500; a **:l 500** will plot line 500.) The plot will resemble that of Fig. 3. Using the cursor (capital **C**) determine the width near the base of the spatial profile. The data shown here extends from columns 154 through 164; i.e., about 11 pixels spatially. (The full width at half maximum

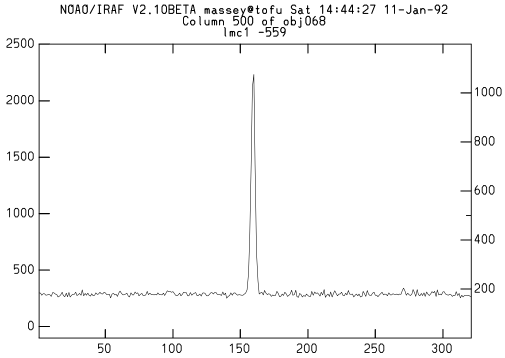

Figure 3: A cut perpendicular to the dispersion axis shows the location and width of the spatial profile. By using the cursor in **implot** we find that the base of the spatial profile extends from columns 154 through 164.

[fwhm] is of course much smaller than this; we estimage this to be about 3-3.2 pixels.)

Next let us examine a comparison spectrum (See Fig. 4). We want to answer a couple of questions from this frame. How good is the resolution? How well did we line up the slit along a column? To answer the first of these questions we simply have to plot a line along the spectrum (Fig. 5), and use the cursor to measure the fwhm and base-to-base width of a typical (strong) comparison line. For this data we find that a typical comparison line has a fwhm of 2.1 pixels, and a base-to-base width of 7.5 pixels.

We can check the alignment by using **implot** to overplot a row near the bottom, a row near the middle, and a row near the top (Fig. 5). From these plots we find that the overall alignment from one end to the other is rather poor, about 2.0 pixels over 300 rows. However, since we will keep our sky regions restricted to a full range of less than 50 pixels, this means that the maximum misalignment will actually be one-third of a pixel, not enough to get excited about. Had the alignment been worse than this, or if we had needed to use a larger region of the chip, we could have improved our sky subtraction by first untilting these lines by using the procedure described in Sec.B.

3.3 Extracting the Spectrum

Although the individual steps for extracting a spectrum outlined in Sec. 2 can be done by separate routines, for convenience the routines and all the associated parameters have been consolidated in a single task called **apall**. You can edit the parameters of **apall** and then use **apall** for running through all the steps involved in extracting the spectrum. Alternatively, you *could* change the values of the parameters within individual tasks, run the individual tasks, and still accomplish the identical extraction.

Before proceeding to actually extract a spectrum using **apall**, we will discuss the prin-

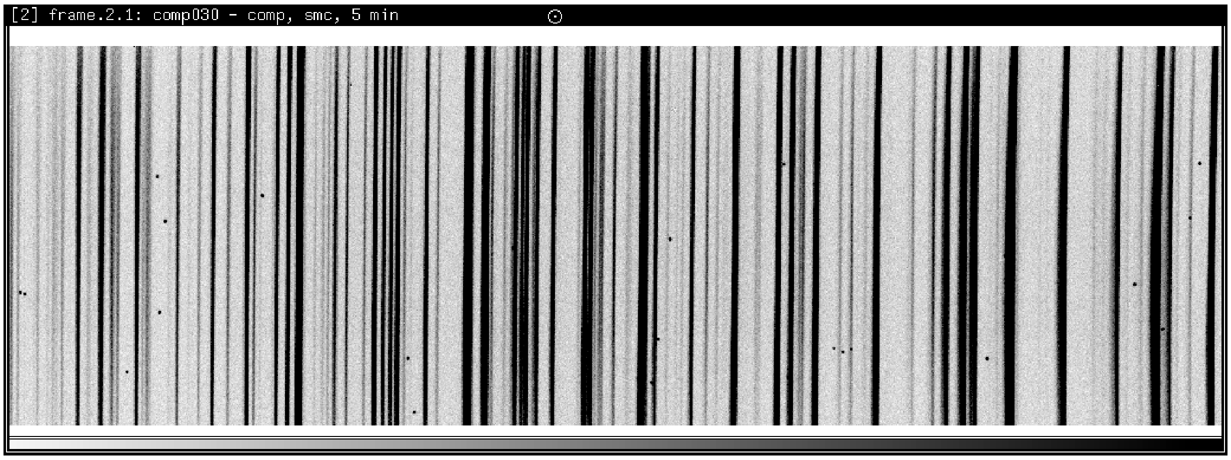

Figure 4: A comparison spectrum is shown for the same Reticon data.

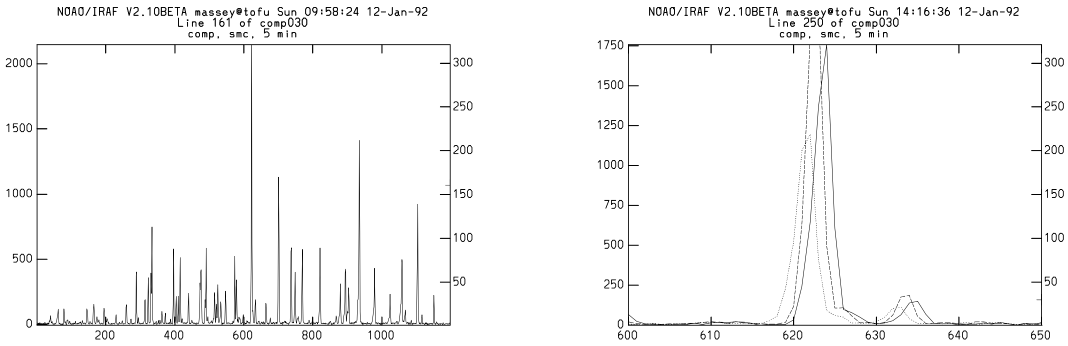

Figure 5: We can measure a typical comparison line to find that it extends 7.5 pixels and has a fwhm of 2.1 pixels. The figure on the right shows the misalignment of the slit, evident in this overplot of three rows of the comparison spectrum.

ciple extraction parameters below. Be aware that there are plenty of other parameters controlling the extraction; the user is referred to the detailed "help" pages for a particular task. Most of these parameters can be adjusted interactively during the extraction process, and we will use the discussion below to start out with reasonable values, and then fine-tune them during the first attempt at extraction.

3.3.1 Parameters for Extraction in *apall*

The following paragraphs spell out the meaning of the major parameters that you *might* want to change from the defaults for your particular needs.

Parameters Controlling the Extraction Aperture: The first stage of extracting the stellar spectrum will be in finding the exact spatial center of the stellar profile at some point along the dispersion axis, and in setting the lower and upper limits of the extraction aperture. The following are the fundamental parameters defining these operations:

- **line** The dispersion line used for finding the center of the spatial profile. Setting this equal to "INDEF" results in using the middle of the dispersion axis.
- **nsum** The number of dispersion lines summed (around **line**) used in finding a good center for the spatial profile. Setting this to a number greater than 1 (10, say) simply improves the signal-to-noise of the cut used for centering.
- **width** The profile width is the base-to-base size of the stellar profile; this value is used in the centering algorithm.
- **lower** The lower aperture limit (relative to aperture center).
- **upper** The upper aperture limit (relative to aperture center).
- **resize** Do you wish to use an aperture set by **lower** and **upper**, or do you wish to use an extraction aperture whose size is set by when the spatial profile sinks to some fractional value of the peak value? If the latter, turn **resize** to "yes".
- **ylevel** If **resize=yes**, then the value of **ylevel** is used to define the fractional level of the peak used for resizing the aperture.

For the data presented here, we would set **width** to 10 pixels, and we might choose to set **lower** and **upper** to -5 and 5 respectively—that would give us an extraction window that is 11 pixels wide. Alternatively, we could set **resize** to yes. In this case, the default behavior would be that the extraction aperture would be as wide as the spatial profile at 10% of its peak. (If you don't like 10%, reset the value **ylevel**.) Thus reasonable starting values for these particular data discussed here are:

- **line=INDEF**

- **nsum=10**
- **width=10**
- **lower=-5**
- **upper=5**
- **resize=no**

Parameters Controlling the Background Windows: Although conceptionally simple, the background regions are somewhat subtle in their specification. Fortunately, it is easy to change the background region interactively; we will demonstrate this in detail when we deal with crowded backgrounds in Sec. A. Below are the parameters we must deal with:

- **b_sample**. This specifies the location of the background regions relative to the center of the spatial profile. For the data discussed above we might choose "$-20:-8, 8:20$", which would then specify a background region to either side of the star, beginning 8 pixels from spatial center and extending to 20 pixels from the center.
- **b_naver**. The specifies how many adjacent points within each background window will be used in determining a value to go into the background fit. A negative number specifies that a median will be taken, which helps weeds out cosmic rays in the background region. If you want to simply use two good values, one from either background window, you should specify a very large, negative number, such as -100.[1]
- **b_funct** Is the type of function (chebyshev, spline3) used to fit to the values determined from taking the average or median of the points in the background sample.
- **b_order** Is the order of the function used to fit the background points.

To summarize, the background value used (*if* we also turn background subtraction on) will be determined by taking the average or median within **b_naver** adjacent points within the sample regions specified by **b_sample**, and either averaging these values or fitting a function of type **b_funct** and order **b_order**.

[1] In this example, each background region contains 13 points (20-8+1), so you might expect that -13 would work as well as -100, but it won't: owing to how the routines treats fractional pixels, you would often get two equally-weighted data points in each background region, one of which would consist of a single point—which could easily introduce a big cosmic-ray spike into your sky. However, -14 would work perfectly well.

In the example here we want to do about the simpliest, good thing we can: make a clean estimate of the background on the left side, make a clean estimate of the background on the right side, and take the average of these two values. Given that our spatial profile extends to a radius of 5 pixels, we thus choose the background sample region to include "lots of points" that start a few pixels outside that range. We specify a large negative value for **b_naver** so we get a single value (the median) for the data in each window. We specify a function type **b_funct** of chebyshev, and an order of 1 so we fit the best "constant" to these two values. If the windows were asymmetrically chosen with respect to the spatial profile, we might make this a straight line (**b_order=2**). Thus:

- **b_sample=**$-20 : -8, 8 : 20$
- **b_naver=**-100
- **b_funct=chebyshev**
- **b_order=1** or **2**

Parameters Controlling the Trace: For well-exposed spectra, the trace is an easy thing to get right. Parameters that control the trace are as follows:

- **t_nsum** Number of dispersion lines summed before looking for the peak of the spatial profile.
- **t_step** Step size along the dispersion axis that will be used in determining the trace.
- **t_funct** Fitting function.
- **t_order** The order of the fit.
- **t_niter** The number of rejection iterations for automatically rejecting points that deviate by a few sigma of the mean, where the sigma is determined from the entire trace. Setting this to 1 or 2 will remove the most discrepant points.

The tracing routines starts with the center of the spatial profile found at some point along the dispersion axis when the aperture was first defined as described above. It then looks for the center of the spatial profile **t_steps** further along the dispersion axis, having first summed **t_nsum** dispersion lines. It continues in either direction from the original dispersion line until it reaches the edges of the image, or until the trace is lost. The set of centers as a function of position along the dispersion axis is then fit with a function of type **t_funct** with order **t_order**, and the average error of this fit is determined. If **t_niter** is greater than 1, any points that deviate from the fit by more than 3σ (by default) are rejected, the function is refit.

My experience is that either a straight line is good enough or you will need a modest order (2 or 3) cubic-spline. For most data, you will find an RMS of the fit to be a small fraction (≤ 0.1) of a pixel. Thus for the data here, reasonable choices of parameters are:

- **t_nsum=10**
- **t_step=10**
- **t_funct=legendre** or **spline3**
- **t_order=2**
- **t_niter=1**

Parameters Controlling the Summation and Background Subtraction: There are basically two types of extractions that you may choose: a simple "sum all the data points in the extraction window with no weighting" scheme, or a *variance*-weighted, "optimal" extraction scheme.

The "optimal extraction" algorithm was first developed by Keith Horne (1986 PASP **98**, 609) and makes use of the known noise characteristics of the CCD to do a mathematically optimal extraction, in that the weight that is used in computing the sum is inversely proportional to its statistical uncertainty.

If you use the optimal extraction algorithm, you have a bonus: you can clean out very deviant points from your spectra by turning on the **clean** algorithm. As you can imagine, it is somewhat tricky to distinguish cosmic-rays from, say, strong emission lines. The cleaning works by assuming that any variations in the *shape* of the spatial profile vary slowly with wavelength. For optimal extraction to work correctly, you must specify reasonable values for the gain and readnoise of the CCD.

Finally, you you must explicitly state whether or not the background level so lovingly set above is actually going to be subtracted or not.

The parameters that you need to set for the extraction are:

- **background** If you want to subtract the background using the parameters set above, you need to set this to "fit". For no background subtraction, set this to "none".
- **weights** Should be set to "variance" for optimal extraction, or "none" for just-add-'em-up.
- **clean** Should be set to "yes" for cleaning very deviant points; set to "no" to not clean. Note the subtlety that if you have **clean** turned on, your extraction algorithm will use a **weight**ing of "variance" even if you did set **weights** to "none": **clean=yes** demands variance weighting, and that's what you'll get.

However, you can turn **clean** off and still have optimal extraction if **weights** is set to "variance".

If you elect to do optimal extraction, you must also specify good values for:

- **saturatation** The saturation level of the CCD. For the data described here, the full-well is several hundred thousand electrons, and we are limited by the A-to-D converter to a maximum output of 32,767 ADUs. However, the DC-offset subtracted from these data was of order 300. Thus we will adopt a value of 32400.
- **readnoise** The read-noise of the CCD. This can either be a numerical value (in electrons) or it can be the name of the readnoise parameter in the image header. Note that ICE adds the readnoise to the header as "RDNOISE".
- **gain** The gain of the CCD. This can also be a numeric value (in electrons per ADU) or the name of the gain parameter in the image header. ICE calls the CCD gain "GAIN".
- **lsigma** is the lower rejection threshold for declaring a point really deviant. A value of 4 appears to be nicely conservative.
- **usigma** is the upper rejection threshold for declaring a point really deviant. A value of 4 appears to be nicely conservative.

For the data described here, we have chosen to do our extraction the following way:

- **background=fit**
- **weights=variance**
- **clean=yes**
- **saturation=32400**
- **readnoise=12**
- **gain=1.**
- **lsigma=4.**
- **usigma=4.**

Parameters Controlling The Output Format: Having settled on the pizza toppings, the type of crust, and the thickness, we still have to settle on the size we want to order. There are basically two choices. We can select an output image whose **format** parameter is either "onedspec or "multispec". If we choose the latter, we can opt for a special treat: **extras=yes** will give us an image that contains not only our optimally-extracted spectrum, but hidden away in another dimension will be the spectrum we would have gotten without optimal extraction, the spectrum of the sky (if we were using sky subtraction), and a spectrum containing the variance of the optimally-extracted spectrum.

The **onedspec** format is clearly the simpliest: it is simply a one-dimensional image.

The **multispec** format was developed to deal with multiple objects extracted from a single image. However, it can be useful even when dealing with a single object on the slit, in that it provides a mechanism for keeping the **extras** described above around. The image has three dimension: the x axis is the dispersion axis, the y axis is the aperture axis (which will be "1" for extracting a single object), and the z axis contains the "extras". (If sky subtraction and variance weighting are used, the z axis will have length 4, with $z=1$ being the optimally-extracted spectrum, $z=2$ being the "no-weights" extraction, $z=3$ being the sky spectrum, and $z=4$ being the "error spectrum" [sigma spectrum] of the optimal-extraction spectrum.) The IRAF routines refer to this third dimension as the **band**, while different apertures (objects) are referred to either as **apertures** or **lines**.

Most software works as well on these "multispec" images as on the "onedspec" images, and so for this exercise we will choose:

- **format=multispec**
- **extras=yes**

We're ready to order that pizza now!

3.3.2 Extracting a Well-Exposed Spectrum Using *apall*

The parameters for running **apall** are shown in Figs. 6 and 7.

We will go through all the steps of a run of **apall**, describing what you'll see and suggesting what diagnostics to look for. Nearly every detail of the extraction can be modified interactively as we go along, and an easy way to use **apall** is to begin with reasonable parameters (as described above) and then extract a sample spectrum, seeing "on the fly" what needs to be modified. Once this is done, you can go back and change the parameters (such as the function used to fit the trace) that will do a better job on your data.

When we begin running **apall** it will first put you in the aperture editor, at which point you can not only modify the extraction aperture, but also examine and modify the background fit. When you are done with this stage, you will move on to the trace routine,

```
                              I R A F
                Image Reduction and Analysis Facility
PACKAGE = slit
   TASK = apall

input   =              obj068  List of input images
(output =                    ) List of output spectra
(format =           multispec) Extracted spectra format
(referen=                    ) List of aperture reference images
(profile=                    ) List of aperture profile images

(interac=                 yes) Run task interactively?
(find   =                 yes) Find apertures?
(recente=                 yes) Recenter apertures?
(resize =                  no) Resize apertures?
(edit   =                 yes) Edit apertures?
(trace  =                 yes) Trace apertures?
(fittrac=                 yes) Fit the traced points interactively?
(extract=                 yes) Extract spectra?
(extras =                 yes) Extract sky, sigma, etc.?
(review =                 yes) Review extractions?

(line   =               INDEF) Dispersion line
(nsum   =                  10) Number of dispersion lines to sum

                               # DEFAULT APERTURE PARAMETERS

(dispaxi=                   1) Dispersion axis (1=along lines, 2=along cols)
(lower  =                 -5.) Lower aperture limit relative to center
(upper  =                  5.) Upper aperture limit relative to center
(apidtab=                    ) Aperture ID table (optional)

                                # DEFAULT BACKGROUND PARAMETERS

(b_funct=           chebyshev) Background function
(b_order=                   1) Background function order
(b_sampl=        -20:-8,8:20) Background sample regions
(b_naver=                -100) Background average or median
(b_niter=                   0) Background rejection iterations
(b_low_r=                  3.) Background lower rejection sigma
(b_high_=                  3.) Background upper rejection sigma
(b_grow =                  0.) Background rejection growing radius

                               # APERTURE CENTERING PARAMETERS

(width  =                 10.) Profile centering width
(radius =                 10.) Profile centering radius
(thresho=                  0.) Detection threshold for profile centering
```

Figure 6: The first half of the parameters for **apall.**

```
                                # AUTOMATIC FINDING AND ORDERING PARAMETERS

nfind   =                     1  Number of apertures to be found automatically
(minsep =                    5.) Minimum separation between spectra
(maxsep =                 1000.) Maximum separation between spectra
(order  =            increasing) Order of apertures

                                # RECENTERING PARAMETERS

(apertur=                      ) Select apertures
(npeaks =                 INDEF) Select brightest peaks
(shift  =                   yes) Use average shift instead of recentering?

                                # RESIZING PARAMETERS

(llimit =                 INDEF) Lower aperture limit relative to center
(ulimit =                 INDEF) Upper aperture limit relative to center
(ylevel =                   0.1) Fraction of peak or intensity for automatic
(peak   =                   yes) Is ylevel a fraction of the peak?
(bkg    =                   yes) Subtract background in automatic width?
(r_grow =                    0.) Grow limits by this factor
(avglimi=                    no) Average limits over all apertures?

                                # TRACING PARAMETERS

(t_nsum =                    10) Number of dispersion lines to sum
(t_step =                    10) Tracing step
(t_nlost=                     3) Number of consecutive times profile is lost
(t_funct=              legendre) Trace fitting function
(t_order=                     2) Trace fitting function order
(t_sampl=                     *) Trace sample regions
(t_naver=                     1) Trace average or median
(t_niter=                     1) Trace rejection iterations
(t_low_r=                    3.) Trace lower rejection sigma
(t_high_=                    3.) Trace upper rejection sigma
(t_grow =                    0.) Trace rejection growing radius

                                # EXTRACTION PARAMETERS

(backgro=                   fit) Background to subtract
(skybox =                     1) Box car smoothing length for sky
(weights=              variance) Extraction weights (none|variance)
(pfit   =                 fit1d) Profile fitting type (fit1d|fit2d)
(clean  =                   yes) Detect and replace bad pixels?
(saturat=                32400.) Saturation level
(readnoi=                    12) Read out noise sigma (photons)
(gain   =                    1.) Photon gain (photons/data number)
(lsigma =                    4.) Lower rejection threshold
(usigma =                    4.) Upper rejection threshold
(nsubaps=                     1) Number of subapertures per aperture
(mode   =                    ql)
```

Figure 7: The rest of the parameters for **apall** (Whew!)

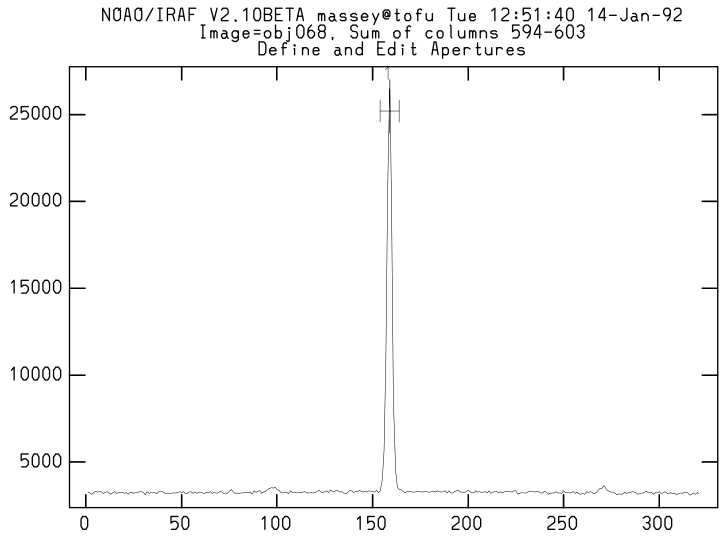

Figure 8: The extraction aperture has been found and centered. Normally you will also see details of the aperture listed along the bottom line of this plot.

and when you are done there, you will get to see the extracted spectrum. After you are done with each stage you will type a **q** to quit, and then automatically be moved onto the next step.

When we begin **apall** we will be asked a number of questions: "Find apertures? (yes)", "Resize apertures? (yes)" (if you have turned the **resize** option on), and finally "Edit apertures? (yes)". After you have accept the default (yes) answers to these questions, you will have entered the first stage: the aperture editor.

Inspecting and Altering the Aperture Location and Size: As soon as we answer "yes" to "Edit apertures?", we will find ourselves looking at a plot like that of Fig. 8

At this point you are in the aperture "editor", and there are a number of things you can do if you don't like the size (or location) of the extraction aperture. To get a complete list of the myriad of cursor options available, type a **?**.

Any of the parameters we set in the parameter file that defined the aperture (**lower**, **upper**, **line**, **width**, for instance) could be explicitly changed at this point by typing a colon, the parameter name, and a new value. For instance, we could set a new **upper** aperture limit by typing **:upper 7.** We could examine the aperture centered on the dispersion line (it's an image column but still called a dispersion line) near one end of the spectrum by typing **:line 1000** and use this as the basis of its center by then typing a **c**.

We could also play around with the size of the aperture using cursor commands. For

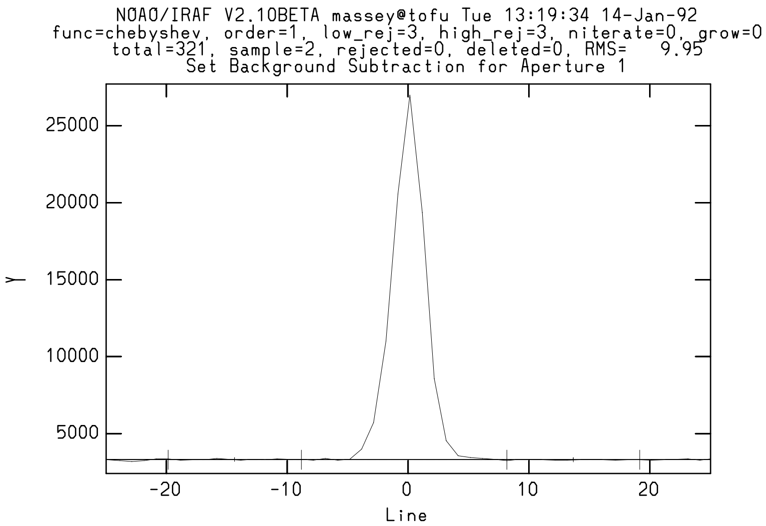

Figure 9: The background windows, and the fit of the data on this line are shown.

instance, we could set the width of the extraction aperture using the height of the cursor (e.g., where the "x" axis of the cursor intersects the profile) to define new **upper** and **lower** limits by typing a **y**. We could delete the aperture by typing a **d** and mark a new aperture by positioning the cursor and typing an **m**.

However, in this example we see that the edges of the aperture in fact nicely included the profile, and we are ready to check the background.

Inspecting and Altering the Background Windows and Fit: In order to check the background regions, we need to type a **b**. We will then be facing a plot that resembles that of Fig. 9.

Now that we are actually looking at the background fit, we are no longer quite in the aperture editor—we are instead in the interactive curve fitting routine called **icfit**. Thus trying to change the sample range or fitting order by **:b_sample −50:−8,8:50** or **:b_order 1** will in fact not work. But relax! The **icfit** routines will allow you to change these parameters anyway: type **?** for a complete list of the cursor commands available at this level. To change the background sample range we instead simply need to say **:sample −50:−8,8:50**. We can change the order of the fit by saying **:order 2**. We can change the averaging of points within the sample range by typing **:naverage −10**.

However, more useful is the ability to change things interactively with the cursor. To clear the current sample region type a **t**. To declare a new sample region on the left, move the cursor to the one side of the region you have in mind and type an **s**. Move it to the other end of the same region and type an **s** again. You will see the new region marked along the bottom of the plot. Mark another pair on the right side, type an **f** (for new fit) and you are back in business! To return to the aperture editor, type a **q**.

We could, if we choose, examine what the background looks like at some other point along the dispersion axis: type a **:line nnn** followed by a new **b** followed by a **q** when we are done.

Sometimes one finds that one would like to see a larger section of the sky than the default. When looking at the background plot one can extend the plot by typing **:\xwindow −50 50** for instance, followed by **:\redraw**. To see the entire spatial extent type **:\xwindow INDEF INDEF** followed by **:\redraw**.

Inspecting and Altering the Trace: Now we are about to leave the aperture editor and enter the routine that generates the trace and then lets us interact with the fit of the trace: type a **q** to go on. You will be asked if you want to run traces for the image, then whether you wish to fit the traces interactively, and finally if you want to fit the trace for the first aperture interactively. Accept the "yes" defaults for all three questions, and you be confronted with a plot like that shown on the left side of Fig. 10. We are once again in the interactive curve fitting program, and therefore we can alter the order or function not with **:t_order 3** or **:t_function spline3** but simply by **:order 3** and **:function spline3**. Note that we cannot alter the size of the trace step or sampling—we can only interact with the fit to the plot, or delete points.

Note that the fit on the left has an RMS of only 0.028 pixels—pretty darn good! The trace itself goes from 162 on the left to 155 on the right—the spectrum is tilted by -7 pixels over its 1200 pixel length. (If we hadn't traced at all our extraction aperture of ± 5 pixels would still have covered the centers of the profile, but we would have been losing a lot of the light at either end.) The points in the trace deviate slightly from a straight line at high column numbers, and it is enough for 4 points to be automatically rejected from our sample of 119 points. (Remember that our step size was 10; our spectrum is 1198 points long, so it makes sense that we have 119 points in the trace.) Can we improve the fit by going to a more flexible function? We can change the function to a cubic spline by doing a **:function spline3**, and we can change the order to 3 by doing a **:order 3**. An **f** then does a new fit. The fit we get out is shown on the right of Fig. 10. The RMS has been improved miniscully (to 0.020) The curve now actually goes through the slight upturn at the right. However, with an extraction aperture 11 pixels wide, this slight improvement will have no noticeable effect on the extraction.

We could, if we wish, delete a point by positioning the cursor near that point and typing a **d**. We could restore it by a **u**. Again, a **?** shows you the full array of cursor commands available within the **icfit** program.

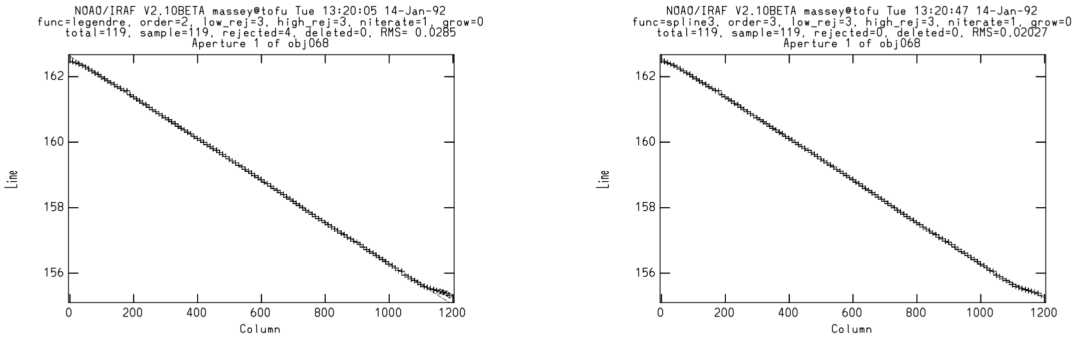

Figure 10: The linear fit (**function=legendre, order=2** on the left does a good job of fitting the trace, but doesn't quite match the upturn of the trace at high column numbers. Changing to a more flexible function (**function=spline3, order=3** on the right results in a good fit to the points of the trace throughout the entire spectrum.

Seeing Your Spectrum It is finally time to actually *see* the fruits of your labor: exit the trace fit by going a **q**. You will be asked if you wish to write the apertures to the **database**—again, accept the default "yes", as this will record the values of the aperture, background, and trace parameters in the local subdirectory **database** as explained above. You will next be asked if you wish to extract the spectrum. Accept the "yes" default. Next you will be asked if you wish to "review" the spectrum. Again, accept the "yes" default and you will see your spectrum looking something like that of Fig. 11. The image name created for this image will be **obj068.ms.imh**, with the "ms" extension signifying its **multispec** format. You can interact further with your spectrum using the powerful **splot** command, described in Sec. C.

3.3.3 Extracting a Weakly-Exposed Spectrum Using *apall*

The example used in 3.3.2 was that of a "well-exposed spectrum"—by which we really meant that there were plenty of counts in the continuum. Because we had plenty of counts, it was easy to *find* and *center* the spectrum, and it was easy to *trace* the spectrum. What if you were extracting a weakly-exposed star? Or what if you wanted to extract the spectrum of a stellar-like HII region which has *no* continuum but plenty of good strong emission lines?

The answer to this depends heavily on the data itself. If there are *any* discernible signal in the spectrum, you can get a good center for the spatial profile: simply pick a point along the dispersion axis where there is such a signal. (You can do this interactively when looking at the aperture editor by doing a **:line nnn**; to then recenter the aperture there, type a **c**). You can also increase the number of dispersion lines summed before finding the center by increasing the value of **nsum**.

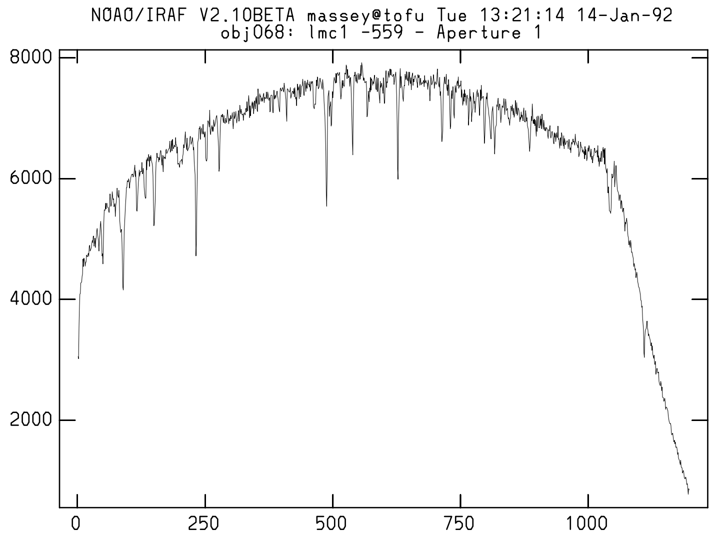

Figure 11: The final extracted spectrum.

Getting a good trace is trickier. If there is continuum throughout the spectrum, you may be able to do fine by increasing the number of dispersion lines summed in determining a point along the trace: increase **t_nsum**. (You may also want to make the step size coarser by also increasing **t_nstep**.) However, this won't help if there just isn't any continuum. In that case you will probably want to use a well-exposed stellar spectrum as a **reference** for the trace—the trace of the well-exposed spectrum will be used as the trace for the weak-continuum exposure object you are trying to extract. This will work fine if the tilt of the reference spectrum does actually match the tilt of the weak-continuum source, but remember our discussion earlier: this tilt is going to be partially due to the effects of differential refraction, and hence the deviation will depend upon the wavelength, the wavelength coverage, the airmass of the observation, and the orientation of the slit to the horizon. You can work out for yourself how significant this is by using the tables given by Fillippenko (1982 *PASP* **94**, 715) and a knowledge of the scale at the detector. As an example we consider the coude telescope at Kitt Peak and the observation of a star at an airmass of 2.0 observed due south (so the slit is oriented nearly at the parallactic angle). The difference in atmospheric refraction from 4000Å to 4500Å is 0.7 arcsec, and this translates to a 1.3 pixel shift for a typical CCD at the scale of the Kitt Peak coude spectrograph camera.

Thus we recommend using a hierarchical approach in dealing with the "weakly exposed" cases:

1. Weak continuum: use **apall** as described above but change the following parameters:

- **nsum** Increase so that it is easier to find a good center at some point along the dispersion axis. If possible, also change **line** to a dispersion line that contains strong continuum (the red end of the spectrum if the star is red or a region around an emission line).
- **t_nsum** Increase to something like what you set **nsum** so that many dispesion lines will be summed in looking for the center of the spatial profile at the next trace step.
- **t_nstep** You may want to increase this to something like **t_nsum**, but only if the tilt is small enough that the center at the next trace step is not too different than the center at the previous trace point.

2. No continuum: Use some other exposure as the reference spectrum for the trace. First decide if you can set the center of the aperture from the data itself, or do you need to actually adopt the center of the reference spectrum as well? (If so, you'd better hope the positioning was identical in the slit!) If you can find the center from the data itself, then

   - **apall** *spectrum* **reference=***refspect* **trace- recenter+** When you are in the aperture editor, play around with the **line** and **nsum** parameters as discussed above to establish a good center; you will have to type a **c** to actually recenter the aperture. Once that is done, the trace will automatically be shifted to this center at whatever **line** you use for establishing it.

   If you need to use the reference spectrum itself to establish not only the trace but the center as well, then

   - **apall** *spectrum* **reference=***refspect* **trace- recenter-** You will at least be able to inspect the final results.

3.4 Going Further: Calibration

3.4.1 Wavelength Calibration

Having extracted the spectrum of our program object, we will probably want to now put the data on a (linear) wavelength scale. The conceptional steps in doing the wavelength calibration were discussed in Sec. 2, and we will follow these closely:

Extracting the Comparison Exposure(s): The first step is to extract the comparison spectrum with the identical centering and trace used for the program spectrum. This can be accomplished by leaving the parameters for **apall** set as shown in Figs. 6 and 7, and doing the following:

apall comp030 out=cobj068 ref=obj068 recen- trace- back- intera-

This will extract a comparison spectrum from the image *comp030*. The extraction aperture used for extracting the spectrum from image *obj068* will be used centered exactly as it was in *obj068*, and not be recentered (**recen-**) on the comparison exposure. The comparison exposure will not be traced (**trace-**) but instead the trace from image *obj068* will be used. Finally, there will be no background subtracted (**back-**). (If we *did* subtract background from a comparison exposure there should be nothing left!) The **intera-** switch says not to bother to do it interactively. This step created a new spectrum called **cobj068.ms.imh**. If there were a second comparison exposure we wished to use with the first in order interpolate in time, say, to the object exposure, we should have used an output name like **cobj068a.ms.imh**, and extract the other exposure now:

apall comp039 out=cobj068b ref=obj068 recen- trace- back- intera-

Determinating the Dispersion Solution: The next step in the reduction process is to run the task **identify** on the extracted comparison spectrum. This will allow us to identify which comparison lines have what laboratory wavelengths, and to fit a function to these data. The parameters for the **identify** task are shown in Fig. 12. We have set the **coordlist** parameter to be that containing the lab wavelength of the HeAr lines prominent in the CTIO comparison lamps; **page linelists$README** to see what other choices are available. We have also set the parameter **fwidth** to the base-to-base width determined earlier, and set the number of rejection iterations to 1.

Running the task will present you with a plot of the comparison spectrum. To obtain a good dispersion solution, try the following steps, following along with the sequence shown in Fig. 13.

- Identify three or four good lines along the spectrum by positioning the cursor to the comparison line and typing an **m**. This will "mark" the line, and invoke the centering algorithm (Fig. 13, top left). Each time, you will be asked for the corresponding wavelength. You can enter an approximate value for this, and it will choose the nearest entry to this value from the table of laboratory wavelengths you specified with **coordlist**. [2]

- Do an **f** to obtain a "fit" of wavelength as a function of pixel number using the two or three points you established in the previous step. You are now in your old friend, the interactive curve fitting routine **icfit**. Change the fitting function to a straight line (**:funct cheb** followed by a **:order 2** followed by an **f**), and hit a **j** to see a plot of residuals (in Å) vs. pixel number (Fig. 13, top right). Are any points very

[2] Watch this step: if it simply repeats your approximate value of the wavelength back to you, you either have mistyped the value, or the coordinate table isn't working for some reason.

```
                    Image Reduction and Analysis Facility
PACKAGE = slit
   TASK = identify

images  =       cobj068.ms.imh  Images containing features to be identified
(section=          middle line) Section to apply to two dimensional images
(databas=             database) Database in which to record feature data
(coordli=linelists$ctiohear.dat) User coordinate list
(nsum   =                   10) Number of lines or columns to sum in 2D images
(match  =                  10.) Coordinate list matching limit in user units
(maxfeat=                   50) Maximum number of features for automatic
(zwidth =                 100.) Zoom graph width in user units
(ftype  =             emission) Feature type
(fwidth =                  7.5) Feature width in pixels
(cradius=                   5.) Centering radius in pixels
(thresho=                  10.) Feature threshold for centering
(minsep =                   2.) Minimum pixel separation
(functio=              spline3) Coordinate function
(order  =                    1) Order of coordinate function
(sample =                    *) Coordinate sample regions
(niterat=                    1) Rejection iterations
(low_rej=                   3.) Lower rejection sigma
(high_re=                   3.) Upper rejection sigma
(grow   =                   0.) Rejection growing radius
(autowri=                   no) Automatically write to database
(graphic=             stdgraph) Graphics output device
(cursor =                     ) Graphics cursor input
```

Figure 12: The parameters for the **identify** task. Note the specific reference to the wavelength table **coordlist** and the feature width **fwidth** being set to the base-to-base width of the comparison lines determined earlier.

discrepant, indicating that you misidentified one of the lines? If so, you can delete it with a **d**.

- Type a **q** to return to the plot of the comparison spectrum. Type an **l** to automatically mark and center all the other comparison lines in the coordinate list that are not too far from the location predicted by the preliminary fit (Fig. 13, middle left).

- Type a **f** to obtain a new fit. Inspect the residuals, and you will doubtless see a systematic effect with pixel number. (Fig. 13, middle right). Try high orders or other functions. For the data we are using here, we find that a third-order, cubic-spline (**:function spline3 :order 3**) produces a good fit (Fig. 13, bottom left), but in general it's good to avoid cubic-splines if you can—they are a little *too* flexible. Type an **l** to see the "non-linear" part of the function, and how well this is fitting the points (Fig. 13, bottom right).

- Return to the identification part of the routine by doing a **q**. If you were happy with your fit (comparison lines are marked throughout your spectrum, your fit has a low RMS [.04Å in this case, which is acceptable given that the dispersion is 1Å/pixel], and no obvious systematics in the residuals plot), you can exit the routine by doing another **q**. Otherwise you may wish to do another **l** to use the improved solution to identify additional lines, or do a **z** followed by an +'s to examine each identified line in detail, etc. A **?** will show you the full range of cursor keys available.

Doing More Dispersion Solutions The Easy Way: Additional comparison spectra can be handled very easily based upon this first solution, using the **reidentify** task. Edit the parameter task for **reidentify** so it resembles that of Fig. 14. This task will use the lab wavelengths and line centers from the entry in the local directory **database** for the reference spectrum as a starting point for the new spectrum. It will attempt to center all the lines using the old line centers as the starting points: if it loses any lines, **reident** will fail if **nlost=0**. When we run **reidentify** in interactive mode (**interact=yes**) we will be told what the RMS is of fit, and asked if we wish to interactively change it using **identify**. We expect the RMS of the new fit to be similar to that of the old fit. Note that by having using **niter=1** in the parameters for **identify** when we did the previous solution, it means that **reidentify** *can* in fact reject a very deviant point without the feature being "lost".

Assigning the Dispersion Solution(s) to the Object Exposure Having obtained the dispersion solution(s) for the extracted comparison exposures, we must now somehow link each solution with the program spectrum. This link is provided by placing the keyword **REFSPEC1=***compimagename* in the header of the program object. If a second

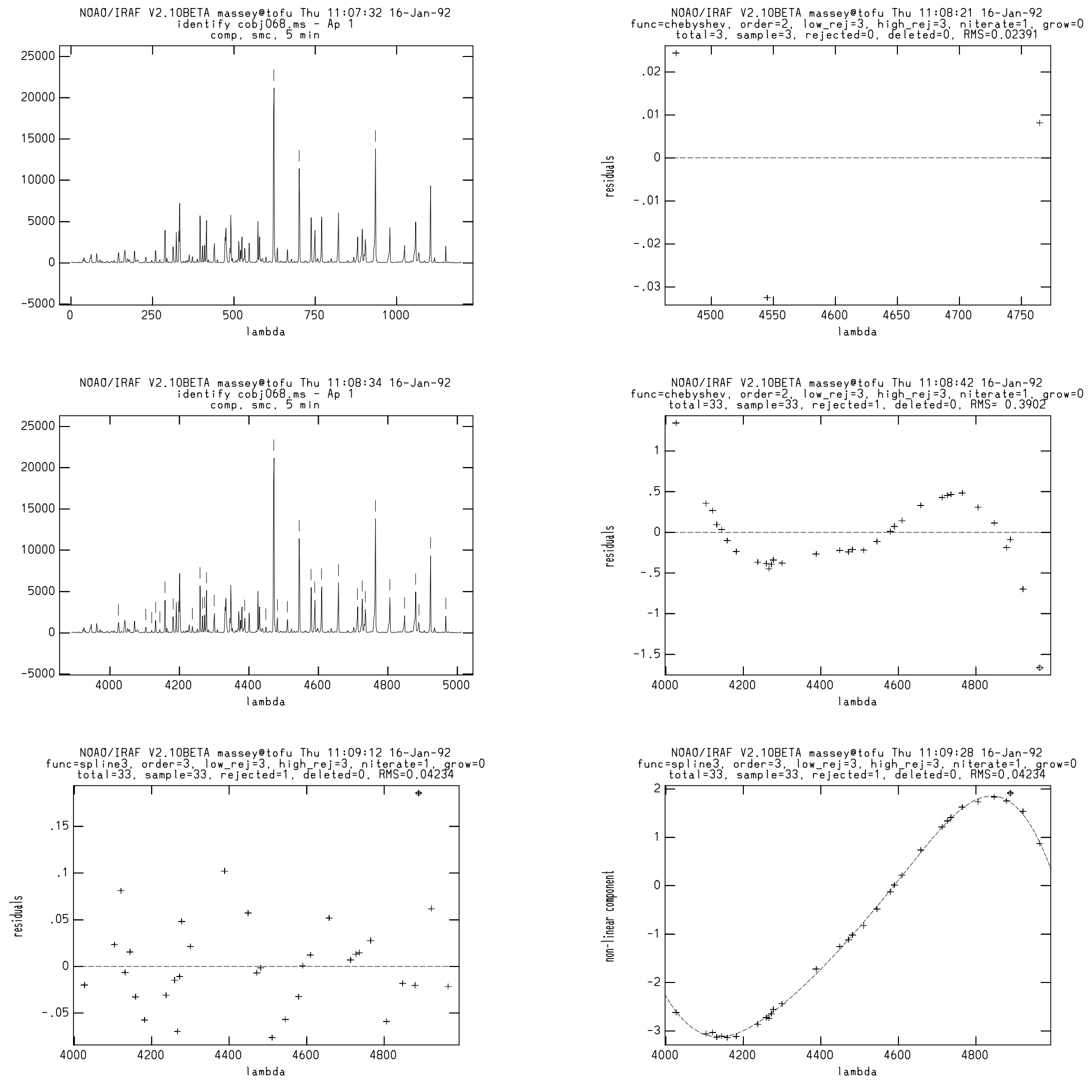

Figure 13: Three comparison lines are initially identified (upper left) to create a linear fit of wavelength as a function of pixel number (upper right). The RMS of this straight-line fit is 0.02Å over this very limited wavelength range. This preliminary fit is then used to automatically find additional lines in the spectrum (middle left). These additional lines show a strong systematic trend with pixel number when only a linear solution is used, with residuals of 0.4Å (middle right). Making the function more flexible (third-order, cubic spline) removes this systematic trend trend and reduces the residuals to 0.04Å (lower left). The non-linear part of this dispersion solution looks sensible (lower right).

```
                    Image Reduction and Analysis Facility
PACKAGE = kpnoslit
   TASK = reidentify

referenc=          cobj068a.ms  Reference image
images  =          cobj068b.ms  Images to be reidentified
(interac=                 yes) Interactive fitting?
(section=          middle line) Section to apply to two dimensional images
(newaps =                 yes) Reidentify apertures in images not in reference?
(overrid=                 yes) Override previous solutions?
(refit  =                 yes) Refit coordinate function?

(trace  =                  no) Trace reference image?
(step   =                  10) Step for tracing an image
(nsum   =                  10) Number of lines or columns to sum
(shift  =                  0.) Shift to add to reference features
(nlost  =                   0) Maximum number of features which may be lost

(cradius=                  5.) Centering radius
(thresho=                 10.) Feature threshold for centering
(addfeat=                  no) Add features from a line list?
(coordli=linelists$ctiohear.dat) User coordinate list
(match  =                 10.) Coordinate list matching limit in user units
(maxfeat=                  50) Maximum number of features for automatic identi
(minsep =                  2.) Minimum pixel separation

(databas=             database) Database
(logfile=              logfile) List of log files
(plotfil=                    ) Plot file for residuals
(verbose=                 yes) Verbose output?
(graphic=             stdgraph) Graphics output device
(cursor =                    ) Graphics cursor input
answer  =                  no  Fit dispersion function interactively?
(mode   =                  ql)
```

Figure 14: Parmaeters for the **reidentify** task.

comparison is used, then a second keyword (**REFSPEC2=***othercompimage*) is placed in the header, and the fractional amount of each to be used is given.

There are several ways to accomplish this. If your entire observing run consisted of a single program exposure and a single comparison exposure, the easiest thing to do would be to simply add this keyword to the header using the task **hedit**:

hedit obj068.ms.imh REFSPEC1="cobj068.ms "add+ ver- show+

would do it. Or if you had taken two comparison exposures, one immediately before, and one immediately afterwards, you might want to say

hedit obj068.ms.imh REFSPEC1="cobj068a.ms 0.5" add+ ver- show+
hedit obj068.ms.imh REFSPEC2="cobj068b.ms 0.5" add+ ver- show+

However, it is more likely that you have better observing weather than that! The task **refspec** provides a number of schemes by which you can automatically assign comparison spectra to object spectra. The help page for **refspec** will show you many possibilities; we list a few of the more likely below:

- **assignments based upon time:** You can automatically assign the dispersion solutions to the program spectra based upon the julian day. First run the task **setjd** to add the "normal" julian day (jd), the heliocentric julian day (hjd), and a "local" julian day (ljd) that starts at local noon to the headers. If we were to then run the **refspec** task we could specify that we wanted to take the comparison spectrum that was taken (a) nearest in time to the program exposure (**select=nearest**), (b) nearest in time before the program exposure (**select=preceding**), (c) nearest in time after the program exposure (**select=following**), or (d) interpolate in time between bracketing comparison exposures (**select=interpolate**). Thus to do the latter we would be want to do

  - **setjd *.imh**
  - **refspec obj*.ms.imh reference=cobj*.ms.imh select=interp sort=jd confirm- verb+**

  The task is sophisticated enough to distiguish between days using the **group** parameter, which defaults to "ljd", so you will not be trying to use comparison spectra from the wrong day.

- **assignment table:** You can create an assignment table that explicitly states what comparison exposure goes with which object exposure. For instance, a table might look something like that of Fig. 15

  You could, if you wish, specify that you want to average the dispersion solutions if two appear on a line:

  refspect obj*.ms.imh ref=cobj*.ms.imh refe=reftable select=ave verb+

```
obj032.ms cobj032a.ms,cobj032b.ms
obj068.ms cobj068.ms
obj069.ms
obj070.ms
obj100.ms cobj032a.ms
```

Figure 15: This arc reference table *reftable* will link the dispersion solutions for *cobj032a.ms, cobj032b.ms* with the extracted program spectrum *obj032.ms*, will link the dispersion solutions for **cobj068.ms** with the extracted program spectra *obj068.ms, obj069.ms, obj070.ms*, and will link the dispersion solution for *cobj032a.ms* with the extracted program spectrum *obj100.ms* using whatever scheme is given in **refspec**.

Applying the Dispersion Solution(s) To Your Program Objects: The final step of applying the wavelength calibration is to actually use the dispersion solution(s) to set the wavelength scale. The task for this is **dispcor**, and a sample parameter file is given in Fig. 16.

We first will put all the object spectra into a file called *objects*:

files obj*.ms.imh > objects

When we run **dispcor** we will get new images, each with the letter "d" appended to the old name. We choose to actually *linearize* the spectra—this is no longer strictly necessary with the "world coordinate system" provided by V2.10 of IRAF, but none of us have much experience with this yet, and so for now, let's just do the same thing we've come to know and love. By setting **global=yes** we would have assured that all the object spectra will have the identical wavelength scale. Run **dispcor** and you will see something like the output in Fig. 17. We can explicitly change the defaults for any three of the four dispersion parameters (starting wavelength, ending wavelength, wavelength per pixel, and number of output pixels), but you will want to keep the dispersion value within factors of 20% or so to keep the interpolation errors small.

3.4.2 Flux Calibration

If you have observed spectrophotometric standards you can calibrate your program observations to real flux units. The directory **onedstds$** houses several subdirectories containing the standard star calibrations. The file **onedstds$README** explains where these calibrations come from; at present, all of these data are more or less on the Hayes and Latham (1975 *ApJ* **159**, 175) calibration of Vega, essentially that of the Palomar AB79 system of Oke and Gunn (1983 *ApJ* **266**, 713). Nevertheless, let the buyer beware, the sanctity of the fluxes used are up to the user to evaluate.

Each star has its own individual file within a given subdirectory, which specifies the

```
PACKAGE = kpnoslit
   TASK = dispcor

input   =            @objects  List of input spectra
output  =         d//@objects  List of output spectra
(lineari=                 yes) Linearize (interpolate) spectra?
(databas=            database) Dispersion solution database
(table  =                    ) Wavelength table for apertures
(w1     =               INDEF) Starting wavelength
(w2     =               INDEF) Ending wavelength
(dw     =               INDEF) Wavelength interval per pixel
(nw     =               INDEF) Number of output pixels
(log    =                  no) Logarithmic wavelength scale?
(flux   =                 yes) Conserve flux?
(samedis=                  no) Same dispersion in all apertures?
(global =                  no) Apply global defaults?
(confirm=                 yes) Confirm dispersion coordinates?
(listonl=                  no) List the dispersion coordinates only?
(verbose=                 yes) Print linear dispersion assignments?
(logfile=                    ) Log file
(mode   =                  ql)
```

Figure 16: The parameters for **dispcor**.

```
kp> dispcor
List of input spectra (@objects):
List of output spectra (d//@objects):
dobj023.ms.imh: ap = 1, w1 = 3894.174707202079, w2 = 5008.458618630227,
                                          dw = 0.9308971691129057, nw = 1198
  Change wavelength coordinate assignments? (yes|no|NO): yes
  Starting wavelength (3894.1747072021): 3895.
  Ending wavelength (5008.4586186302): INDEF
  Wavelength interval per pixel (0.93089716911291): 1.0
  Number of output pixels (1198): INDEF
dobj023.ms.imh: ap = 1, w1 = 3895.000000000001, w2 = 5008., dw = 1., nw = 1114
  Change wavelength coordinate assignments? (yes|no|NO) (yes): NO
dobj023.ms.imh: ap = 1, w1 =    3895., w2 =    5008., dw =       1., nw = 1114
dobj068.ms.imh: ap = 1, w1 =    3895., w2 =    5008., dw =       1., nw = 1114
```

Figure 17: When running **dispcor** it is straightforward to change the wavelength assignments.

wavelength, width of the bandpass, and the magnitude of the standard star.

The procedure for flux calibrating requires executing the following steps:

- Correct the airmasses to mid-exposure by running the **setairmass** routine.
- Establish the correct subdirectory for the standard star observations in the parameter file of the package (e.g., **kpnoslit**, **ctioslit**, or **kpcoude**).
- Establish the correct file to use for extinction correction of your data. If you have chosen to use the **kpnoslit** package to reduce data taken at KPNO the default would already be sensibly set to **onedstds$kpnoextinct.dat**. If you had chosen to use the **ctioslit** package to reduce CTIO data this will have been sensibly set to **onedstds$ctioextinct.dat**. However, since we have chosen the rather bizarre route of choosing to reduce CTIO data using the **kpnoslit** package, we will have to change this.
- For each standard star observation, execute the task **standard**. This routines takes a single observation of a spectrophotometric flux standard and asks you to give the version of its name listed in the calibration subdirectory. The **standard** routine then integrates your data over the appropriate bandpasses, divides by the exposure time, and outputs a single file containing an observation-by-observation listing of the observed counts within each bandpass along with the standard star fluxes.
- The **sensfunc** task will allow you to interactively fit the sensitivity function as a function of wavelength using the output file from **standard**. Extinction corrections using the standard extinction table you adopted will be used, or you can try to choose to determine the extinction empirically from your data. You can perform "grey shifts" of a particular observation, delete points or observations, and generally interact until you have a satisfactory fit to the points.
- The sensitivity function determined by **sensfunc** is applied to your data by the **calibrate** task. The program spectra are corrected for atmospheric extinction, divided by the exposure time, and finally transformed using the sensitivity curve.

Let us run through this step-by-step for the data we have been discussing.

We begin by running **setairmass** task. The results are shown in Fig. 18.

Flux standards were observed for the data being used here primarily to help in flattening the continuum; we were not interested in obtaining good spectrophotometry but we did require frequent spacing of the calibration points so we could remove the vignetting at the high wavelength end. Thus we observed only a single standard each night, but we did choose them from the "Kitt Peak Spectrophotometric Standards" list, as these have flux points every 50Å. The subdirectory for the calibration data is **onedstds$spec50cal/**. In

```
kp>setairmass *.ms.imh
#              Image    UT middle  effective  begin   middle     end   updated
# Setairmass: Observatory parameters for Cerro Tololo Interamerican Observatory
#       latitude = -30.16527778
       obj058.ms.imh    0:34:41.0   1.3641   1.3606   1.3641   1.3676  yes
       obj068.ms.imh    5:47:50.0   1.3617   1.3599   1.3617   1.3634  yes
       obj090.ms.imh    0:39:13.0   1.4016   1.3978   1.4016   1.4055  yes
       obj129.ms.imh    0:42:26.0   1.4366   1.4323   1.4365   1.4408  yes
```

Figure 18: Running **setairmass**.

addition, we must make sure that the atmospheric extinction table we are using is the correct one: **onedstds$ctioextinct.dat** in this case. We have been running all of the spectral extraction and calibration routines from the **kpnoslit** package, and it thus this package whose parameters we must now edit. We do an **epar kpnoslit**, and see something resembling that of Fig. 19

We now need to run **standard** for each observation of a standard star we make. The output file defaults to **std** but can be called anything; the important thing is that we keep this name the same for each execution of **standard** so that new data will be appended to the same file. A sample run is shown in Fig. 20.

We are now ready to do the determine the sensitivity function for our data by executing **sensfunc**. The parameters are shown in Fig. 21. Execute the task and you will be presented with a plot like that shown on the left half of Fig. 22. The top plot of the pair shows the sensitivity function vs. wavelength. The bottom plot of the pair shows the residuals (magnitude) vs. wavelength.

We can see immediately that using a dinky slit that was about half or less of the seeing fwhm did not lead to incredibly good absolute spectrophotometry (nor did observing though clouds; note the title of the second observation of Feige110 shown in Fig. 20): the observation-to-observation residuals span a range of nearly 0.6 mags. The full range of cursor options is available by typing a **?**. For the data presented here, we decide to do a grey-shift by typing a **s**. We next find that one of the three observations has a different slope than the other two, and we decide to ax it by doing a **d** and specifying that we wish to delete all the points associated with that particular observation by saying “star”. Finally we up the order until the curve bends enough to deal with the extreme tilts at the end of the spectrum. The final fit is shown on the right of Fig. 22.

The final step is to actually apply this sensitivity function to our data, using **calibrate**. The parameter set for **calibrate** is shown in Fig. 23. We have assumed that you have gathered together the spectra you wish to calibrate into a file called *examples*; you could do this by having done a **files d*.ms.imh > examples**. The output files will be given

```
                                   I R A F
                    Image Reduction and Analysis Facility
PACKAGE = imred
   TASK = kpnoslit

(extinct= onedstds$ctioextinct.dat) Extinction file
(caldir =  onedstds$spec50cal/) Standard star calibration directory
(observa=         observatory) Observatory of data
(interp =               poly5) Interpolation type

(databas=            database) Database
(verbose=                 yes) Verbose output?
(logfile=             logfile) Log file
(plotfil=                    ) Plot file

(nsum   =                   1) Aperture sum for 2D images
(records=                    ) Record number extensions
(version= KPNOSLIT V3: January 1992)
(mode   =                  ql)
($nargs =                   0)
```

Figure 19: The package parameters for **kpnoslit**, modified to specify the ctio atmospheric extinction table. Note the final "/" on the subdirectory for **caldir**.

```
kp> standard
Input image file root name: obj058.ms
Output flux file (used by SENSFUNC) (std):
dobj058.ms[1]: f110!
Star name in calibration list: ?
Standard stars in onedstds$spec50cal/

bd284211        feige66         hd217086        pg0216032       pg0939262
cygob2no9       feige67         hilt600         pg0310149       pg1121145
eg81            g191b2b         hz14            pg0823546       pg1545035
feige110        gd140           hz44            pg0846249       pg1708602
feige34         hd192281        pg0205134       pg0934554       wolf1346
Star name in calibration list (?): feige110
obj058.ms[1]: Edit bandpasses? (no|yes|NO|YES|NO!|YES!) (no): no
kp> standard dobj090.ms
Output flux file (used by SENSFUNC) (std):
dobj090.ms[1]: F110 in some clouds
Star name in calibration list (feige110):
dobj090.ms[1]: Edit bandpasses? (no|yes|NO|YES|NO!|YES!) (no):
kp> standard
Input image file root name: dobj129.ms
Output flux file (used by SENSFUNC) (std):
dobj129.ms[1]: Feige110
Star name in calibration list (feige110):
dobj090.ms[1]: Edit bandpasses? (no|yes|NO|YES|NO!|YES!) (no):
```

Figure 20: Running **standard** for all the standards. By answering with a **?** to "Star name in calibration list" we were automatically shown the list of legal star names. The entry is case and punctuation insensitive, however.

```
PACKAGE = kpnoslit
   TASK = sensfunc

standard=                 std  Input standard star data file (from STANDARD)
sensitiv=                sens  Output root sensitivity function imagename
(apertur=                   ) Aperture selection list
(ignorea=                yes) Ignore apertures and make one sensitivity funct
(logfile=            logfile) Output log for statistics information
(extinct=       )_.extinction) Extinction file
(newexti=         extinct.dat) Output revised extinction file
(observa=         observatory) Observatory of data
(functio=             spline3) Fitting function
(order  =                   6) Order of fit
(interac=                yes) Determine sensitivity function interactively?
(graphs =                 sr) Graphs per frame
(marks  =      plus cross box) Data mark types
(cursor =                   ) Graphics cursor input
(device =           stdgraph) Graphics output device
answer  =                yes  (no|yes|NO|YES)
(mode   =                 ql)
```

Figure 21: The parameters for **sensfunc**.

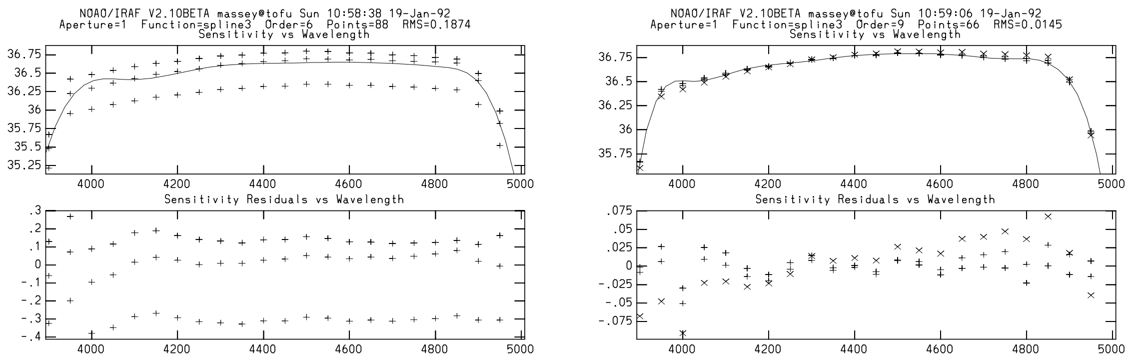

Figure 22: Making a good fit with **sensfunc**. The pair of plots on the left are from the original fit; the pair on the right is what we have after (1) doing a grey shift with **s**, (2) deleting one of the three stars using a **d**, and (3) uping the order to 9 with an **:order 9**.

```
PACKAGE = kpnoslit
   TASK = calibrate

input   =             @examples  Input spectra to calibrate
output  =          c//@examples  Output calibrated spectra
(extinct=                   yes) Apply extinction correction?
(flux   =                   yes) Apply flux calibration?
(extinct=         )_.extinction) Extinction file
(observa=          observatory) Observatory of observation
(ignorea=                   yes) Ignore aperture numbers in flux calibration?
(sensiti=                  sens) Image root name for sensitivity spectra
(fnu    =                    no) Create spectra having units of FNU?
(mode   =                    ql)
```

Figure 23: The parameter file for **calibrate**.

new names made of the old names with a "c" in front.

The final, fluxed spectrum is shown in Fig. 24. We can see that we didn't do a great job at the ends, particularly where the sensitivity function has been extrapolated beyond the calibration points on either side. But overall the spectrum is much flatter than that shown in Fig. 11.

3.4.3 Normalization

Instead of trying to flatten our ugly spectrum by fluxing it, we could instead simply attempt to fit a function to the continuum and normalize the spectrum by that fit. The task for this is called **continuum**. Parameters for this task are shown in Fig. 25.

The trick in running **continuum** is to have a sufficiently flexible function that you

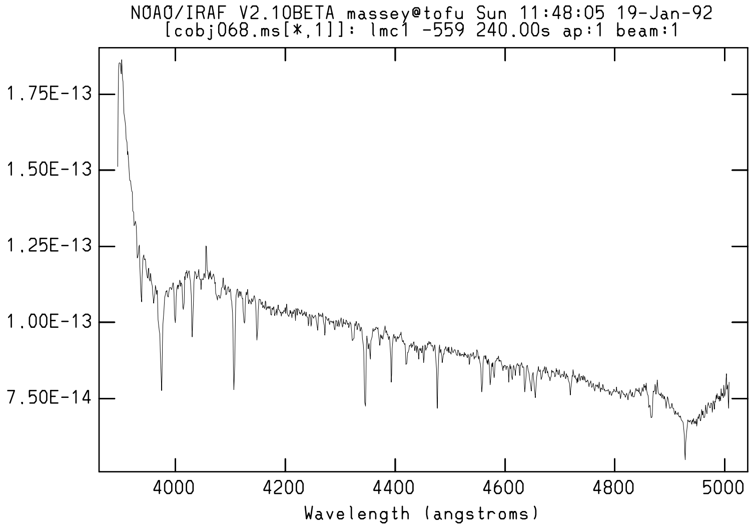

Figure 24: The final extracted, flux-calibrated spectrum.

```
PACKAGE = kpnoslit
   TASK = continuum

input   =       dobj068.ms.imh  Input images
output  =       nobj068.ms.imh  Output images
(lines  =                   *) Image lines to be fit
(type   =               ratio) Type of output
(replace=                  no) Replace rejected points by fit?
(wavesca=                 yes) Scale the X axis with wavelength?
(logscal=                  no) Take the log (base 10) of both axes?
(overrid=                  no) Override previously fit lines?
(listonl=                  no) List fit but don't modify any images?
(logfile=             logfile) List of log files
(interac=                 yes) Set fitting parameters interactively?
(sample =                   *) Sample points to use in fit
(naverag=                   1) Number of points in sample averaging
(functio=             spline3) Fitting function
(order  =                   1) Order of fitting function
(low_rej=                  2.) Low rejection in sigma of fit
(high_re=                  0.) High rejection in sigma of fit
(niterat=                  10) Number of rejection iterations
(grow   =                  1.) Rejection growing radius in pixels
(markrej=                 yes) Mark rejected points?
(graphic=            stdgraph) Graphics output device
(cursor =                    ) Graphics cursor input
ask     =                 yes
(mode   =                  ql)
```

Figure 25: Parameter file for **continuum**.

actually are fitting the continuum, even over those extreme bends shown in Fig. 11, but to not fit true spectral features. We start off by having a large number of rejection iterations, a low sigma for the rejection of points, and decide to reject only high points. (You should reconsider this for emission-lined objects!)

The top graph in Fig. 26 shows the initial fit. Although all the absorption lines have been automatically rejected, so have the very bent ends of the spectrum. By upping the order to 25 or so we are now doing much better—the garbage on the extreme left is still being automatically rejected, and hence won't divide out, but that's not too bad. We can inspect the ratio by striking the **k** key—afterall, this is just the interactive curve fitting program **icfit** that we have used numerous times in this reduction. When we get done we can examine the final spectrum using **splot**, shown in the bottom-most figure. (More about **splot** can be found in Sec. C.)

4 Doing It All Automatically: *doslit*

In Sec. 3 we went through the individual steps of extracting and calibrating your spectra. You may have felt "There *must* be an easier way." You're right, there is, and the name of this is the task **doslit**. This task combines *all* the steps discussed above into one very powerful and versatile routine. We've dealt with the individual steps because knowledge of them is still essential for dealing with **doslit** in an intelligent manner—but in fact, **doslit** is all you really need to use to reduce your spectra.

When we run **doslit** the following things will happen:

- The **setairmass** and **setjd** tasks will be automatically and invisibly run to update the headers with the effective airmass, the UT time of mid-exposure, and the various julian dates.
- The program exposures and comparison exposures will be identified based upon the value of "IMAGETYP" in the header.
- You will be given the chance to examine the apertures and background region and the traces for all the program objects interactively.
- If you have chosen to dispersion-correct your data, the first comparison spectrum will be extracted and, if a wavelength solution has not been previously found for it, you will do this first one interactively. (The dispersion solution of all subsequent comparison spectra will be found automatically via **reidentify**.)
- If you have chosen to flux-calibrate your spectra, the standard star data will be extracted, dispersion corrected, and run through **standard** and **sensfunc** to produce the sensitivity function, if this has not previously been done.

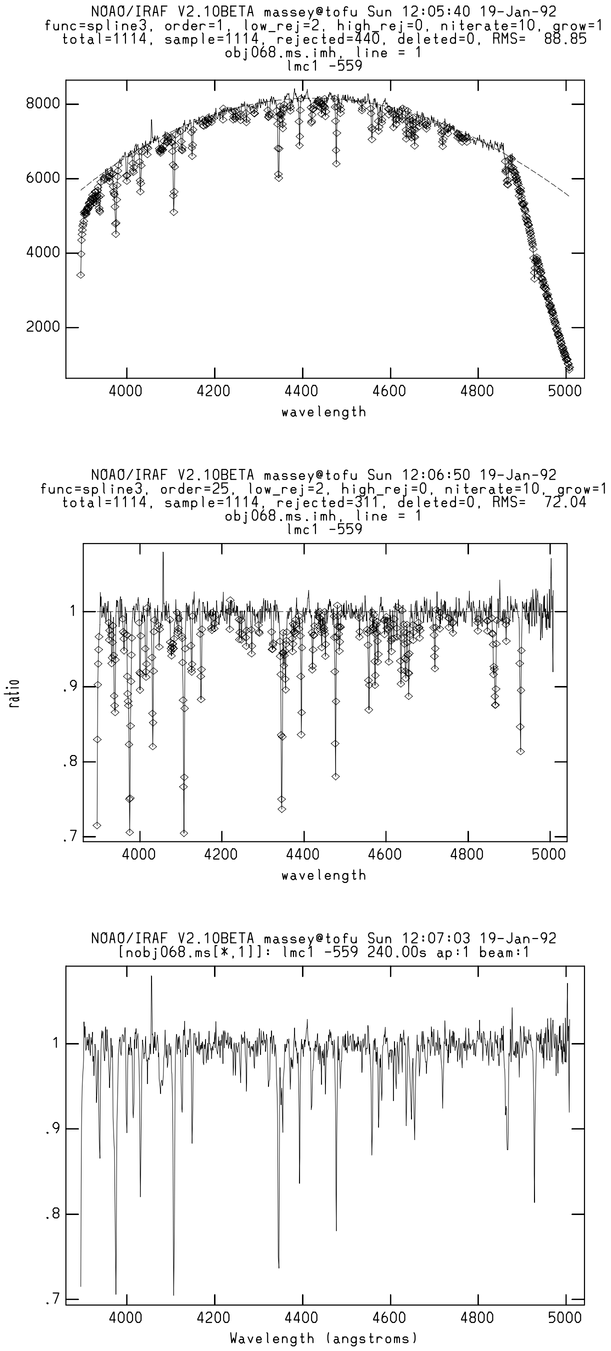

Figure 26: Normalizing to the continuum is straightforward, even for this twisty spectrum. Compare the bottom plot to those shown in Figs. 11 and 24.

- Finally, each program spectrum will be processed in turn:
  - The program spectrum will be extracted.
  - If you have chosen to dispersion-correct your data, the appropriate comparison spectrum or spectra will be identified using whatever scheme you have specified (closest in time, interpolated in time, selected from a reference table, etc.) The comparison spectrum will then be extracted using the identical trace and center as that of the program object, and the dispersion solution determined via **reidentify**. The program spectrum will then be dispersion corrected.
  - If you have also chosen to flux-calibrate, the dispersion-corrected program spectrum will be corrected for extinction and the sensitivity function used to convert to flux.
  - If you choose, each program spectrum will be plotted on the screen as it is processed.

You can rerun **doslit** any number of times, including new spectra, or additional steps (such as flux calibration) and only those things that have not yet been done will be done (assuming that the **redo** parameter is off!) In addition, there is a **quicklook** option which, if selected, will provide only minimal interaction with the spectrum. We will use **doslit** to do the extractions and dispersion-correction for the same data as we did step-by-step in Sec. 3. We begin by examining the parameters of **doslit** as shown in Fig. 27. We have modified the parameters to specify that we wish to dispersion correct our spectra and plot the final spectra as we finish; we have also specified the readnoise, the gain, the dispersion axis, and the width of the stellar profile. We have turned **clean** on, which, as you'll remember, means that the extraction algorithm is **variance** weighting regardless of what we specify for **weights** in **sparams**.

Most of the guts of the parameters, however, are tucked away in the parameter file **sparams**. To edit these we need to set the cursor on the **sparams** line and type an **:e**. Alternatively, we could simply type **sparams** after we are done editing the main parameter file for **doslit**. The parameters stored in **sparams** are shown in Figs. 28 and 29, and will be familiar from our use of **apall**. Note that in addition to the extraction parameters (which are identical to those in **apall** described in Sec. 3.3.2), we also have parameters controlling the wavelength calibration tasks **identify**, **reidentify**, **refspec**, and **dispcor**, as well as the flux-calibration task **sensfunc**.

We need to select the assignment scheme for the comparison spectra. In the parameters shown in Fig. 29 we chose to interpolate in time (jd), but automatically ignoring spectra not taken the same night (using the local-noon day number ljd to distinguish one night from the next). Alternatively, we could also have created a file with explicit references of object and comparison spectra. An example of such a table is shown in Fig. 30. Note that this table is much simplier in form than that shown in Fig. 15, as the *original* names of the

```
PACKAGE = kpnoslit
   TASK = doslit

objects =                        List of object spectra
(arcs   =                      ) List of arc spectra
(arctabl=                      ) Arc assignment table (optional)
(standar=                      ) List of standard star spectra

(readnoi=                   12.) Read out noise sigma (photons)
(gain   =                    1.) Photon gain (photons/data number)
(dispaxi=                     1) Dispersion axis (1=along lines, 2=along cols)
(width  =                   10.) Width of profiles (pixels)

(dispcor=                   yes) Dispersion correct spectra?
(extcor =                    no) Extinction correct spectra?
(fluxcal=                    no) Flux calibrate spectra?
(resize =                    no) Automatically resize apertures?
(clean  =                   yes) Detect and replace bad pixels?
(splot  =                   yes) Plot the final spectrum?
(redo   =                    no) Redo operations if previously done?
(update =                    no) Update spectra if cal data changes?
(quicklo=                    no) Minimally interactive quick-look?
(batch  =                    no) Extract objects in batch?
(listonl=                    no) List steps but don't process?

(sparams=                      ) Algorithm parameters
(mode   =                    ql)
```

Figure 27: The parameters for the **doslit** task.

```
PACKAGE = kpnoslit
   TASK = sparams

(line   =                INDEF) Default dispersion line
(nsum   =                   10) Number of dispersion lines to sum
(extras =                  yes) Extract sky, sigma, etc.?

                                -- DEFAULT APERTURE LIMITS --
(lower  =                  -5.) Lower aperture limit relative to center
(upper  =                   5.) Upper aperture limit relative to center

                                -- AUTOMATIC APERTURE RESIZING PARAMETERS --
(ylevel =                 0.05) Fraction of peak or intensity for resizing
(peak   =                  yes) Is ylevel a fraction of the peak?
(bkg    =                  yes) Subtract background for resizing?
(avglimi=                   no) Average limits over all apertures?

                                -- TRACE PARAMETERS --
(t_step =                   10) Tracing step
(t_funct=              spline3) Trace fitting function
(t_order=                    3) Trace fitting function order
(t_niter=                    1) Trace rejection iterations
(t_low  =                   3.) Trace lower rejection sigma
(t_high =                   3.) Trace upper rejection sigma

                                -- APERTURE EXTRACTION PARAMETERS --
(weights=             variance) Extraction weights (none|variance)
(pfit   =                fit1d) Profile fitting algorithm (fit1d|fit2d)
(lsigma =                   4.) Lower rejection threshold
(usigma =                   4.) Upper rejection threshold

                                -- BACKGROUND SUBTRACTION PARAMETERS --
(backgro=                  fit) Background to subtract
(b_funct=             legendre) Background function
(b_order=                    2) Background function order
(b_sampl=        -20:-10,10:20) Background sample regions
(b_naver=                 -100) Background average or median
(b_niter=                    0) Background rejection iterations
(b_low  =                   3.) Background lower rejection sigma
(b_high =                   3.) Background upper rejection sigma
```

Figure 28: The first half of the parameters of **sparams**. These are identical to those of **apall**.

```
                                   -- ARC DISPERSION FUNCTION PARAMETERS --
(coordli=linelists$ctiohear.dat) Line list
(match  =                   10.) Line list matching limit in Angstroms
(fwidth =                    8.) Arc line widths in pixels
(cradius=                   10.) Centering radius in pixels
(i_funct=               spline3) Coordinate function
(i_order=                     3) Order of dispersion function
(i_niter=                     1) Rejection iterations
(i_low  =                    3.) Lower rejection sigma
(i_high =                    3.) Upper rejection sigma
(refit  =                   yes) Refit coordinate function when reidentifying?
(addfeat=                    no) Add features when reidentifying?

                                   -- AUTOMATIC ARC ASSIGNMENT PARAMETERS --
(select =                interp) Selection method for reference spectra
(sort   =                    jd) Sort key
(group  =                   ljd) Group key
(time   =                    no) Is sort key a time?
(timewra=                   17.) Time wrap point for time sorting

                                   -- DISPERSION CORRECTION PARAMETERS --
(lineari=                   yes) Linearize (interpolate) spectra?
(log    =                    no) Logarithmic wavelength scale?
(flux   =                   yes) Conserve flux?

                                   -- SENSITIVITY CALIBRATION PARAMETERS --
(s_funct=               spline3) Fitting function
(s_order=                     9) Order of sensitivity function
(fnu    =                    no) Create spectra having units of FNU?
```

Figure 29: The rest of the parameters of **sparams**. These now go beyond the **apall** parameters to include the details of wavelength calibration and flux calibration.

```
obj058 comp030
obj068 comp030 comp039
obj090 comp039
obj129
```

Figure 30: A reference table *reftable* suitable for **doslit**. The comparison spectrum *comp030* will be used for the program spectrum *obj058*, the comparison exposures *comp030* and *comp039* will be used for the program exposure *obj068* (weighted following whatever rules are selected by the **select** parameter), and the comparison exposure *comp039* will be used for object exposures *obj090* and *obj129*.

object exposures and comparison exposures are used. If you do choose to use a reference table for the comparison exposures, you need only set the **doslit** parameter **arctabl** equal to the name of the reference table (see Fig. 27).

We are ready to proceed! Simply run

doslit *.imh arcs=*.imh

or

doslit *.imh arcs=*.imh arctable="reftable"

You will find yourself in the "aperture editor", described in Sec. 3.3.1. You can change the size of the aperture or otherwise modify it, and you can type a **b** to examine and modify the background windows as described in Sec. 3.3.1. When you **q**uit out of these you will go on to the trace (Sec. 3.3.1). Note that if you answer the various questions asked ("Edit apertures for *imagename*?", "Fit traced positions for *imagename* interactively?") with "YES" rather than "yes", it will quit asking you and simply assume the answer is always going to be yes. Similarly, if you find you don't need to interactively examine the apertures or traces, you can answer "NO" to either or both of these questions, and you will not be asked again.

After all the images have had their apertures defined and traced, a dummy extraction will be made on the first comparison exposure in the **arcs** list. You will find yourself in the **identify** task, just as we described in Sec. 3.4.1. Mark three or four lines, do a **f**it, return to the identify portion with a **q**, identify more lines with an **l**, do a new **f**, and adjust the function and order until you have a good fit and no bad points. Exit with a **q** or two.

The next choice you will face is whether or not to change to change the dispersion constants; whatever you choose here will be applied to all the spectra.

Once you've settled this, things are pretty much automatic. The first program spectrum will be extracted, and if wavelength calibration has been selected (**dispcor=yes** in Fig. 27) the appropriate comparison arc(s) will be extracted with the same aperture and trace and **reidentify** run on the extraction. You will be told how **reidentify** did (the number of lines found and fit, the average shift of the spectrum, and the RMS of the fit), and given a choice of fitting the dispersion function interactively. (Again, "no" means "not this time", while "NO" means, "quit asking me".) You will be given a chance to see the extracted, wavelength-calibrated spectrum plotted if you chose **splot=yes** in the parameters for **doslit** (Fig. 27); if you answer this with "YES" you will of course see every spectrum without having to deal with the question again. The process will continue for the remaining spectra.

The **doslit** routine does not leave intermediate products lying around—it is the extracted spectra themselves that were dispersion corrected, and the final files are automatically in "multispec" format.

Note that in running **doslit** we simply specified "all images" for both the image names and for the arcs. The **doslit** task uses the imagetype parameter **IMAGETYP**, if present in the headers, to decide what is an object exposure (**IMAGETYP=object** or **IMAGETYP=OBJECT**, and what is a comparison arc (**IMAGETYP=comp** or **IMAGETYP=COMPARISON**). The values for these key-words are consistent with the NOAO Kitt Peak and CTIO data-taking systems (both ICE and the FORTH codes).

We could also have done the flux calibration using **doslit**—we would merely have to have created a file containing the names of the files containing standard stars, and specified this as the **standards** parameter (Fig. 27). For instance,

files obj058,obj090,obj129 > stands

followed by

doslit *.imh arcs=*.imh standards=@stands fluxcal+

would not only have extracted and wavelength calibrated our spectra, but flux calibrated them as well. In point of fact it is unnecessary to redo anything—if we have already done the extractions and wavelength calibrations we will automatically do only the steps that haven't been done, unless **redo=yes** (Fig. 27).

One last mode of **doslit** should be mentioned: it is possible to turn **quicklook=yes**. This provides an absolute minimum of interaction: this should work fine if the spectrum is the strongest peak in a spatial cut, and if the size of the aperture, the location of the background, and the trace parameters have all been reasonably set (probably from a previous run of **doslit**). In fact, if we were to set the parameters like those of Fig. 27, but turn on flux calibration as well

doslit *newspect* **arcs=*.imh standars=@stands fluxcal+**

we would achieve the goal stated on the first page of this manual: it is possible to obtain a plot of the wavelength-calibrated, flux-calibrated spectrum with a minimum of fuss—and a careful combining of this with **ccdproc** would result in seeing such a plot before the telescope finishes its slew to the next object! Have fun!!!

A Dealing with Multiple Stars On the Slit

There is very little extra that needs to be said for dealing with the "trickiness" of having multiple objects on the slit. When one is in the aperture editor, one merely has to mark with **m** the additonal stars that one wants to reduce. Typing a **b** with the cursor positioned to any of these marked objects will then allow you to interactively examine and modify the background for that particular object—and you may well need to do this for each object if the region is complicated. If you want to make a global change to some parameter in the aperture editor—such as the **upper** limit, you may find it convenient to activate the **all** switch by typing a **a**. Then, if you execute any of the normal commands (such as **:upper 3**) this will affect *all* your apertures. This will stay in effect until you turn it off with a second **a**.

In Fig. 31 we show this procedure for the spectra of three SMC stars. We could have run either **apall** or **doslit**, and it would have automatically found the brighter star. We have then positioned the cursor on each of the other two stars and typed an **m**. Note that the apertures are numbered in the order they were found. Next we position the cursor on the far-left aperture (numbered "2") and strike a **b** to examine the background. The middle panel shows that the default background is not suitable—another star (the star being extracted in aperture 3, as it happens) is in one of the two background windows. We type an **t** to clear the background regions, and mark new regions using an **s**. The new regions, and fit (done by an **f**) are shown in the bottom panel.

If we have chosen **multispec** format and **extras+** the final image in this case will have dimensions (1198, 3, 4): there are 1198 points along the dispersion axis, there are 3 stars (called "apertures"), and there are 4 spectra associated with each of these objects: the variance-weighted, cleaned spectrum, the no-weights extracted spectrum, the sky spectrum, and the sigma of the variance-weighted extractions. These four "extras" are called "bands".

All of the spectral routines can handle these multi-dimensional images just fine—if you use **apall** to extract the comparison spectrum using the stellar exposure as a reference, you will wind up with three extractions of the comparisons, and **identify** and **reidentify** will keep track of what's what—and things are completely transparent if you are using **doslit**. The plotting routine **splot** (see Sec. C) handles these data just fine.

One book-keeping device that you may want to make use of is to assign separate names to each individual object you've extracted. In complicated cases you can do this by creating a table that consists of an aperture number, a "beam number" (always equal to the aperture number the way we have described defining apertures), and an aperture-specific title. The table might resemble that of Fig. 32. One could then specify the name of this table in **apall** as the parameter **apid**. Alternatively, one could accomplish the same goal by doing

hedit obj029.ms.imh APID1 "SMC h53-148" add+ ver- show+
hedit obj029.ms.imh APID2 "SMC h53-177" add+ ver- show+
hedit obj029.ms.imh APID3 "SMC h53-171" add+ ver- show+

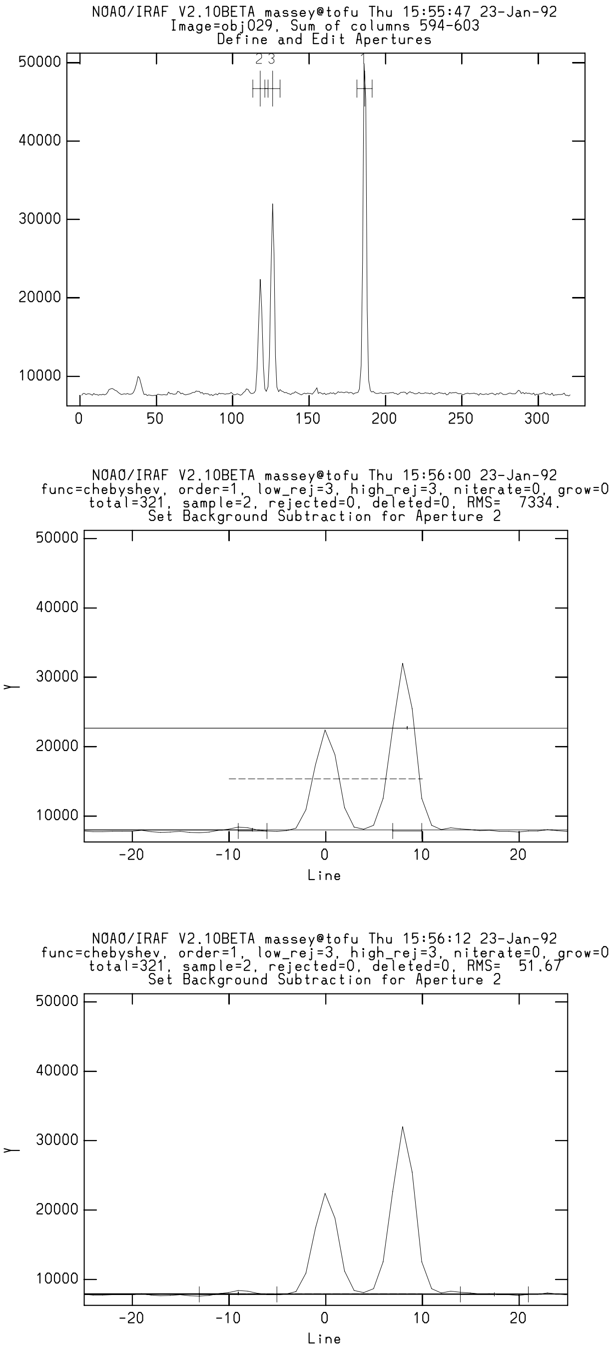

Figure 31: Three stars are marked for extraction. Checking the background of star 2 (the one left of center) shows that the background region is contaminated by a neighboring star. The bottom panel shows the newly defined background region and fit.

```
1 1 SMC h53-148
2 2 SMC h53-177
3 3 SMC h53-171
```

Figure 32: A table relating aperture number and star name.

B The "Long-Slit Case", or When Distortions are Really Bad

The "long-slit case" means different things depending upon your application, but the one problem common to all of them is the need to geometrically correct your data so you can then go on and do your extractions and calibrations. In Sec. 2 we discussed the need for the spatial axis to be "exactly" along a line or column over the region used for sky subtraction; if your background region extends over a considerable length of the slit, this demand becomes more stringent. In addition, when you are extracting an object that covers a substantial length of the slit, you will degrade the spectral resolution if there are misalignments or optical distortions that result in the slit not being quite parallel to the array. In these cases you may need to remove the geometrical distortions (be they tilts or curves) in the spatial axis. You can do this quite readily using only a comparison arc as the calibration image. We explain the procedure for doing all this in this section.

Furthermore, if you have data with is intrinsically untraceable, such as, say, emission line objects, you may wish to make the dispersion axis really be parallel to the chip's axis at *all* positions along the slit. The best calibration data for this is a "multi-hole" image—the spectrum made by placing a mask with a series of holes parallel to the slit. One could in fact accomplish this same thing, albeit not as well, with one or more exposures of a star at different positions along the slit.

The procedure for correcting your data for geometrical distortions is straightforward:

1. Identify the comparison spectrum along some dispersion line using **identify**.

2. Reidentify the features at other dispersion lines using **reidentify**.

3. Perform a fit of a two-dimensional function to wavelength as a function of column and line number using the task **fitcoords**. This fit then provides the transformation of wavelength as a function of (x,y) on the image.

4. Repeat steps (1)-(3) for the multi-hole spectrum and/or multiple-star exposure. This then provides the transformation of slit position as a function of (x,y) on the image.

5. Using the **transform** task, perform the geometrical correction. The two transformations are inverted to map the wavelength as a linear function along one axis, and position along the slit as a linear function along the other axis. If there is only one transformation equation, **transform** will simply do a one-to-one mapping for the other coordinate.

The **identify**, **reidentify**, **fitcoords**, and **transform** tasks are all located within the **twodspec.longslit** package. Load **noao**, **twodspec**, and **longslit**.

We begin with the comparison spectrum shown back in Fig. 4. The shift from one side of the end of the slit to the other was only a few pixels, but the data will serve as a reasonable enough example. The parameters for **identify** are identical to those shown in Fig. 12. (Note that if your dispersion axis runs along a column rather than a row, you should set **section=middle col** rather than **section=middle line**.) Simply run **identify** on the comparison exposure you plan to use for correcting the distortions, i.e., **identify comp030** We follow the procedure outline in Sec. 3.4.1, i.e.,

1. Mark **m** three or four lines, giving the approximate wavelength of each line.

2. Do a fit (**f**).

3. Do a **q** to return to the identification part of the task. Use the preliminary fit to automatically identify other features using the wavelength list by typing an **l**.

4. Do a new fit (**f**), and modify the fitting order to whatever is needed to make a good fit. You can examine the residuals by typing a **j**, and you can examine the non-linear portion of the fit by typing a **l**. You can change the order of the fit by doing a **:order 3**, for example, or change the fitting function by typing a **:function chebyshev**, for example.

5. When you are happy with the fit, do a **q** or two, and answer "yes" to the question "Write to database?"

Next we run **reidentify**. The parameters are shown in Fig. 14. (Again, remember that if you had to specify **section=middle col** in **identify** you must do that here as well.) This task will look for the same comparison line as found in **identify** every 10 lines along the spatial axis.

reidentify comp030 comp030 verbose+

Since the **verbose** switch on, we will see output like that shown in Fig. 33.

We now have a dispersion solution every 10 rows throughout the data. We can use **fitcoords** to find a surface that defines wavelength as a function of x, y position in the image. The parameters for this run of **fitcoords** is given in Fig. 34.

```
REIDENTIFY: NOAO/IRAF V2.10BETA massey@tofu Mon 16:49:19 20-Jan-92
  Reference image = comp030, New image = comp030, Refit = yes
        Image Data    Found      Fit Pix Shift  User Shift  Z Shift      RMS
comp030[*,151]        26/26   26/26    -0.0785     -0.0747  -1.7E-5   0.0458
comp030[*,141]        26/26   26/26     -0.088     -0.0821  -1.8E-5   0.0489
comp030[*,131]        26/26   26/26    -0.0772     -0.0721  -1.6E-5   0.0464
comp030[*,121]        26/26   26/26    -0.0878      -0.082  -1.8E-5   0.0468
comp030[*,111]        26/26   26/26    -0.0799     -0.0746  -1.7E-5   0.0541
comp030[*,101]        26/26   26/26    -0.0815     -0.0761  -1.7E-5   0.0506
comp030[*,91]         26/26   26/26    -0.0801     -0.0748  -1.7E-5   0.0529
comp030[*,81]         26/26   26/26    -0.0823      -0.077  -1.7E-5   0.0561
comp030[*,71]         26/26   26/26    -0.0348     -0.0324  -7.3E-6     0.12
comp030[*,61]         26/26   26/26    -0.0935     -0.0877  -1.9E-5   0.0526
comp030[*,51]         26/26   26/26    -0.0537     -0.0502  -1.2E-5   0.0487
comp030[*,41]         26/26   26/26    -0.0666     -0.0626  -1.4E-5    0.144
comp030[*,31]         26/26   26/26    -0.0106    -0.00976  -2.4E-6   0.0565
comp030[*,21]         26/26   26/26    -0.0297     -0.0278  -6.3E-6   0.0489
comp030[*,11]         26/26   26/26    -0.0202      -0.019  -4.8E-6   0.0625
comp030[*,1]          26/26   26/26    -0.0086    -0.00808  -1.9E-6   0.0603
comp030[*,171]        26/26   26/26     0.0786       0.072  1.60E-5   0.0471
comp030[*,181]        26/26   26/26     0.0844      0.0787  1.75E-5   0.0487
comp030[*,191]        26/26   26/26     0.0732      0.0682  1.50E-5   0.0544
comp030[*,201]        26/26   26/26     0.0865      0.0806  1.80E-5    0.063
comp030[*,211]        26/26   26/26     0.0849      0.0792  1.75E-5   0.0657
comp030[*,221]        26/26   26/26     0.0863      0.0805  1.78E-5   0.0791
comp030[*,231]        26/26   26/26     0.0892      0.0832  1.83E-5   0.0834
comp030[*,241]        26/26   26/26      0.111       0.103  2.29E-5   0.0912
comp030[*,251]        26/26   26/26      0.107      0.0994  2.19E-5   0.0981
comp030[*,261]        26/26   26/26      0.126       0.117  2.61E-5    0.113
comp030[*,271]        26/26   26/26      0.118        0.11  2.42E-5    0.117
comp030[*,281]        26/26   26/26      0.131       0.122  2.69E-5    0.118
comp030[*,291]        26/26   26/26      0.157       0.146  3.23E-5    0.127
comp030[*,301]        26/26   26/26      0.193       0.179  4.00E-5     0.12
comp030[*,311]        26/26   26/26      0.163       0.152  3.32E-5    0.125
comp030[*,321]        26/26   26/26       0.19       0.177  3.94E-5     0.13
```

Figure 33: The output from **reidentify**.

```
PACKAGE = longslit
   TASK = fitcoords

images  =              comp030  Images whose coordinates are to be fit
(fitname=                    ) Name for coordinate fit in the database
(interac=                 yes) Fit coordinates interactively?
(combine=                  no) Combine input coordinates for a single fit?
(databas=            database) Database
(deletio=        deletions.db) Deletion list file (not used if null)
(functio=           chebyshev) Type of fitting function
(xorder =                   6) X order of fitting function
(yorder =                   6) Y order of fitting function
(logfile=      STDOUT,logfile) Log files
(plotfil=            plotfile) Plot log file
(graphic=           stdgraph) Graphics output device
(cursor =                    ) Graphics cursor input
(mode   =                  ql)
```

Figure 34: The parameter file for the running the dispersion data through **fitcoords**.

In running **fitcoords**[3] we are faced with the usual problem of trying to display 3 dimensions of information (x, y and residuals) on a two-dimensional screen. Imagine a piece of paper representing your surface fit; the residuals would appear above and below the paper. Now you may view the paper from above, in which case you see (x,y) and the residuals are projected onto the plane of the paper, or you view the paper along one edge so that the residuals appear compressed in one coordinate (x,r) or (y,r). You may select orientations with the cursor command **x** and **y**; these will respond with a question asking which axis of the data you wish to show in x and y respectively. For example, x=x and y=y followed by a **r** (redraw) will show the location of the data points but no residuals (equivalent to viewing the paper from above); x=y and y=r (followed by a **r** for replot) will show the residuals as a function of y but all information in x is lost in the projection (equivalent to viewing the paper on its side).

Start by showing x vs y: type a **x** and specify the x-axis; type a **y** and specify the y-axis; type a **r** to redraw. You will now see a map of the comparison lines. Delete the points along the top edge of the spatial axis, and delete the points along the bottom edge of the spatial axis. For the data shown here, where the spatial axis is parallel to the y axis (more or less) we can do this by positioning the cursor to a point along the top row and hitting a **d** followed by a **y**. This is also a good time to delete any very deviant points (**d** followed by a **p**, for "point"). Do a **f** for a new fit. The top plot of Fig. 35 shows this perspective (*x vs y*) with the points on the spatial ends deleted.

Let's see what the residuals look like: first let's look at the residuals as a function of "x", with all the "y" information compressed: do a **y** followed by a **r** (for residual) followed by a **r** (for redraw). Again, delete any very deviate points by positioning the cursor to the point and typing a **d** followed by a **p**. Do another **f** to generate a new fit. We will see a plot like that in the middle of Fig. 35. We can up the order of the fit to the x-axis by doing a **:xorder 7**, say. I suggest you choose an order that is as low as you can without there being obvious systemic residuals. Here we choose to stay with **xorder=6**.

Finally let's examine the residuals as a function of "y" Repeat the steps above, deleting any terrible points, and choosing a low order. Again we chose to stay with **yorder=6** (bottom of Fig. 35).

We exit **fitcoords** with a **q**.

Although this fit above will linearize the data in the wavelength direction, remember that its other use to us is to straighten out the distortions (primarily tilt in this case) in the spatial direction—this can improve your sky subtraction (and improve the spectral resolution) if your object is extended and covers a large fraction of the slit. Where we to now use this fit alone to transform our data, we would expect that we have (nearly) a constant number of Å per pixel in the *x* direction and that the comparison lines would now

[3]Some of this material is explained so well in the Tololo cookbook *2D-Frutti Reductions with IRAF* by Mario Hamuy and Lisa Wells that it is lifted verbatim from that manual.

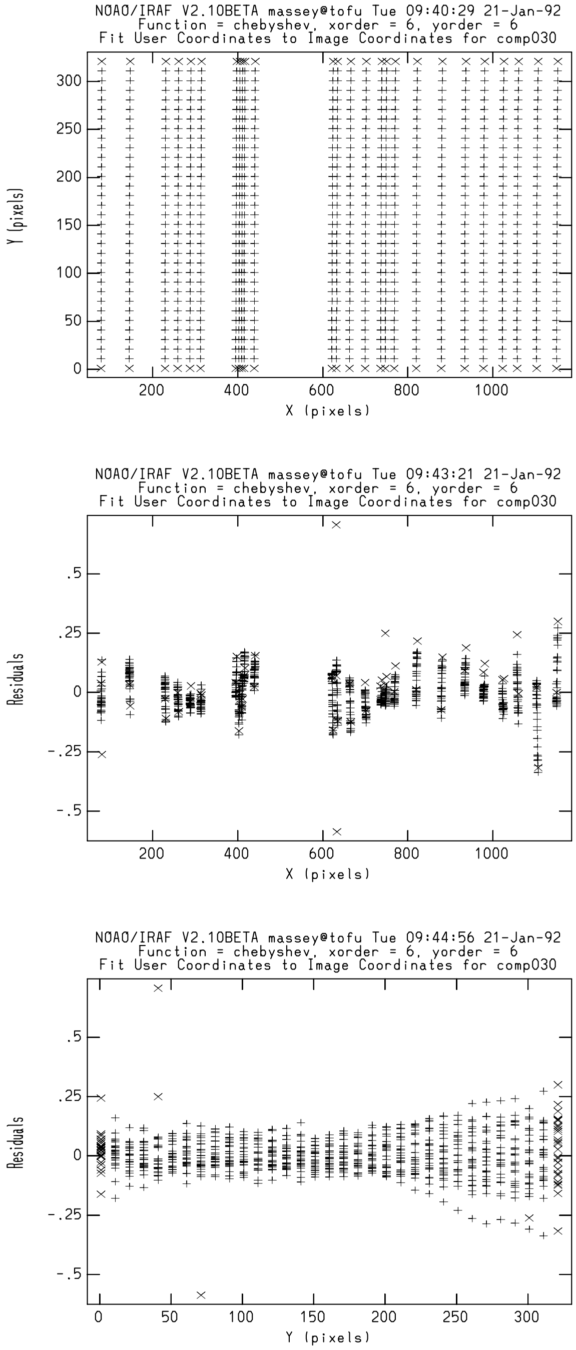

Figure 35: Three views of fitting the comparison line data with **fitcoords**.

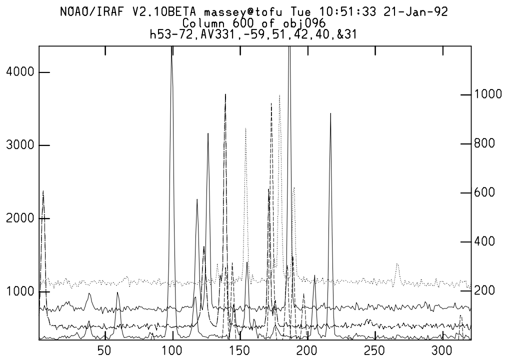

Figure 36: An overplot of the six images, each containing a few stars at different positions along the slit.

be exactly parallel to the y axis. However, a stellar spectrum would still be quite tilted (i.e., as in Fig. 2), and we would not have linearized the spatial axis—we would not have a constant number of arcseconds per pixel along the spatial axis. The tilt of this spectrum would normally be taken out when we trace our specta, but tracing will only work if there is continuum throughout the length of the spectrum. The alternative is to either use a stellar exposure as a "reference" spectrum (i.e., adopt its trace) or to transform the data to remove this tilt. Since the trace will be a function of position along the slit, your reference spectrum will have to be in the same place as your no-continuum object in order for it to work—although how strongly the trace varies along the slit depends upon how much distortion the camera optics introduce. By transforming the data we can in fact interpolate between points on the slit.

To obtain a fit using multi-hole data, you can do exactly the same procedure as above, except **coordlist** in **identify** must be replaced with the map of the physical location of the holes in the slitlet mask. However, if you don't really need this map if you don't insist on there being a constant number of arcseconds per pixel in the spatial direction. But what if you don't have multi-hole data? For the data we have been reducing here we don't have a multi-hole exposure—but what we do have is a number of exposures with stars at different places along the slit. We can use these data with *no* coordinate list to at least remove the tilts.

We show in Fig. 36 an overplot of the six images we plan to use—stars are found all along the slit.

We run **identify** on the first of these images as follows:

identify obj023 sec="middle col" coordlist="" function="cheb" order=2

(If the dispersion axis had run along a column, rather than a line, then **sec=middle line** rather than **middle col**.) For every star along the slit, mark the peak with an **m** and simply accept the coordinate value of the marked position. Then type **f** to obtain a linear fit, and exit with a couple of **q**'s. Next run **reidentify** on the same image:

reidentify obj023 obj023 sec="middle col" nlost=INDEF

Repeat **identify** and **reidentify** for each frame that you plan to use.

We now want to find the transformation that will "untilt" these spectra. We run **fitcoords** again, but this time with one major difference:

fitcoords obj023,obj029,obj032,obj061,obj093,obj096 fitname=stars combine+

Turning the combine switch on tells it to simply merge all the data from the six entries in the **database**.

The top plot in Fig. 37 shows the data we have: we see that the traces are pretty much parallel (there really isn't much change of the trace along the slit, making this exercise rather silly), with a few deviant points. We can delete those as before. The middle plot of Fig. 37 demonstrates what we might expect—given that we have simply fit **x** as a linear function of **x** in indentify, we should simply adopt a constant here (**xorder=1**). The bottom plot shows that we are in fact dealing with a simple tilt that doesn't change as we go along the slit: we adopt **yorder=2**

The final step in this procedure is to transform the images to remove the geometrical distortion. The parameters of this task, **transform**, is shown in Fig. 38.

We suggest that you begin by transforming the comparison exposure, and a sample object exposure, and then comparing both the transformed and original frames. (Use **display** and blink the frames.) If you like this, you can go ahead and transform all your images:

files *.imh > original
transform @original t//@original comp030,stars

will create new files with a "t" appended to the names.

When you reduce these frames (using either **apall** or **doslit**) you should neither have to trace nor dispersion correct your data.

C More Than Just A Pretty Plot: *splot*

One of the most powerful tools within IRAF for allowing you to interactively examine your spectra is **splot**. However, more than just a plotting tool, **splot** also provides many useful functions for analysis that are best done interactively. The purpose of this section is not to provide a comprehensive list of these features, but merely to point the user in the correct direction: read the help page for **splot** for complete details **phelp splot**. Examples of what can be done with **splot** include:

- Measure the equivalent width and line flux of a spectral line.
- Determine the best-fit parameters of Guassian fit to multiple blended lines.

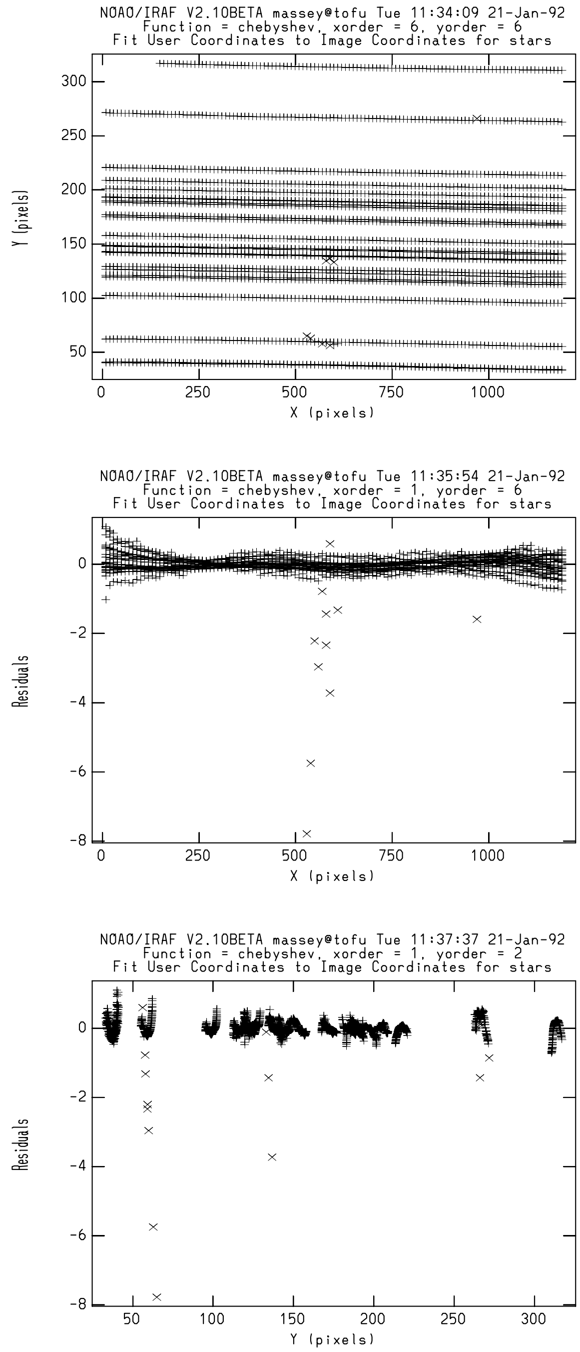

Figure 37: Three perspectives on fitting the spatial data.

```
PACKAGE = longslit
   TASK = transform

input   =       obj096,comp030  Input images
output  =          test1,test2  Output images
fitnames=       comp030,stars  Names of coordinate fits in the database
(databas=            database) Identify database
(interpt=             spline3) Interpolation type
(x1     =               INDEF) Output starting x coordinate
(x2     =               INDEF) Output ending x coordinate
(dx     =               INDEF) Output X pixel interval
(nx     =               INDEF) Number of output x pixels
(xlog   =                  no) Logrithmic x coordinate?
(y1     =               INDEF) Output starting y coordinate
(y2     =               INDEF) Output ending y coordinate
(dy     =               INDEF) Output Y pixel interval
(ny     =               INDEF) Number of output y pixels
(ylog   =                  no) Logrithmic y coordinate?
(flux   =                 yes) Conserve flux per pixel?
(logfile=      STDOUT,logfile) List of log files
(mode   =                  ql)
```

Figure 38: The parameters for the **transform** task.

- Smooth a spectrum.
- Compare two spectra (by overplotting).
- Perform simple arithmetic on the spectra (add, subtract, multiply, divide two spectra, or perform simple functions on a single spectrum).
- Fudge a data point or a region to a particular value.
- Perform simple statistical analysis of a spectrum (RMS, SNR).

D Pre-Extraction Reductions for CCD Data

The details of doing the preliminary reductions of CCD data are given in the *A User's Guide to Reducing CCD Data with IRAF* manual, but for convenience we give an outline here of the steps involved in the preliminaries before one should begin extracting the spectra.

We assume that the calibration exposures you have are:

- **biases (zero-second exposures)** Used to remove pre-flash illumination or any residual structure in the DC offset not removed by the over-scan region.

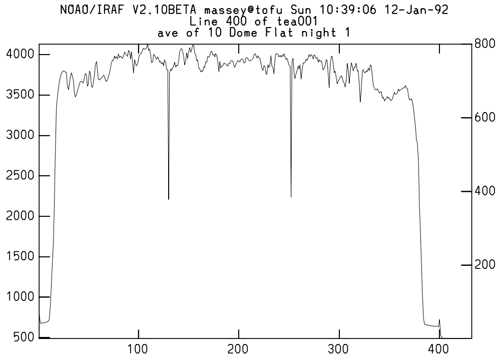

Figure 39: A line cut through this flat-field shows that the region containing good data extends from column 25 through 368. Expanding the region on the right shows that the overscan region is flat from columns 404 through 431. A plot of a column near the middle shows that the first few rows and last few rows are not of good quality, but that good data extends from lines 4 through 795.

- **flat-field exposures** These are used to remove the pixel-to-pixel gain variations and possibly some of the lower-order wavelength-dependent sensitivity variations. Depending upon the instrument, these flat-field exposures may or may not do an adequate job of matching the illumination function along the slit (i.e., in the spatial direction).

- **twilight exposures** These are used to correct any mismatch between the flat-field exposure and the slit illumination function.

The reduction steps are as follows:

1. Examine a flatfield exposure and determine the area of the chip that contains good data. Fig. 39 shows a sample cut through a chip that had been formated to $400 + 32$(overscan) $\times 800$. At the same time determine the columns where the overscan is flat. By expanding the plot and making a plot along a column near the middle, we conclude that the region containing good data is [25:368,4:795], and that the good region of the overscan is [404:431,4:795].

2. load **ccdred** and run **setinstrument specphot**. This will allow you to inspect the parameters the **ccdred** package and the **ccdproc** task. Make sure that only *overscan*, *trim*, and *zerocor* are turned on. Insert the image section containing good data as "trimsec" and the image section containing the good overscan region as "biassec". Insert the name "Zero" for the "zero" entry; this is the average bias frame that will be created in the next step.

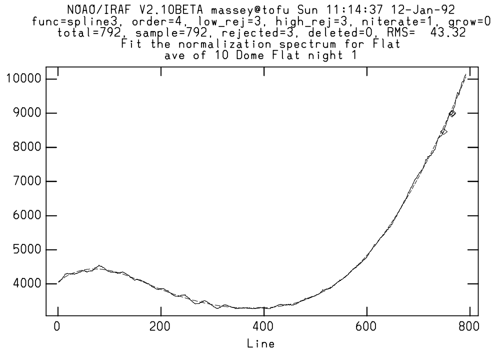

Figure 40: The fit from **response**.

3. Combine the bias frames:
   - **zerocombine *.imh output=Zero**
4. Create a perfect normalized, illumination corrected flat-field exposure.
   (a) Combine the flat-field exposures:
   - **flatcombine *.imh output=Flat combine=average reject=avsigclip ccdtype=flat scale=mode proc+ subsets+**

   (b) Combine the twilight flats:
   - **flatcombine** *sky1,sky2,sky3,sky4* **output=Sky combine=average reject=avsigclip ccdtype="" scale=mode proc+ subsets+**

   (c) Fit a function in the dispersion direction to the combined flat-field using the routine **response** in the **twod.longslit** package:
   - **response Flat Flat nFlat intera+ thresho=INDEF sample=* naver=1 function=spline3 order=1 low_rej=3. high_reject=3. niterat=1. grow=0.**

   Up the order of the fit until you get something that looks good at the ends and more or less fits the shape of the flat. (See Fig. 40.) Alternatively, you may want to keep the order basically to a constant, (function=cheb order=1) if you believe that the wavelength dependence of the flat is mainly due to the instrument and not the lamps and projector screen.

   (d) Process the averaged twilight flat *Sky* through **ccdproc**, this time using the normalized flat-field exposure *nFlat* as the flat:
   - **ccdproc Sky flatcor+ flat="nFlat"**

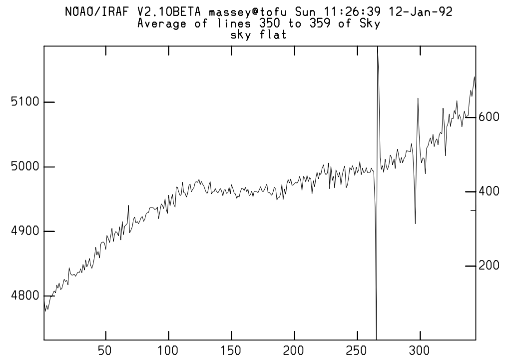

Figure 41: The number of counts in this "flattened" sky changes from 4780 on the left to 5110 on the right; this 7% gradient will lead to problems in sky subtraction if it is not removed.

(e) Decide how well the twilight flats and dome flats agreed: is a plot of the now "flattened" image *Sky* flat in the spatial direction? Fig. 41 shows an example where there is a significant gradient from one side of the slit to the other. At this point it is worth plotting cuts along the spatial axis at section of the chip, to see if these gradients change with wavelength. In the data presented here we found there was a gradual change from the blue to the red.

(f) If the previous step revealed some gradient, determine the correction for the slit function:

- **illum Sky nSky nbins=5 low_reject=3 high_reject=3**

Up the order of the fit until you get something reasonable; see Fig. 42.

(g) Having done this, we can now make the "perfect" flat:

- **imarith nFlat * nSky perfectFlat**

5. Process all of the rest of your data through **ccdproc** using *perfectFlat* for your flat:

- **ccdproc *.imh flatcor+ flat="perfectFlat"**

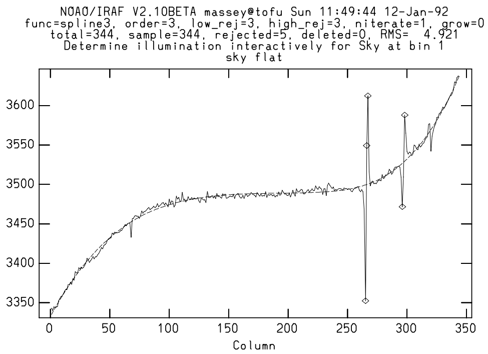

Figure 42: The fit along the spatial axis to the twilight sky exposure that has been "flattened" by the normalized dome-flat.