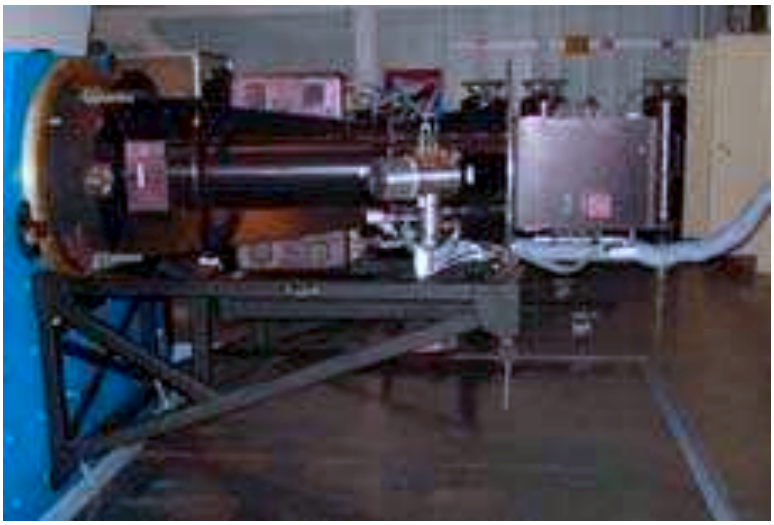

Dual Imaging Spectrograph Data Reduction Cookbook

Karen Kinemuchi
Apache Point Observatory, Sunspot, NM

Version 1.1 - January 2014

DISCLAIMER: I have made this write up to reduce my raw spectra from the DIS spectrograph using IRAF/PyRAF. It is assumed the user of this cookbook is familiar with the IRAF environment and the basic steps of data reduction for CCD images. If not, please refer to the excellent manuals provided by the folks at IRAF[1]. My nomenclature for many of the input and output files were made to keep things simple and less confusing for me, and my choices may not be the best for you.

Please visually inspect your data throughout the processing to make sure you are getting things that are sensible. If not, step back and check if the previous task did not screw up or introduce something wrong.

To use this cookbook, all IRAF tasks are in ALL CAPS for readability. I also provide the parameters for many of the IRAF tasks for convenience, but the user is encouraged to play around with some of the parameters to optimize the steps for their individual science. Keep in mind, I do variable stars, so I have optimized the parameters for my science.

If you have any questions or suggestions, feel free to contact me: kinemuchi@apo.nmsu.edu

[1]http://iraf.noao.edu/docs/spectra.html – *A User's Guide to CCD Reductions with IRAF* and *A Beginner's Guide to Using IRAF*

To get started, you will need to load the following IRAF packages:

- noao
- imred
- ccdred
- twodspec
- apextract
- onedspec
- astutil

Also have a ds9 or saoimage window up to visually inspect all your data.

1. Run ZEROCOMBINE to make the master bias in both the red and blue. First check to make sure your biases have reasonable numbers by running IMSTAT:

```
imstat bias*b.fits
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
      bias.0001b.fits   2261644     97.86     3.252        85.       761.
      bias.0002b.fits   2261644     97.86      3.17        84.       720.
      bias.0003b.fits   2261644      97.8     3.105        84.       186.
      bias.0004b.fits   2261644     97.81     3.288        84.      1229.
      bias.0005b.fits   2261644     97.81     3.183        82.      1077.
      bias.0006b.fits   2261644      97.8     3.128        82.       576.
      bias.0007b.fits   2261644     97.79     4.249        83.      4065.
      bias.0008b.fits   2261644     97.77     3.214        84.       854.
      bias.0009b.fits   2261644     97.76     5.844        82.      6256.
      bias.0010b.fits   2261644     97.78     6.853        84.      8931.

imstat bias*r.fits
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
      bias.0001r.fits   2261644     115.3     6.135       103.      3114.
      bias.0002r.fits   2261644     115.3     5.373       104.       614.
      bias.0003r.fits   2261644     115.3     5.341       103.       370.
      bias.0004r.fits   2261644     115.3     5.421       103.       800.
      bias.0005r.fits   2261644     115.2     5.649       103.      1832.
```

```
bias.0006r.fits   2261644     115.3     5.469      101.     1212.
bias.0007r.fits   2261644     115.2     5.803      103.     1932.
bias.0008r.fits   2261644     115.2     5.362      103.      398.
bias.0009r.fits   2261644     115.2     5.676      103.     1033.
bias.0010r.fits   2261644     115.2     5.681      102.     2180.
```

If the IMSTAT numbers look okay, then create two @-files:

```
cl> files bias.*b.fits > inbbias
cl> files bias.*r.fits > inrbias
```

Other @-files mentioned below can be made the same way in the IRAF environment. Otherwise, you can use unix *ls* commands to make these files.

Create a master blue and master red bias frame:

```
lpar zerocombine
       input = "@inbbias"      List of zero level images to combine
    (output = "ZeroB")         Output zero level name
   (combine = "average")       Type of combine operation
    (reject = "avsigclip")     Type of rejection
   (ccdtype = "")              CCD image type to combine
   (process = no)              Process images before combining?
    (delete = no)              Delete input images after combining?
   (clobber = no)              Clobber existing output image?
     (scale = "none")          Image scaling
   (statsec = "")              Image section for computing statistics
      (nlow = 0)               minmax: Number of low pixels to reject
     (nhigh = 1)               minmax: Number of high pixels to reject
     (nkeep = 1)               Minimum to keep (pos) or maximum to reject (neg)
     (mclip = yes)             Use median in sigma clipping algorithms?
    (lsigma = 3.)              Lower sigma clipping factor
    (hsigma = 3.)              Upper sigma clipping factor
   (rdnoise = "4.6")           ccdclip: CCD readout noise (electrons)
      (gain = "1.75")          ccdclip: CCD gain (electrons/DN)
    (snoise = "0.")            ccdclip: Sensitivity noise (fraction)
     (pclip = -0.5)            pclip: Percentile clipping parameter
     (blank = 0.)              Value if there are no pixels
      (mode = "ql")
```

The output files are called ZeroB.fits and ZeroR.fits for the blue and red, respectively. You may change the *combine* and *reject* algorithms to *median* and *sigclip* here as well. Check

your master bias frames to make sure they don't introduce an artifact into your final reduced data.

2. Run CCDPROC to do the overscan, trim, and bias subtraction of the obj, HeNeAr or "arc", and flat field frames. Apply a bad pixel mask for the red spectra to get rid of the bad column in that camera. As of January 2014, the following bad pixel mask (saved as a text file called "badpix_DISr.txt") works for the red images:

```
1274  1276  165  1024
```

The input @-file "inbtrim" contains all the arcs, flats, and objects from the blue camera. Make another @-file for the arcs, flats, and objects from the red camera (inrtrim). Run CCDPROC twice - once for the blue and again for the red files.

Be sure to use the correct master bias for the blue and red files.

The trimsec used in the lpar below will trim fairly close to the star's spectrum. If you want to preserve most of the 2-D spectrum, you may want to adjust this so that you get more of the image. Maybe something like trimsec = [1:2048, *].

** CAVEAT EMPTOR! If you bin your data, then the trimsec region used here may not be correct. You will have to determine what is the best region is for your binned data.

In my case, I added a prefix "T" to the file names to create the "outtrim" output file below. You may or may not want to do this, since intermediary step files can start filling up computer space. On the other hand, you won't accidentally destroy the original files (always back up the original images!).

```
lpar ccdproc
       images = "@inbtrim"      List of CCD images to correct
     (output = "@outbtrim")     List of output CCD images
     (ccdtype = "")             CCD image type to correct
   (max_cache = 0)              Maximum image caching memory (in Mbytes)
      (noproc = no)             List processing steps only?\n
      (fixpix = yes)            Fix bad CCD lines and columns?
    (overscan = yes)            Apply overscan strip correction?
        (trim = yes)            Trim the image?
     (zerocor = yes)             Apply zero level correction?
     (darkcor = no)             Apply dark count correction?
     (flatcor = no)             Apply flat field correction?
    (illumcor = no)             Apply illumination correction?
   (fringecor = no)             Apply fringe correction?
     (readcor = no)             Convert zero level image to readout correction?
```

```
    (scancor = no)             Convert flat field image to scan correction?\n
   (readaxis = "line")         Read out axis (column|line)
    (fixfile = "badpix_DISr.txt")   File describing the bad lines and columns
    (biassec = "[2051:2096,2:1027]") Overscan strip image section
    (trimsec = "[1:2048,1:1027]") Trim data section
       (zero = ZeroB)             Zero level calibration image
       (dark = "")             Dark count calibration image
       (flat = "")             Flat field images
      (illum = "")             Illumination correction images
     (fringe = "")             Fringe correction images
 (minreplace = 1.)             Minimum flat field value
   (scantype = "shortscan")    Scan type (shortscan|longscan)
      (nscan = 1)              Number of short scan lines\n
(interactive = no)             Fit overscan interactively?
   (function = "legendre")     Fitting function
      (order = 3)              Number of polynomial terms or spline pieces
     (sample = "*")            Sample points to fit
   (naverage = 1)              Number of sample points to combine
   (niterate = 3)              Number of rejection iterations
 (low_reject = 3.)             Low sigma rejection factor
(high_reject = 3.)             High sigma rejection factor
       (grow = 0.)             Rejection growing radius
       (mode = "ql")
```

3. Run FLATCOMBINE for both blue and red flat fields to create your master blue and red flats.

The input files contain the trimmed and bias-corrected flat field frames. There should be two @-files – one for the master blue and one for the master red flat field. You may want to play with the combine and reject algorithms depending on how many flat fields you were able to obtain during your observing run.

```
lpar flatcombine
       input = "@inrflat"      List of flat field images to combine
      (output = "redFlat")     Output flat field root name
     (combine = "median")      Type of combine operation
      (reject = "avsigclip")   Type of rejection
     (ccdtype = "")            CCD image type to combine
     (process = no)            Process images before combining?
     (subsets = no)            Combine images by subset parameter?
```

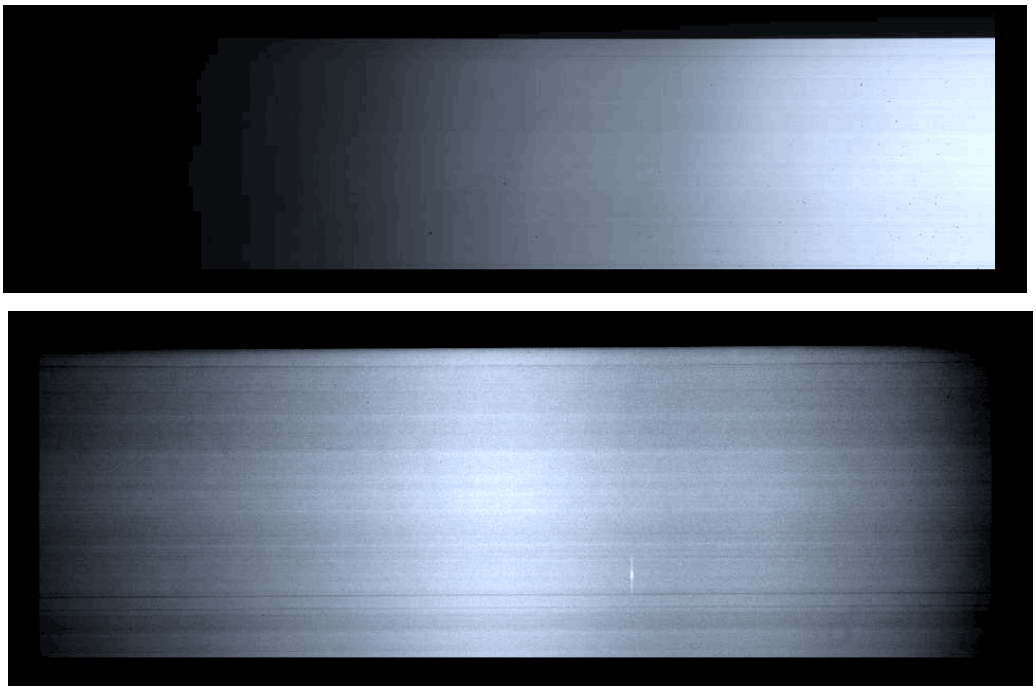

Figure 1: Upper: Master blue flat field. Lower: Master red flat field.

```
 (delete = no)             Delete input images after combining?
(clobber = no)             Clobber existing output image?
  (scale = "mode")         Image scaling
(statsec = "")             Image section for computing statistics
   (nlow = 1)              minmax: Number of low pixels to reject
  (nhigh = 1)              minmax: Number of high pixels to reject
  (nkeep = 1)              Minimum to keep (pos) or maximum to reject (neg)
  (mclip = yes)            Use median in sigma clipping algorithms?
 (lsigma = 3.)             Lower sigma clipping factor
 (hsigma = 3.)             Upper sigma clipping factor
(rdnoise = "0.")           ccdclip: CCD readout noise (electrons)
   (gain = "1.")           ccdclip: CCD gain (electrons/DN)
 (snoise = "0.")           ccdclip: Sensitivity noise (fraction)
  (pclip = -0.5)           pclip: Percentile clipping parameter
  (blank = 1.)             Value if there are no pixels
   (mode = "ql")
```

The output files are called redFlat.fits and blueFlat.fits. Figure 1 shows an example of these master flats.

4. Run CCDPROC again to flat field your arcs and object frames. Again run the task twice, once for the arcs/objects from the blue camera and again for the red camera.

```
lpar ccdproc
       images = "@inprocb"      List of CCD images to correct
     (output = "@outprocb")     List of output CCD images
     (ccdtype = "")             CCD image type to correct
   (max_cache = 0)              Maximum image caching memory (in Mbytes)
      (noproc = no)             List processing steps only?\n
      (fixpix = yes)            Fix bad CCD lines and columns?
    (overscan = yes)            Apply overscan strip correction?
        (trim = yes)            Trim the image?
     (zerocor = yes)            Apply zero level correction?
     (darkcor = no)             Apply dark count correction?
     (flatcor = yes)            Apply flat field correction?
    (illumcor = no)             Apply illumination correction?
   (fringecor = no)             Apply fringe correction?
     (readcor = no)             Convert zero level image to readout correction?
     (scancor = no)             Convert flat field image to scan correction?\n
    (readaxis = "line")         Read out axis (column|line)
     (fixfile = "badpix.txt")   File describing the bad lines and columns
     (biassec = "[2051:2096,2:1027]") Overscan strip image section
     (trimsec = "[1:2048,1:1027]") Trim data section
        (zero = "ZeroB")        Zero level calibration image
        (dark = "")             Dark count calibration image
        (flat = "blueFlat")     Flat field images
       (illum = "")             Illumination correction images
      (fringe = "")             Fringe correction images
  (minreplace = 1.)             Minimum flat field value
    (scantype = "shortscan")    Scan type (shortscan|longscan)
       (nscan = 1)              Number of short scan lines\n
 (interactive = no)             Fit overscan interactively?
    (function = "legendre")     Fitting function
       (order = 3)              Number of polynomial terms or spline pieces
      (sample = "*")            Sample points to fit
    (naverage = 1)              Number of sample points to combine
    (niterate = 3)              Number of rejection iterations
  (low_reject = 3.)             Low sigma rejection factor
 (high_reject = 3.)             High sigma rejection factor
        (grow = 0.)             Rejection growing radius
        (mode = "ql")
```

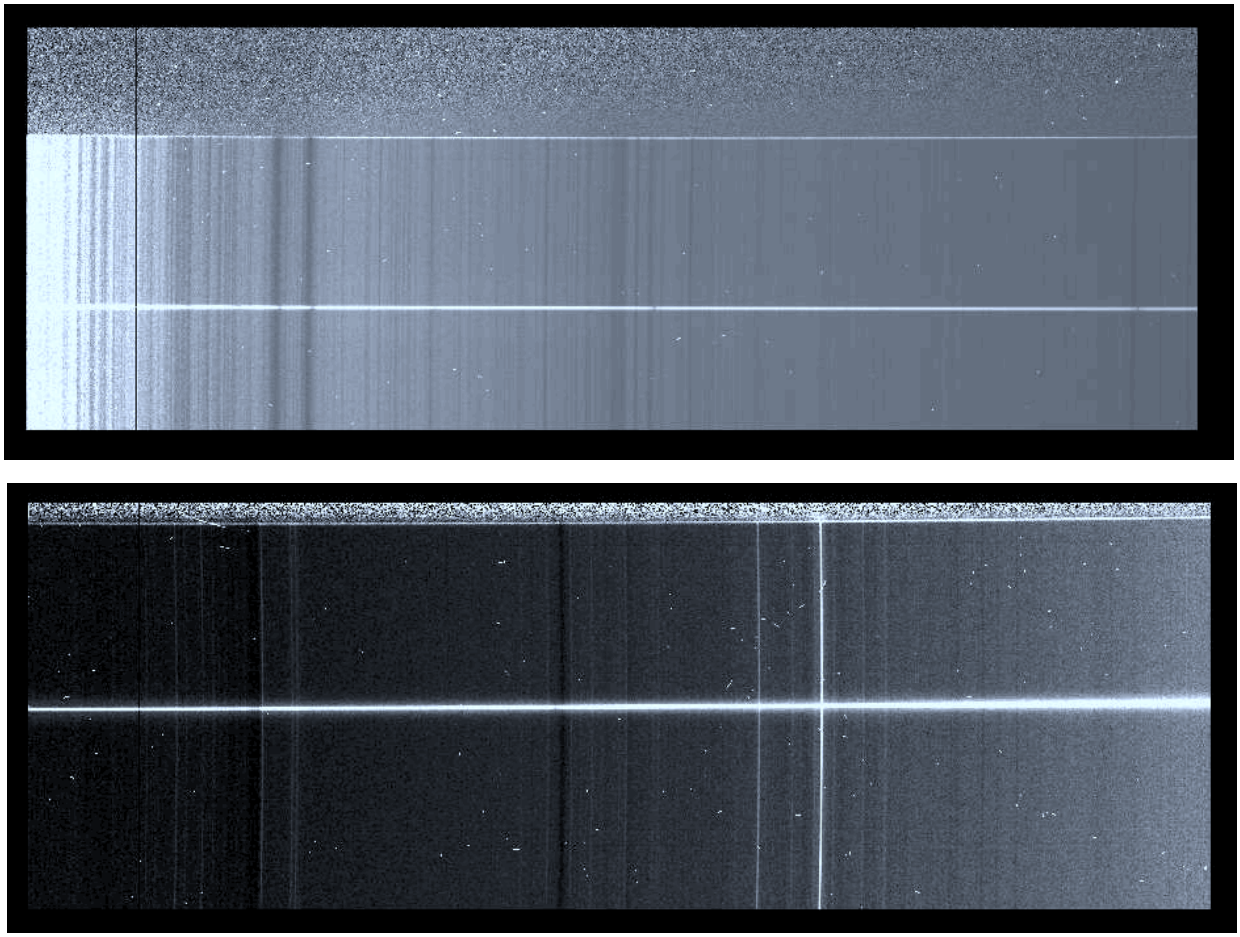

Figure 2: Upper: CCDPROC processed blue spectrum. Lower:CCDPROC processed red spectrum.

Here the output @-file (outprocb) contains the arc and object file names with a prefix of "P" to indicate "processed". You may or may not want to do this as you will be creating more intermediary step files. Examples of the processed blue and red object spectra are presented in Figure 2.

5. Run APALL separately on the red and blue object spectra.

In the example lpar for APALL, the input @-file contains all the red object spectra that have been processed thus far.

A few of the APALL parameters listed below should be adjusted for your data. For ex-

ample, it would be wise to run APALL interactively to get a feel at how well the APTRACE is working on your spectra. To determine the background function, I provide a good starting point, but I would recommend the user to play around with the b_function and b_order parameters. To determine whether you have the appropriate b_sample parameter, use your flat field to determined the appropriate range for the background sampling. For more information about what many of the APALL parameters control, please look to the IRAF manual on longslit spectroscopy[2].

Figure 3 shows the interactive screen of APALL. In this step, we are simply identifying which "spike" is our target star. If one pressed "b", the background level determination can be performed. If you are unhappy with the ranges determined for the background measurements, you can redefined those regions by entering it into the epar of APALL or change them on the fly while running APALL. If you do the latter (while in interactive mode), simply type "s" to the left most region and again on the right most part of the regions you wish to define as your background. Figure 4 shows the APALL interactive screen where one determines where to measure the background levels. To check how well APALL is tracing the aperture (APTRACE), you can interactively change the fitting function, order, or delete outlier points. An example of an APTRACE run is show in Figure 5. More detailed information on how to refine the parameters and to execute them can be found in the IRAF manual for reducing longslit spectra.

The output file below contains the file names with a prefix of "S" to indicate the spectra are now 1-D.

Once the spectra are extracted, check them with SPLOT. Do they look sensible? Examples of extracted blue and red spectra are shown in Figure 6 and 7.

```
lpar apall
       input = "@inrapall"     List of input images
       nfind = 1               Number of apertures to be found automatically
     (output = "@outrapall")   List of output spectra
  (apertures = "")             Apertures
     (format = "onedspec")     Extracted spectra format
 (references = "")             List of aperture reference images
   (profiles = "")             List of aperture profile images
(interactive = yes)            Run task interactively?
       (find = yes)            Find apertures?
   (recenter = yes)            Recenter apertures?
     (resize = yes)            Resize apertures?
       (edit = yes)            Edit apertures?
      (trace = yes)            Trace apertures?
```

[2]http://iraf.noao.edu/docs/spectra.html – *A User's Guide to Reducing Slit Spectra with IRAF*

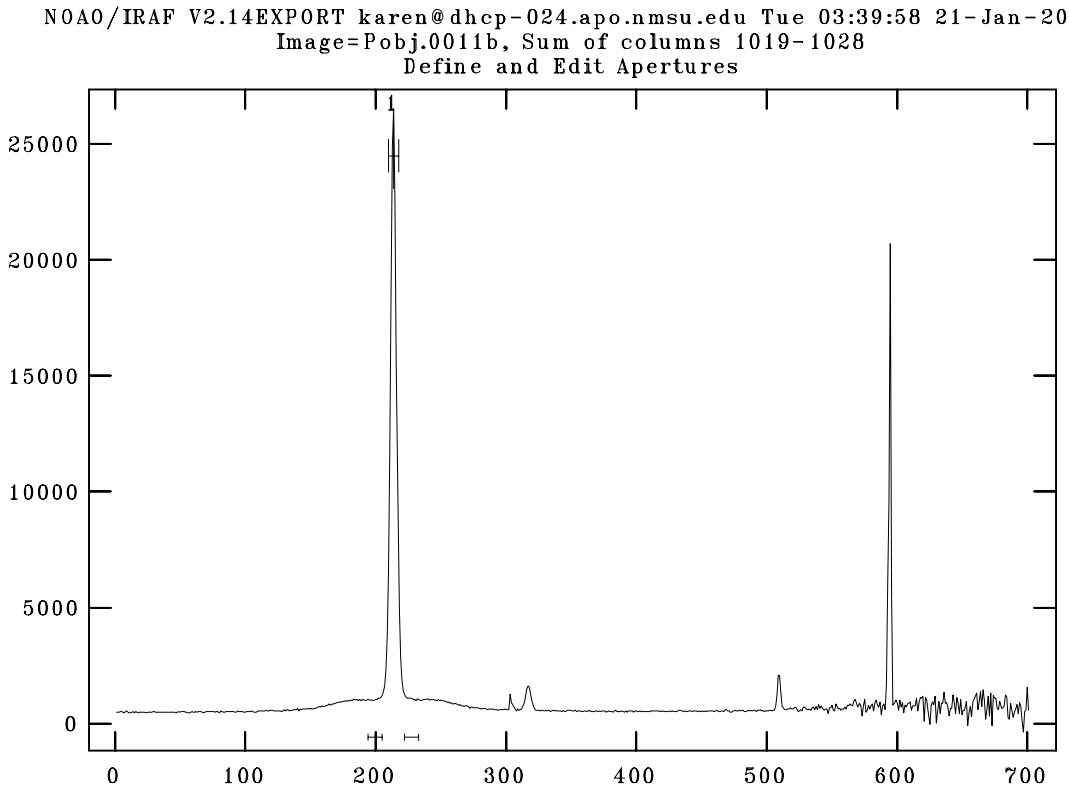

Figure 3: APFIND subtask in APALL.

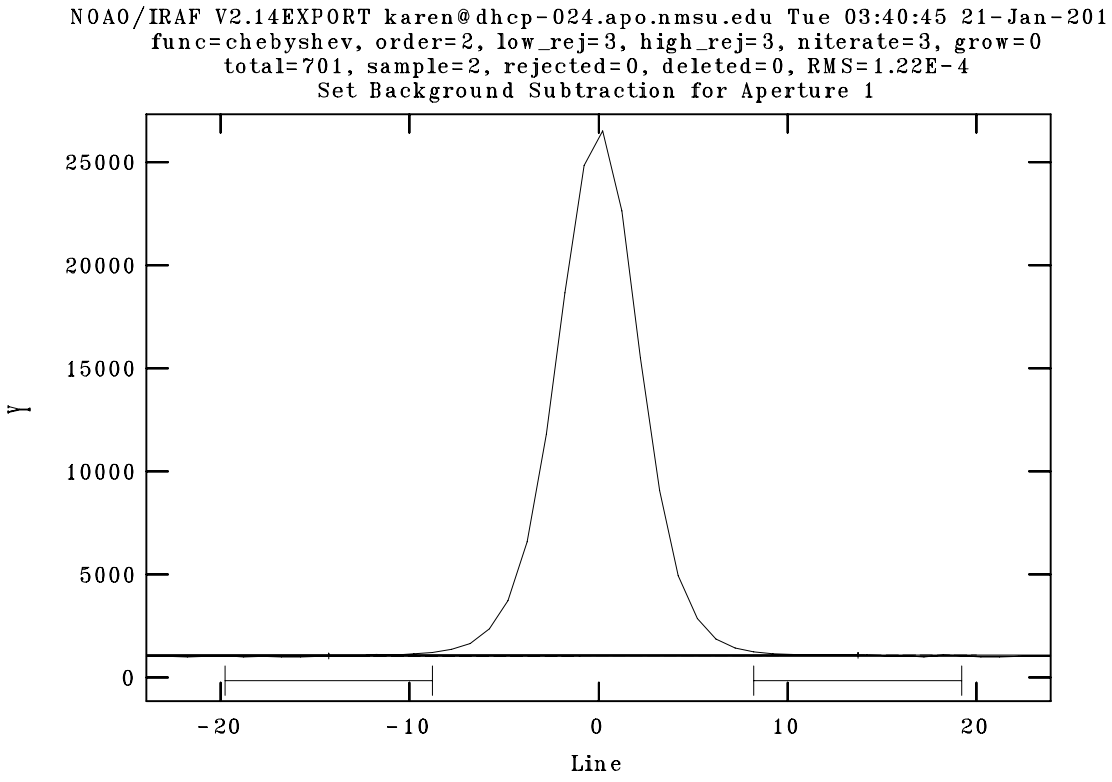

Figure 4: To determine the regions in the spectrum to measure the background levels, type "b" in APALL to get to this screen.

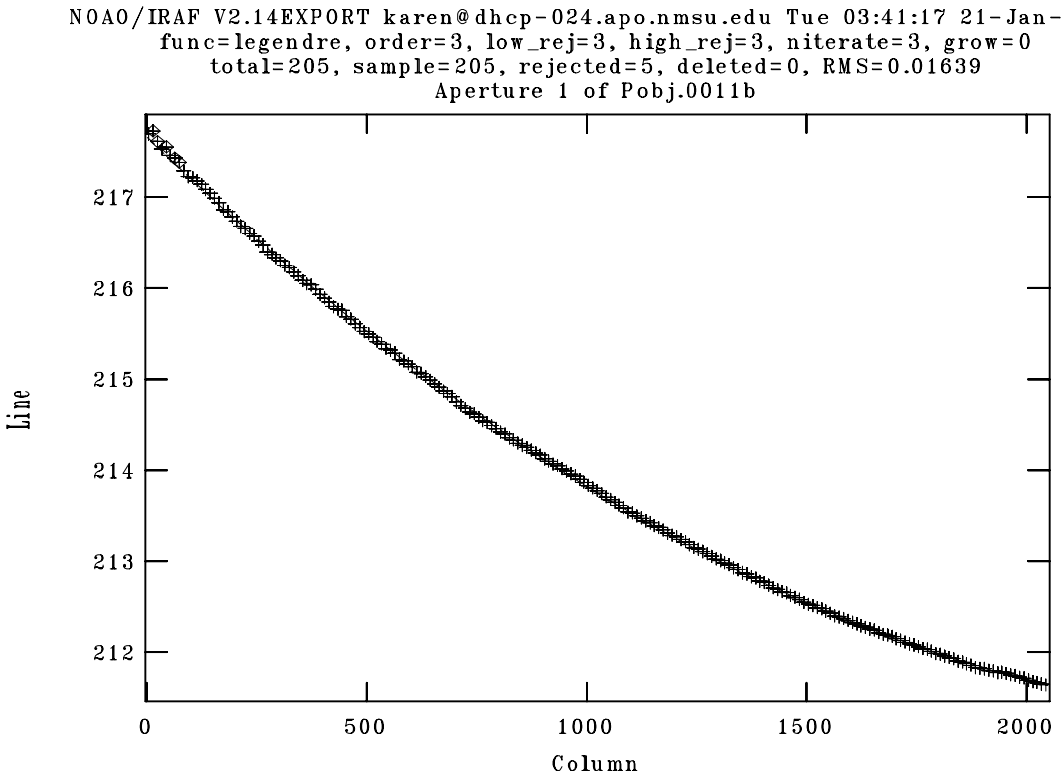

Figure 5: Example of APTRACE subtask in APALL.

```
    (fittrace = yes)            Fit the traced points interactively?
     (extract = yes)            Extract spectra?
      (extras = no)             Extract sky, sigma, etc.?
      (review = no)             Review extractions?
        (line = INDEF)          Dispersion line
        (nsum = 10)             Number of dispersion lines to sum or median
       (lower = -5.)            Lower aperture limit relative to center
       (upper = 5.)             Upper aperture limit relative to center
   (apidtable = "")             Aperture ID table (optional)
  (b_function = "chebyshev")    Background function
     (b_order = 2)              Background function order
    (b_sample = "-20:-8,8:20")  Background sample regions
  (b_naverage = -100)           Background average or median
  (b_niterate = 3)              Background rejection iterations
(b_low_reject = 3.)             Background lower rejection sigma
(b_high_rejec = 3.)             Background upper rejection sigma
      (b_grow = 0.)             Background rejection growing radius
       (width = 10.)            Profile centering width
      (radius = 10.)            Profile centering radius
   (threshold = 0.)             Detection threshold for profile centering
      (minsep = 5.)             Minimum separation between spectra
      (maxsep = 1000.)          Maximum separation between spectra
       (order = "increasing")   Order of apertures
  (aprecenter = "")             Apertures for recentering calculation
      (npeaks = INDEF)          Select brightest peaks
       (shift = no)             Use average shift instead of recentering?
      (llimit = INDEF)          Lower aperture limit relative to center
      (ulimit = INDEF)          Upper aperture limit relative to center
      (ylevel = 0.2)            Fraction of peak or intensity for automatic width
        (peak = yes)            Is ylevel a fraction of the peak?
         (bkg = yes)            Subtract background in automatic width?
      (r_grow = 0.)             Grow limits by this factor
   (avglimits = no)             Average limits over all apertures?
      (t_nsum = 10)             Number of dispersion lines to sum
      (t_step = 10)             Tracing step
     (t_nlost = 5)              Number of consecutive times profile is lost before quitt
  (t_function = "legendre")     Trace fitting function
     (t_order = 3)              Trace fitting function order
    (t_sample = "*")            Trace sample regions
  (t_naverage = 1)              Trace average or median
  (t_niterate = 3)              Trace rejection iterations
```

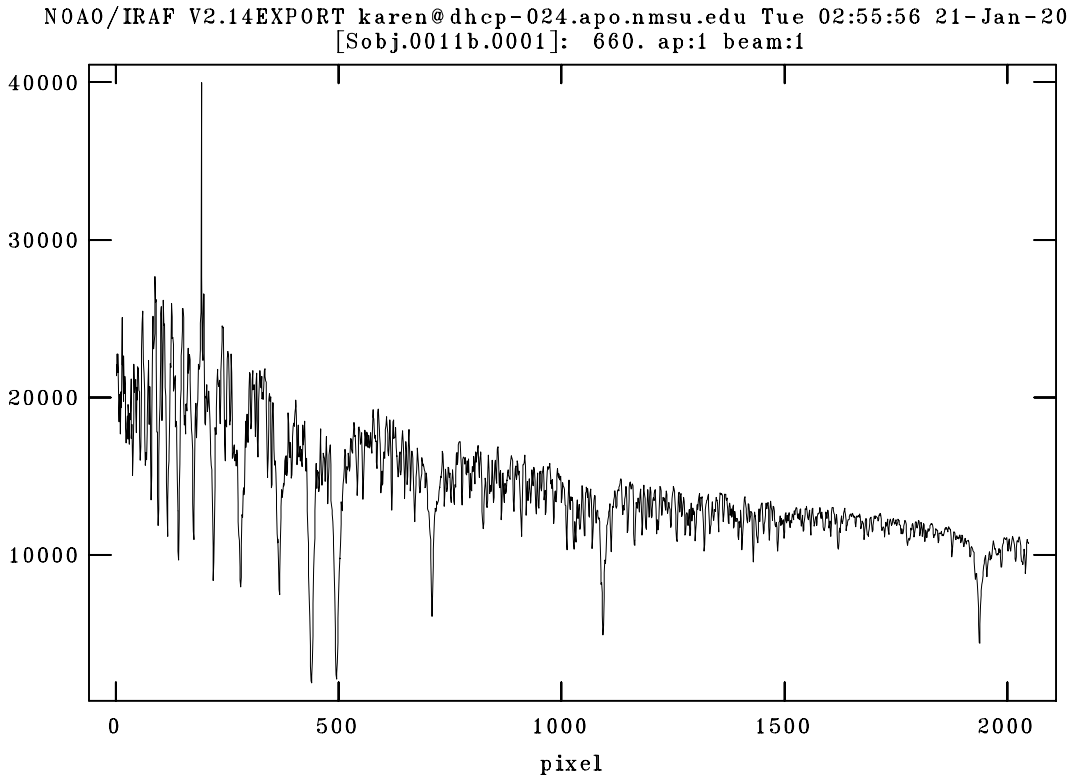

Figure 6: Extracted 1-D blue spectrum from APALL.

```
(t_low_reject = 3.)             Trace lower rejection sigma
(t_high_rejec = 3.)             Trace upper rejection sigma
      (t_grow = 0.)             Trace rejection growing radius
  (background = "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?
  (saturation = INDEF)          Saturation level
   (readnoise = "RDNOISE")      Read out noise sigma (photons)
        (gain = "GAIN")         Photon gain (photons/data number)
      (lsigma = 4.)             Lower rejection threshold
      (usigma = 4.)             Upper rejection threshold
     (nsubaps = 1)              Number of subapertures per aperture
        (mode = "ql")
```

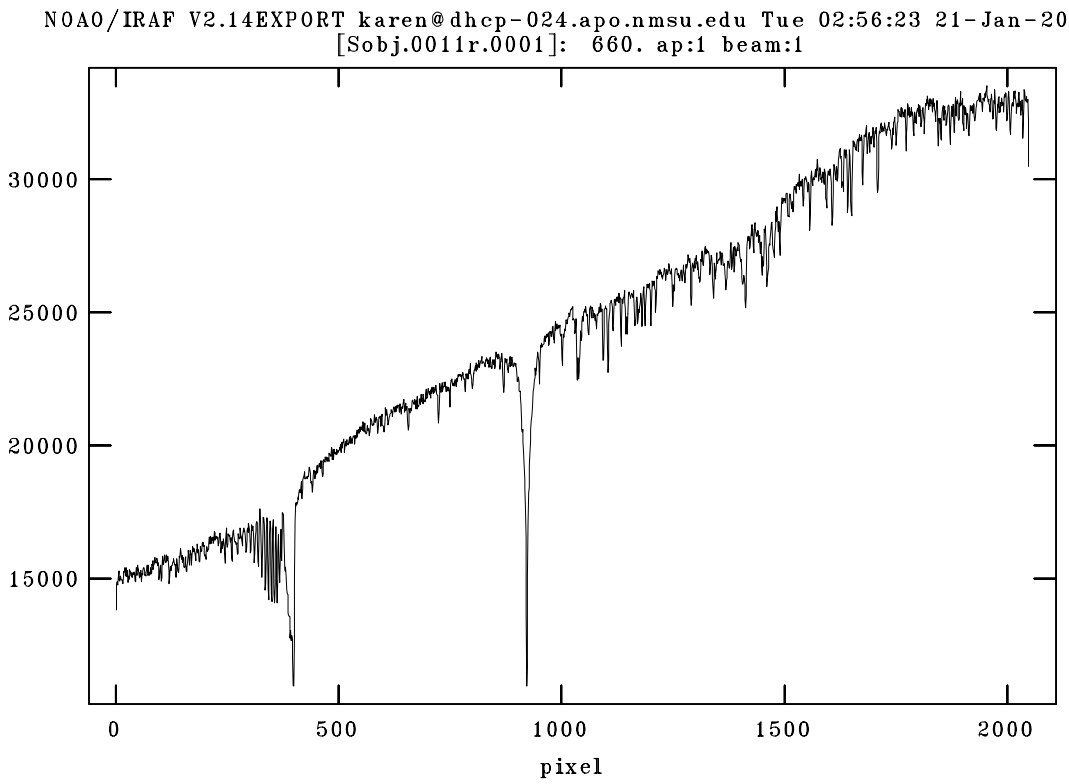

Figure 7: Extracted 1-D red spectrum from APALL. Wavelength increases to the left along the x-axis.

6. Run APALL on the arcs. I used a cl script to do this for each arc in my dataset. I cross reference the arc spectrum with its corresponding object spectrum. An example of my cl script is shown below as an example. This script is saved as a text file ("arcextract.txt").

```
apall Parc.0017b.fits out=comp17b ref=Pobj.0009b rece- trace- back- intera-
apall Parc.0017r.fits  out=comp17r ref=Pobj.0009r  rece- trace- back- intera-
apall Parc.0018b.fits out=comp18b ref=Pobj.0009b rece- trace- back- intera-
apall Parc.0018r.fits out=comp18r ref=Pobj.0009r  rece- trace- back- intera-
apall Parc.0019b.fits out=comp19b ref=Pobj.0010b rece- trace- back- intera-
apall Parc.0019r.fits out=comp19r ref=Pobj.0010r  rece- trace- back- intera-
```

To run a cl script in IRAF, the syntax is as follows:

```
cl> cl < arcextract.txt
```

7. Now that you have 1-D, extracted spectra for your arcs, you need to identify the lines. The HeNeAr reference spectra atlas for the lamps at APO can be found at APO website for the DIS instrument[3].

Select your best arc for both the red and blue images and identify the lines with IDENTIFY. Examples of the HeNeAr arc lamps taken with DIS are shown in Figure 8 and 9.

(Optional advice): If you run the IDENTIFY task interactively, then you can adjust the fitting function and order depending on how good your arcs are (in terms of number of identifiable lines). The functions/orders supplied here are just a suggestion. Check your RMS value (if doing interactively) to see how well you have identified your arc lines.

In my experience, the blue arcs had an RMS of ~ 0.6 and the red arcs had an RMS of 0.01-0.03. These RMS values are simply due to the number of HeNeAr lines available for identification in the blue and red. As seen from the spectra atlas for DIS, there are fewer lines in the blue, and hence the fit can be worse, resulting in a larger RMS value.

(Caveat Emptor!): For the red HeNeAr, the wavelength will increase to the left in your IRAF xterm plot window. You may want to use the window commands of "w" and "f" to flip the x-axis in order to match up against the HeNeAr atlas.

For the red arc example:

```
lpar identify

      images = "comp22r.0001"  Images containing features to be identified
```

[3]http://www.apo.nmsu.edu/arc35m/Instruments/DIS/#4p2

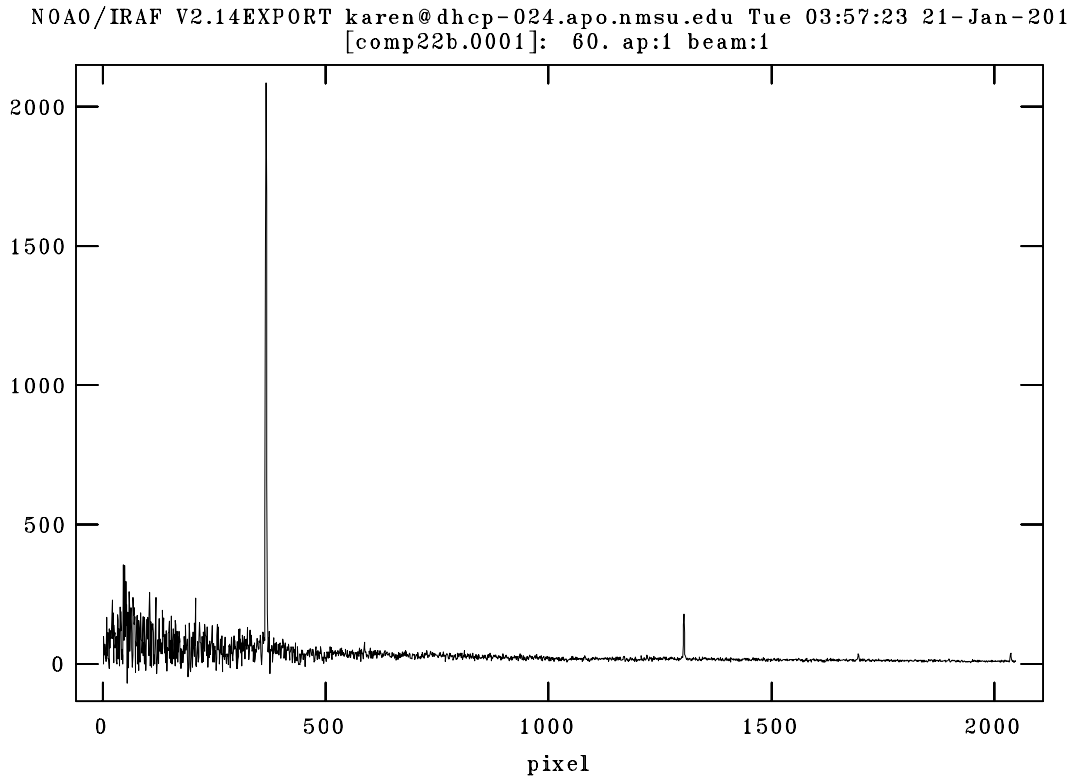

Figure 8: Example of HeNeAr arc spectrum for the blue camera using the B1200 grating. About 4 visible lines can be easily matched to a spectral atlas and identified.

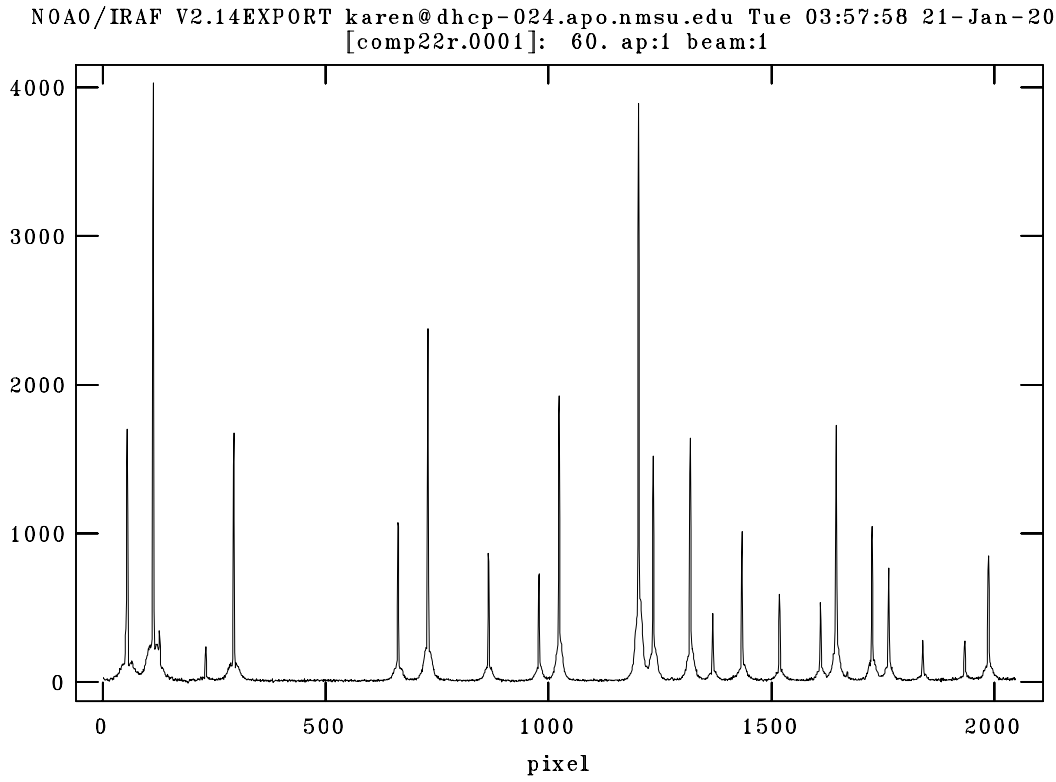

Figure 9: Example of HeNeAr arc spectrum for the red camera using the R1200 grating. Wavelength increases to the left for the red spectrum.

```
        crval =                 Approximate coordinate (at reference pixel)
        cdelt =                 Approximate dispersion
     (section = "middle line")  Section to apply to two dimensional images
    (database = "database")     Database in which to record feature data
   (coordlist = "linelists$idhenear.dat") User coordinate list
        (units = "")            Coordinate units
         (nsum = "10")          Number of lines/columns/bands to sum in 2D images
        (match = -3.)           Coordinate list matching limit
 (maxfeatures = 50)             Maximum number of features for automatic identification
      (zwidth = 100.)           Zoom graph width in user units
       (ftype = "emission")     Feature type
      (fwidth = 4.)             Feature width in pixels
     (cradius = 5.)             Centering radius in pixels
   (threshold = 0.)             Feature threshold for centering
      (minsep = 2.)             Minimum pixel separation
    (function = "spline3")      Coordinate function
       (order = 1)              Order of coordinate function
      (sample = "*")            Coordinate sample regions
    (niterate = 0)              Rejection iterations
  (low_reject = 3.)             Lower rejection sigma
 (high_reject = 3.)             Upper rejection sigma
        (grow = 0.)             Rejection growing radius
   (autowrite = no)             Automatically write to database
    (graphics = "stdgraph")     Graphics output device
      (cursor = "")             Graphics cursor input
     (aidpars = "")             Automatic identification algorithm parameters
        (mode = "ql")
```

For the blue arc (a suggestion is to change the function to Chebyshev and order to 2 since so few lines show up in the blue wavelength regime. One should adjust/change these parameters as the user prefers):

```
epar identify
       images = "comp04b.0001"  Images containing features to be identified
        crval =                 Approximate coordinate (at reference pixel)
        cdelt =                 Approximate dispersion
     (section = "middle line")  Section to apply to two dimensional images
    (database = "database")     Database in which to record feature data
   (coordlist = "linelists$idhenear.dat") User coordinate list
        (units = "")            Coordinate units
         (nsum = "10")          Number of lines/columns/bands to sum in 2D images
```

```
      (match = -3.)           Coordinate list matching limit
 (maxfeatures = 50)           Maximum number of features for automatic identification
     (zwidth = 100.)          Zoom graph width in user units
      (ftype = "emission")    Feature type
     (fwidth = 4.)            Feature width in pixels
    (cradius = 5.)            Centering radius in pixels
  (threshold = 0.)            Feature threshold for centering
     (minsep = 2.)            Minimum pixel separation
   (function = "chebyshev")   Coordinate function
      (order = 2)             Order of coordinate function
     (sample = "*")           Coordinate sample regions
    (niterate = 0)            Rejection iterations
 (low_reject = 3.)            Lower rejection sigma
(high_reject = 3.)            Upper rejection sigma
       (grow = 0.)            Rejection growing radius
  (autowrite = no)            Automatically write to database
   (graphics = "stdgraph")    Graphics output device
     (cursor = "")            Graphics cursor input
    (aidpars = "")            Automatic identification algorithm parameters
       (mode = "ql")
```

8. Reidentify the remaining extracted arc files with REIDENTIFY. Run this task separately for the blue and red arc files.

The "reference" arc image is the one you used in IDENTIFY for the blue and red arcs. For the "images" input parameter, I made an @-file ("inred_reident" and "inblue_reident") that contains all the remaining arcs, blue and red separately.

For the red arc reidentification:

```
lpar reident
    reference = "comp22r.0001"  Reference image
       images = "@inred_reident" Images to be reidentified
       answer = "yes"           Fit dispersion function interactively?
        crval =                 Approximate coordinate (at reference pixel)
        cdelt =                 Approximate dispersion
 (interactive = "yes")          Interactive fitting?
    (section = "middle line")   Section to apply to two dimensional images
     (newaps = yes)             Reidentify apertures in images not in reference?
   (override = no)              Override previous solutions?
      (refit = yes)             Refit coordinate function?\n
      (trace = no)              Trace reference image?
```

```
        (step = "10")           Step in lines/columns/bands for tracing an image
        (nsum = "10")           Number of lines/columns/bands to sum
       (shift = "0.")           Shift to add to reference features (INDEF to search)
      (search = 0.)             Search radius
       (nlost = 0)              Maximum number of features which may be lost\n
     (cradius = 5.)             Centering radius
   (threshold = 0.)             Feature threshold for centering
 (addfeatures = no)             Add features from a line list?
   (coordlist = "linelists$idhenear.dat") User coordinate list
       (match = -3.)            Coordinate list matching limit
 (maxfeatures = 50)             Maximum number of features for automatic identification
      (minsep = 2.)             Minimum pixel separation\n
    (database = "database")     Database
    (logfiles = "logfile")      List of log files
    (plotfile = "")             Plot file for residuals
     (verbose = yes)            Verbose output?
    (graphics = "stdgraph")     Graphics output device
      (cursor = "")             Graphics cursor input\n
     (aidpars = "")             Automatic identification algorithm parameters
        (mode = "ql")
```

For the blue arcs reidentification:

```
lpar reident
    reference = "comp04b.0001"  Reference image
       images = "@inblue_reident" Images to be reidentified
       answer = "yes"           Fit dispersion function interactively?
        crval =                 Approximate coordinate (at reference pixel)
        cdelt =                 Approximate dispersion
 (interactive = "yes")          Interactive fitting?
     (section = "middle line")  Section to apply to two dimensional images
      (newaps = yes)            Reidentify apertures in images not in reference?
    (override = no)             Override previous solutions?
       (refit = yes)            Refit coordinate function?\n
       (trace = no)             Trace reference image?
        (step = "10")           Step in lines/columns/bands for tracing an image
        (nsum = "10")           Number of lines/columns/bands to sum
       (shift = "0.")           Shift to add to reference features (INDEF to search)
      (search = 0.)             Search radius
       (nlost = 0)              Maximum number of features which may be lost\n
```

```
    (cradius = 5.)             Centering radius
  (threshold = 0.)             Feature threshold for centering
(addfeatures = no)             Add features from a line list?
  (coordlist = "linelists$idhenear.dat") User coordinate list
      (match = -3.)            Coordinate list matching limit
(maxfeatures = 50)             Maximum number of features for automatic identification
     (minsep = 2.)             Minimum pixel separation\n
   (database = "database")     Database
   (logfiles = "logfile")      List of log files
   (plotfile = "")             Plot file for residuals
    (verbose = yes)            Verbose output?
   (graphics = "stdgraph")     Graphics output device
     (cursor = "")             Graphics cursor input\n
    (aidpars = "")             Automatic identification algorithm parameters
       (mode = "ql")
```

9. Run REFSPEC to match up the object frames with their corresponding arc frames. For the input parameter, the @-file contains the extracted object spectra files resulting from APALL. For the "references" input parameter, I created a text file ("refer_blue.txt" and "refer_red.txt") that explicitly assigned the corresponding arc to the object frame since I observed an arc either before or after each of my targets (or sometimes both). If you don't want to provide such a file, then leave the "references" parameter blank and fill in "select", "sort", "time", and "timewrap" parameters of REFSPEC.

An example of the reference file, refer_blue.txt is shown here:

```
Sobj.0009b.0001.fits  comp17b.0001.fits,comp18b.0001.fits
Sobj.0010b.0001.fits  comp19b.0001.fits
Sobj.0011b.0001.fits  comp20b.0001.fits
Sobj.0012b.0001.fits  comp22b.0001.fits
```

```
lpar refspec
       input = "@inbluesobj"   List of input spectra
      answer = "yes"           Accept assignment?
  (references = "refer_blue.txt") List of reference spectra
   (apertures = "")            Input aperture selection list
      (refaps = "")            Reference aperture selection list
   (ignoreaps = yes)           Ignore input and reference apertures?
      (select = "interp")      Selection method for reference spectra
        (sort = "UTMIDDLE")    Sort key
       (group = "")            Group key
```

```
      (time = yes)            Is sort key a time?
  (timewrap = 17.)            Time wrap point for time sorting
  (override = yes)            Override previous assignments?
   (confirm = no)             Confirm reference spectrum assignments?
    (assign = yes)            Assign the reference spectra to the input spectrum?
  (logfiles = "STDOUT,logfile") List of logfiles
   (verbose = no)             Verbose log output?
      (mode = "ql")
```

The output to the screen should look something like this, if you use my method of providing a reference file:

```
[Sobj.0009b.0001] refspec1='comp17b.0001 0.47913721'
[Sobj.0009b.0001] refspec2='comp18b.0001 0.52086276'
[Sobj.0010b.0001] refspec1='comp19b.0001'
[Sobj.0011b.0001] refspec1='comp20b.0001'
```

10. Run DISPCOR on your blue and red object spectra, separately.

The input @-file here contains all the extracted blue or red spectra from APALL. The output @-file just provides filenames of your dispersion corrected spectra. In my practice, I used a prefix of "d" to the file names for "done".

```
lpar dispcor
       input = "@inbluedispcor" List of input spectra
      output = "@outbluedispcor" List of output spectra
 (linearize = no)             Linearize (interpolate) spectra?
  (database = "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 = no)             Conserve total flux?
     (blank = 0.)             Output value of points not in input
  (samedisp = no)             Same dispersion in all apertures?
    (global = no)             Apply global defaults?
 (ignoreaps = no)             Ignore apertures?
```

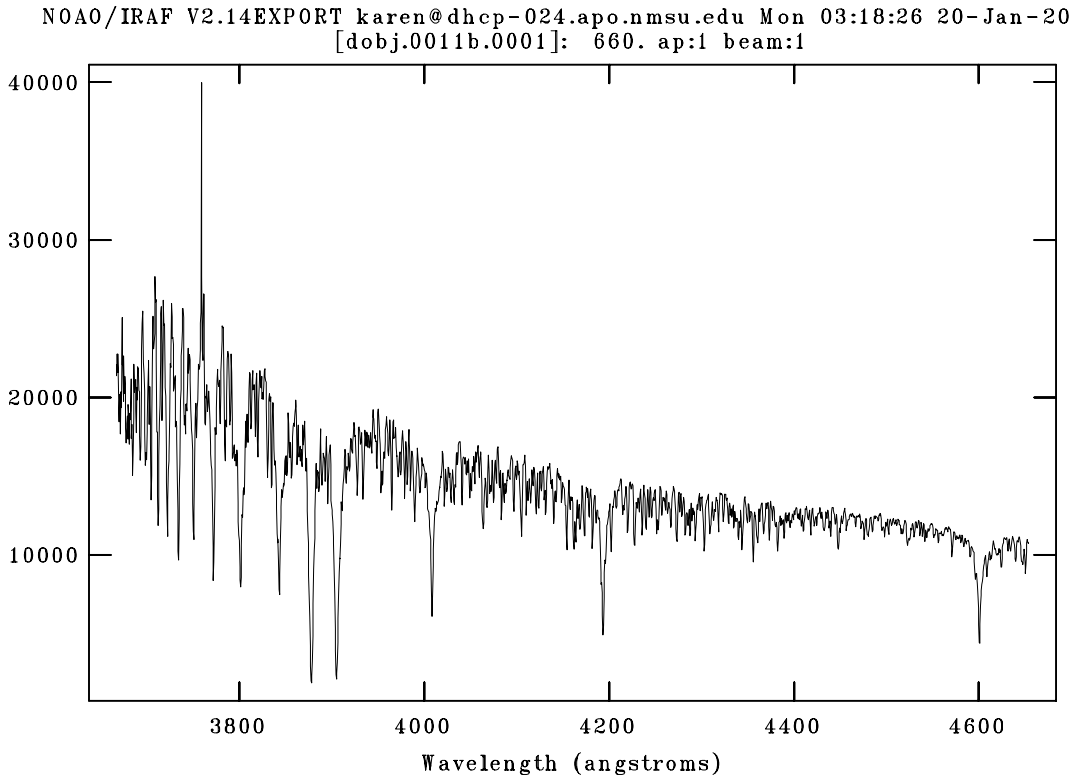

Figure 10: Dispersion corrected blue spectrum from DISPCOR.

```
 (confirm = no)              Confirm dispersion coordinates?
(listonly = no)              List the dispersion coordinates only?
 (verbose = yes)             Print linear dispersion assignments?
 (logfile = "")              Log file
    (mode = "ql")
```

OPTIONAL STEPS

11. Determine the Julian dates and put those values in the header files. Be sure to have loaded ASTUTILS at this point. To obtain the HJD and MJD of your object spectra, use the task SETJD. The task may prompt you for the observatory from which you obtained your spectra. Enter "APO" to the query and your Julian date solution will be calculated and displayed on screen. Check your headers to make sure the Julian dates are saved there.

```
lpar setjd
```

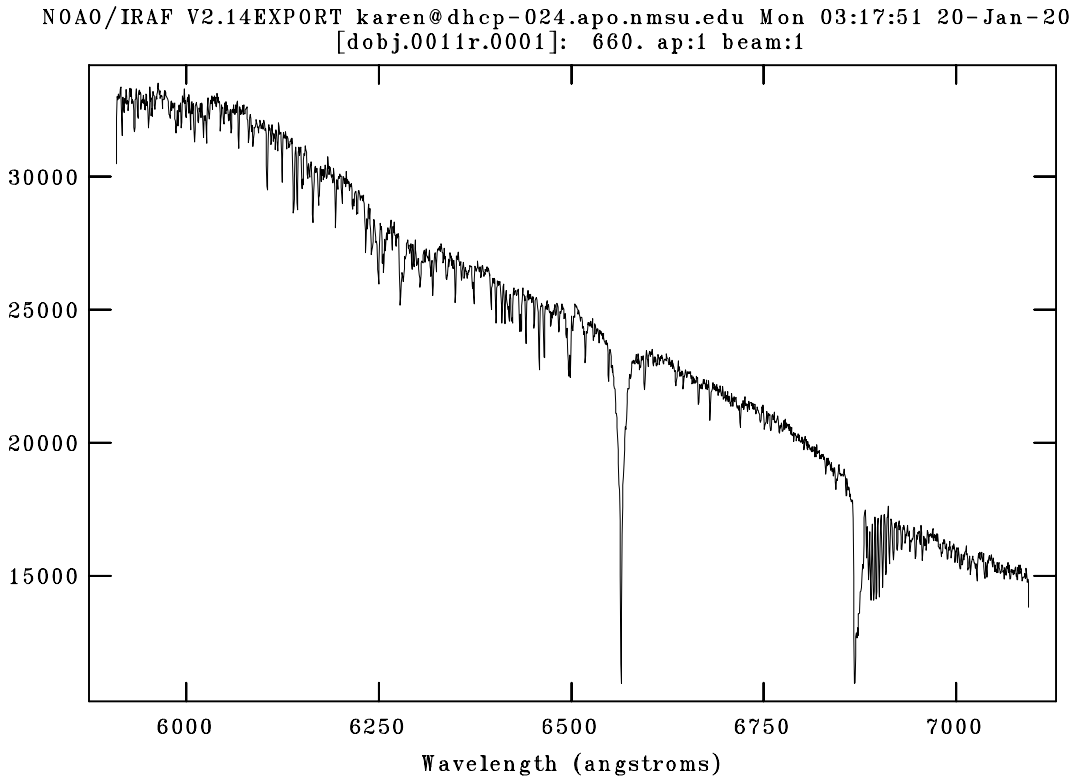

Figure 11: Dispersion corrected red spectrum from DISPCOR.

```
  images  =  d*.fits                Images
  (observa=       )_.observatory) Observatory of observation
  (date   =             date-obs) Date of observation keyword
  (time   =                   ut) Time of observation keyword
  (exposur=              exptime) Exposure time keyword
  (ra     =                   ra) Right ascension (hours) keyword
  (dec    =                  dec) Declination (degrees) keyword
  (epoch  =                equinox) Epoch (years) keyword

  (jd     =                   jd) Output Julian date keyword
  (hjd    =                  hjd) Output Helocentric Julian date keyword
  (ljd    =                  ljd) Output local Julian date keyword

  (utdate =                  yes) Is observation date UT?
  (uttime =                  yes) Is observation time UT?
  (listonl=                   no) List only without modifying images?
  (mode   =                   ql)
```

12. Depending on your science, the continuum fitting and normalizing the spectra must be done carefully. To perform this task, you may choose to do this interactively (highly recommended for the first time) or non-interactively. This task can be run for blue and red spectra together.

If you choose to run CONTINUUM interactively, you can change the fitting function and order to best match where you think the continuum is. You can also adjust the "low_reject" and "high_reject" parameters.

The input @-file contains all the dispersion corrected spectra (in my example, d*.fits). The output @-file contains the object file names with a prefix of "c" for "continuum fitted".

When you have continuum-normalized your spectra, check them visually with SPLOT. Examples of continuum-normalized spectra are presented in Figure 12 and 13. If you see wiggles in your spectrum, it may mean that you used too high an order with your corresponding fitting function. Try again with a lower order, or change the fitting function to something less complicated like Chebyshev or Legendre.

```
lpar continuum
        input = "@incont"       Input images
       output = "@outcont"      Output images
          ask = "yes"
       (lines = "*")            Image lines to be fit
       (bands = "1")            Image bands to be fit
```

```
       (type = "ratio")        Type of output
    (replace = no)             Replace rejected points by fit?
  (wavescale = yes)            Scale the X axis with wavelength?
   (logscale = no)             Take the log (base 10) of both axes?
   (override = no)             Override previously fit lines?
   (listonly = no)             List fit but don't modify any images?
   (logfiles = "logfile")      List of log files
(interactive = yes)            Set fitting parameters interactively?
     (sample = "*")            Sample points to use in fit
   (naverage = 1)              Number of points in sample averaging
   (function = "spline3")      Fitting function
      (order = 1)              Order of fitting function
 (low_reject = 2.)             Low rejection in sigma of fit
(high_reject = 0.)             High rejection in sigma of fit
   (niterate = 10)             Number of rejection iterations
       (grow = 1.)             Rejection growing radius
    (markrej = yes)            Mark rejected points?
   (graphics = "stdgraph")     Graphics output device
     (cursor = "")             Graphics cursor inp
```

13. Response curves and flux calibration. There are several manuals online that go into more depth describing these two tasks. For my own data, I did not perform the flux calibration since my science goals did not require it. I also did not determine the response curves from my standard stars. I refer the reader to please seek other authoritative sources to understand and to perform these task.

14. You are done! You can write a python/PyRAF wrapper for all these steps to expedite the processing once you feel comfortable with the data and the procedures.

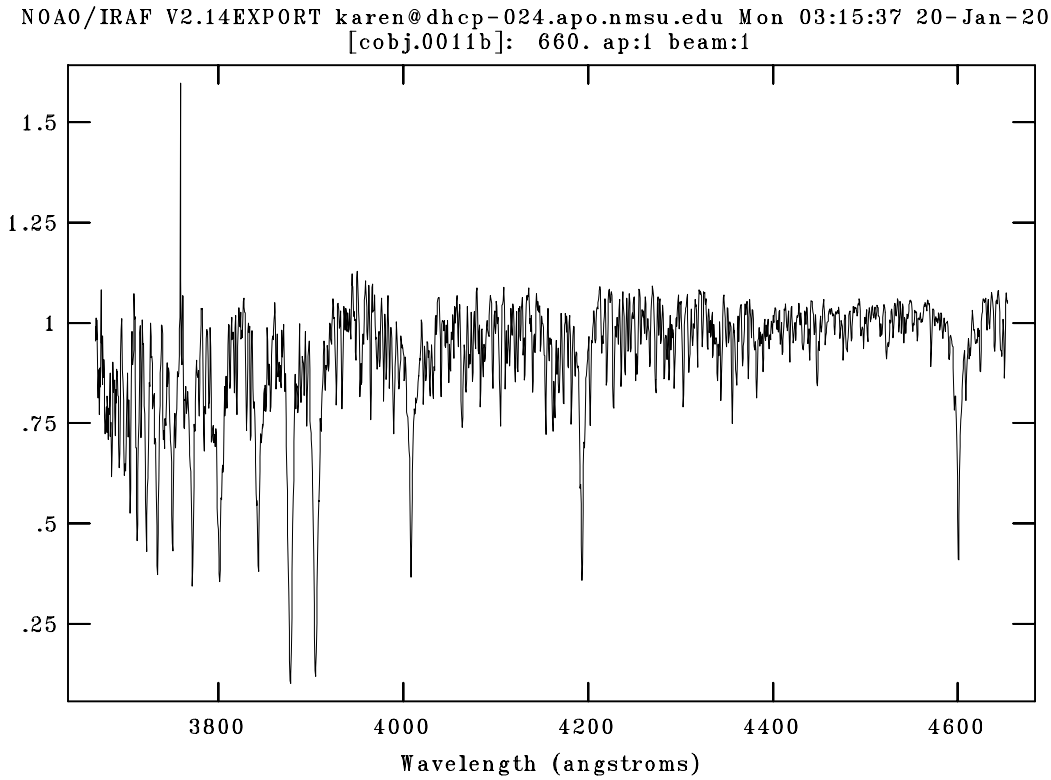

Figure 12: Example of a continuum-flattened blue spectrum.

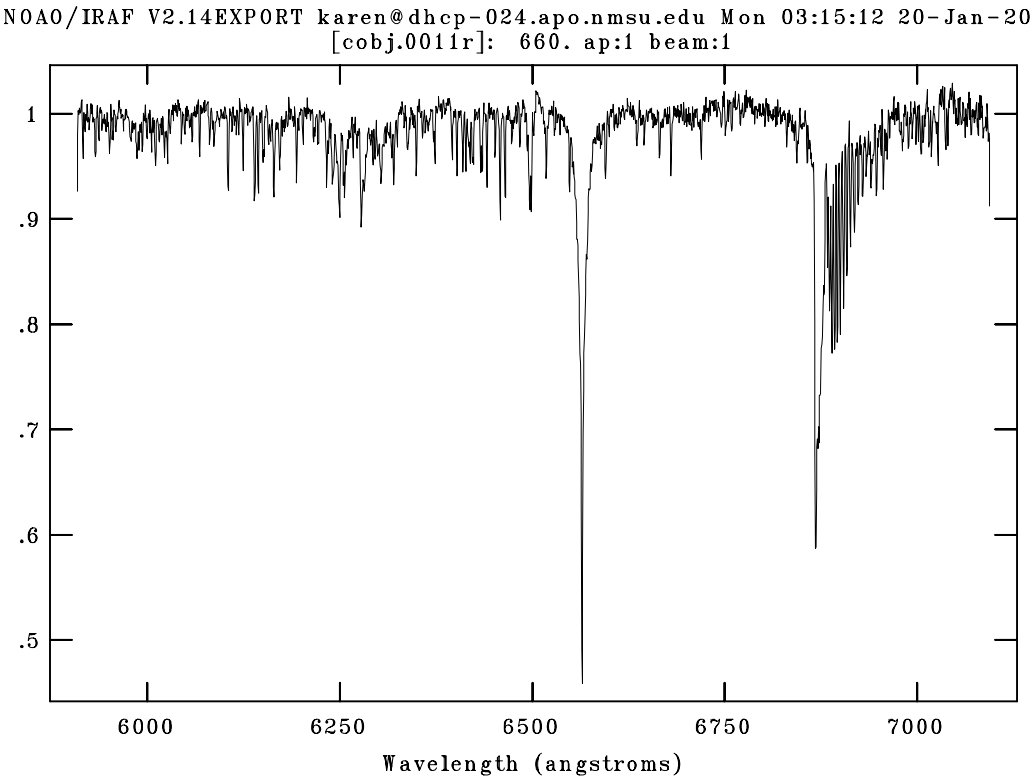

Figure 13: Example of a continuum-flattened red spectrum.