Reducing LRIS longslit spectra in IRAF

Varun B. Bhalerao

Department of Astronomy, California Institute of Technology, Pasadena, CA 91125, USA

varun@astro.caltech.edu

ABSTRACT

The word IRAF scares me to date, though I am slowly getting used to using it. Here I am putting together my notes for reducing LRIS longslit spectra in IRAF. I will try to be a bit general, but only to the extent that I expect my usage to vary. The document will often contain things which are pertinent to my current folders setup, gratings I use, etc.
The scripts referred to in this document are available on request. I might eventually upload a tarball containing all the codes.

1. Preparing the images

1.1. Set up the working directory

Create a working directory. The current organization is
/scr/varun/obsred/*$instrument*/*$yymmdd*/*$side*

For example, an LRIS folder would be /scr/varun/obsred/lris/091016/blue. Copy all the raw data files into a raw/ subdirectory in the working directory. Link the IRAF preferences file.

```
mkdir red
cd red
ln -s ~/login.cl
cd ..
mkdir blue
cd blue
mkdir raw
ln -s ~/login.cl
date +"# %F" > LOG
echo "# copying over raw files: " >> LOG
cd raw
```

1.2. Split the images

Split the images into 4 individual frames. I use a modified `IDL` code from Robert Quimby for this. The code by default extracts from the specified frame (we want 3), adds the headers from fits section 0, and saves the output. It assumes that the raw data is in the `raw/` subdirectory of the present working directory, and creates output files in the `split/` subdirectory.

```
echo splitlrisfits, unit=3 | idl
ds9 &
find split/*fits | sed "s/.fits//" > split.lst
gvim split.lst
ds9slides split.lst
```

Examine the files in `DS9` using the '`n`' & '`p`' keys. Remove any unwanted files from `split.lst`. Save the `IDL` output manually to `split.LOG`. An alternate form of the `IDL` command is:

```
echo splitlrisfits, unit=\"VidInp4\" |idl
```

1.3. Examine image properties

We should also check the headers of all images, for grouping images if required. For example, if slit sizes vary, then you might want to run some of the following processing steps separately for each slit size, etc. We will use this command `checklris`:

```
checklris split/* > checkhead.LOG
gvim checkhead.LOG
```

Internally, what this does is:

```
alias checklris 'pyhead -p exptime,blufilt,dichname,graname,grisname,
grangle,lamps,slitname,trapdoor,targname,object,ttime'
```

1.4. Set up subfolders

First we will create all the lists and subfolders we will use. Using `find` instead of `ls` gives the directory name as a part of the listing. Some items will be modified later as needed

```
mkdir trim
mkdir cosmic
mkdir transform
mkdir spec
mkdir flux
mkdir telluric
```

1.5. Start IRAF

Launch an xgterm. (Remember the "`cd.`" command to get a `cd blah` command printed with proper escapes for whitespaces etc.). Launch the `cl` . Load the default packages. These are not loaded in login.cl itself, so that the user always remembers what has been loaded.

```
imred
ccdred
noao
oned
twod
apex
observatory ("set", "keck", verbose+)
longslit
```

1.6. Add the DISPAXIS keyword

We have made the files list while setting up the directory structure. Now, add the `DISPAXIS` header keyword. The value is '2' for LRIS blue and red sides. It is 1 for dbsp red, and 2 for dbsp blue.

```
hedit @split.lst DISPAXIS 2 addonly=yes verify=no show=no
```

1.7. Trim files

I do not do bias subtraction or flatfielding. Overscan correction is good enough to correct for bias. Flatfields show vertical lines (same direction as the trace), and the lines shift along the slit between different flats. I have certainly seen this effect between flats taken

at the start of the night versus the end of the night. Dividing by these might artificially enhance/suppress the relative strength of source and background regions, hence it is best to not use flat fields. Note that there is a zero image specified in the following command, but it is not used as the `zerocor-` flag is in place. First, let us make the `.lst` file list for output of the trim files

```
sed "s/split/trim/g; s/"\$"/.trim/" split.lst > trim.lst
```

Next, go to the cl and run ccdproc to split the files:

```
ccdproc ("@split.lst", output="@trim.lst", ccdtype="", \
        noproc-, fixpix-, overscan+, trim+, zerocor-, \
        darkcor-, flatcor-, illumcor-, fringecor-, \
        readcor-, scancor-, readaxis="line", \
        biassec="[5:51,1:4096]", trimsec="[52:670,1:4096]", \
        zero="split/bias", scantype="shortscan", nscan=1, interac+, \
        function="cheb", order=2, sample="*", \
        naverage=1, niterate=1, low_rej=3., high_rej=3., grow=1)
```

Observe the "`mean=xxx`" on the right as the output scrolls. A deviant mean value is indicative of some problems. Copy the output manually to `trim.LOG`.

The useful y range is smaller for imaging tasks: use `trimsec="[52:1075,775:3300]"`. The full range is useful in spectra, but the x range is smaller. To select the trimsec and datasec, first open one of the split images in `DS9`. Examine the region of interest, that gives you the trimsec. Next, check the image header for the image size. (The "`l-`" stands for "`longhead-`".) The output should look something like:

```
imhead split/b091016_0090.3 longhead-
split/b091016_0090.3[1155,4096][ushort]: HD26
```

On the command line (or in `ds9`), check the "`DATASEC`" keyword. For this image, it is:

```
[varun@newstar:091016/blue/temp]$ pyhead -p datasec split/b091016_0090.3.fits
# File[ext]    datasec
split/b091016_0090.3.fits    [52:1075,1:4096]
```

DS9 shows that the image extends from "52" to "1075", as said by DATASEC. Hence, the overscan region can be [1:50,1:4096], or [1076:1155,1:4096]. In practice, I have been using the first one. It is better to leave out the first few columns – sometimes they show slightly lower counts. So biassec should be 5:50, as noted in the command above.

1.8. Remove cosmic rays

This involves a little more work: we have to create a script for cosmic ray removal. The commands for L A Cosmic are of the form:

```
lacos_spec split/b091016_0095.3 cosmic/b091016_0095.3.cr \
        cosmic/b091016_0095.3.crbad xorder=7 yorder=30 \
        sigclip=5.5 sigfrac=0.22 gain=1.63 readn=3.6 niter=4
```

Things get a bit tricky here. For bias frames (ttime=0), we do not want to do any cosmic ray removal. These should just be copied over to the output directory with appropriate change of name. For short exposures (say 60 sec or less), we want 2 iterations. For longer exposures, surely ones in 600 sec+ range, we want niter=4. So we need to create 3 file lists: "bias_tr.lst", "short_tr.lst", and "long_tr.lst". We will use HSELECT in IRAF for that. (*Easier-to-copy-paste* versions of these commands are printed on the command line in the next step.) **Can we get rid of bias frames here? the processing is done while trimming...**

```
hselect @trim.lst $I 'ttime = 0' > bias_tr.lst
hselect @trim.lst $I 'ttime > 0 && ttime <= 150' > short_tr.lst
hselect @trim.lst $I 'ttime >150' > long_tr.lst
```

Next, we have to convert these into commands for L A Cosmic. Just use the script saved in ~/bin/tr2cr.sh (see Appendix A for what exactly it does). It asks you for the gain and readnoise. Then it prompts you to ensure that you have already run the hselect command. And finally it produced the required lacosmic.cl file. Note that the script will exit if lacosmic.cl already exists.

```
tr2cr.sh
```

Gain is 1.63 and readnoise is 3.6 for unit3 (vidinp3) on the lris blue side. For the red side (vidinp4), gain=1.164 and readnoise=4.923. Now, run the task:

```
stsdas
task lacos_spec=/home/varun/bin/iraf/lacos_spec.cl
cl < lacosmic.cl > lacosmic.LOG
```

To monitor the progress of the operation, use `monitor lacosmic.LOG`. You will have to `^C` to quit the task when there are no more updates. This is just an alias in `.cshrc`:

```
alias monitor 'tail -f \!:1 --max-unchanged-stats=10 | less'
```

Note that sometimes the task reports a very high number of cosmic rays found: equal to the number of pixels in the image. However, it does not end up subtracting everything from the image. I examined the difference an image with `niter=3` which did not show this problem, and another with `niter=4` which showed this problem: the difference was negligible. In most places it was 0, and in very few pixels over the image it had subtracted what might have been real cosmic rays. Hence, seeing a message like "`2535424 cosmic rays found in iteration 4`" is not a cause for worry, all it means is that you wasted some CPU time.

2. Extracting spectra

The long processing is done, you probably had a good night's sleep while `L.A.Cosmic` was doing its thing. Now it is time to get down to really extracting the spectra.

2.1. Make object lists, prepare arcs

Begin with making a list of all files in the `cosmic/` folder.

```
find cosmic/*.cr.fits | sed "s/.fits//" > cosmic.lst
```

Then, make lists of flats, arcs, and objects. Note that dome flats and internal flats use different settings, and the exact content of the "`lamps`" header will have to be modified for each instrument used. For LRIS blue, arcs are taken with `lamps = "1,0,0,1,1,0"` and internal flats with `lamps = "0,0,0,0,0,1"`. For the red side, internal flats are same as blue, but arcs are taken with `lamps = "0,1,1,0,0,0"`

```
hselect @cosmic.lst $I 'lamps == "1,0,0,1,1,0" && slitname == "long_1.0" && \
        trapdoor=="closed" && exptime > 0' >allarcs.lst
```

```
hselect @cosmic.lst $I 'lamps == "0,0,0,0,0,1" && slitname == "long_1.0" && \
        trapdoor=="closed" && exptime > 0' >internalflats.lst

hselect @cosmic.lst $I 'lamps == "0,0,0,0,0,0" && slitname == "long_1.0" && \
        trapdoor=="open" && exptime > 0' >objects.lst
```

Examine `objects.lst` to make sure that there are no domeflats in the list. If extracting blue side spectra, then make a copy of the arcs list and select the subset of arcs to combine.

```
ds9slides objects.lst
cp allarcs.lst arcs.lst
gvim arcs.lst
```

For the blue side only, make the master arc from `arcs.lst`

```
imcombine @arcs.lst arcs.fits \
      combine=median reject=crreject outtype=real \
      scale=none weight=none zero=none \
      lthreshold=0 rdnoise=3.6 gain=1.63
```

2.2. Wavelength calibration: `IDENTIFY`

For the blue side, we will use arcs. I have verified that for this side, the relative spacing between arcs remains constant, but there may be an offset between various arcs taken on the same night. So, we will use an average set of arcs as the calibration spectrum, and proceed to de-distort the spectrum accordingly. Few steps later, we will calculate and correct for the wavelength offset using a sky line (Section 2.7). For the red side, we will directly use sky lines for wavelength calibration. The lines used for both the sides are listed in Appendix B.

```
identify ("arcs", coordlist="/home/varun/bin/iraf/lris_600blue.list", \
        section="middle column", threshold=0., low_rej=3, \
        high_re=3, niterate=1, match=3., minsep=4, \
        function="cheb", order=5, cradius=12, fwidth=5)
```

First, mark the 4 reference features with `m`. You need to enter only the integer part of the wavelength, it automatically picks the closest line from the coordinates file. Then press '`f`'

to fit, `q` to come back to the full line plot. Then press `l` to identify other lines from the line list, and `f` to fit again. Sometimes, the relatively weak `Hg I` lines at 3131.70 Å, 3125.674 Å can be misidentified at first. In that case, in the fit mode, press `d` to delete that point, `q` to return to the full line list, and `l`, `f` to autoidentify and fit lines again. After this, the lines should be correctly identified. All 15 lines should be fit in good arcs, with residuals less than 0.1 Å.

DBSP: in the default settings, i cover a wavelength range of about 3600-4800 or so (very approx.) The sky lines used are mercury lamps (**?**)

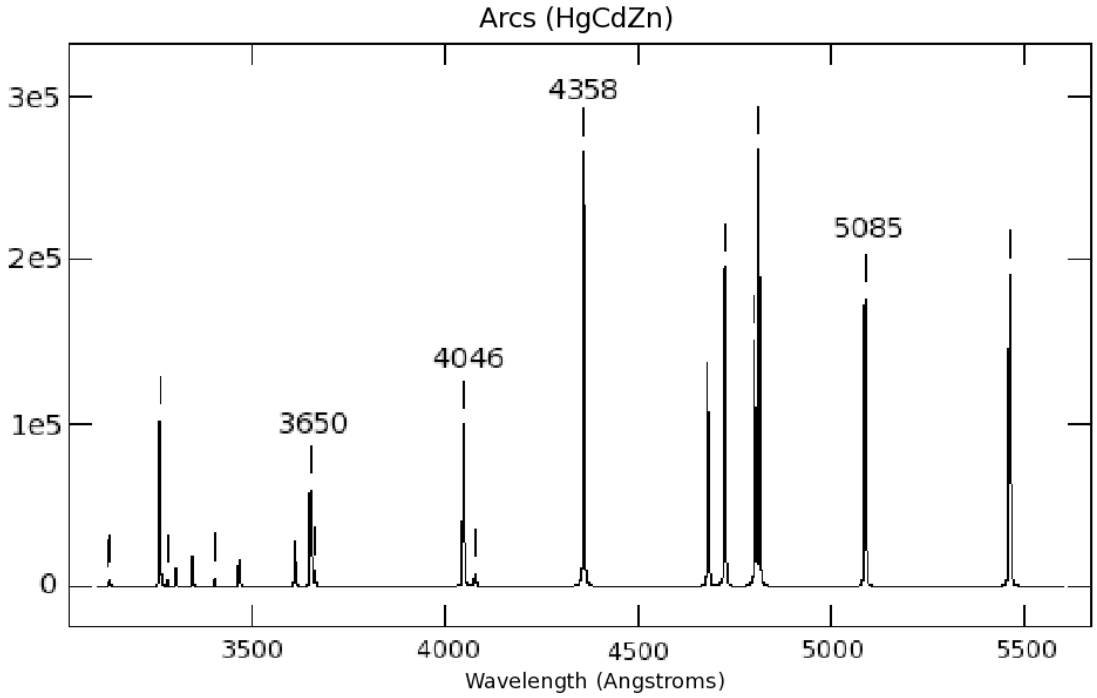

Fig. 1.— Arc lines for blue side of `LRIS`, with "seed" wavelengths marked. There are 15 lines in the file, and good arcs match all giving residuals less than 0.1 Å.

Note: Sometimes the spectra may be flipped. To complicate things further, an impatient observer may not allow the lamps to warm up sufficiently – making it harder to identify arcs in the first place. The test for this is to look for the first 3 lines from Figure 1, namely 3650 Å, 4046 Å and 4358 Å. The group usually retains relative intensity, and the 4358 Å is always the strong, prominent line in the center of the arcs. After marking these three and doing a fit with `f`, hit `q` to return to the plot and see if lines seem more reasonable now.

2.3. Copy wavelengths: REIDENTIFY

The procedure here is slightly different for the red and blue sides. For the blue side, we use only the transformation from "arcs.fits" for all images. For the red side, we REIDENTIFY lines in all images.

```
reidentify ("arcs", "arcs", \
        interac+, section="middle column", newaps+, \
        override-, refit+, trace+, step="5", nsum="5", \
        shift=INDEF, search=INDEF, \
        nlost=5, cradius=5, threshold=0, addfeatures+, \
        coordlist="/home/varun/bin/iraf/lris_600blue.list",\
        match=-3, maxfeatures=50, minsep=2, verbose+, \
        aidpars="", database="database", logfile="logfile")
```

Glance at the last column of the output – it is the RMS error. It should not be larger than 0.1 Å in the central regions. The output is catanated to the file "logfile".

2.4. Correct for geometric distortion FITCOORDS & TRANSFORM

First, we calculate the distortion solution for the instrument:

```
fitcoords ("arcs", fitname="fit", \
        interac+, combine-, database="database", \
        deletions="deletions.db", function="cheb", \
        xorder=4, yorder=6, \
        logfiles="STDOUT,logfile", plotfile="")
```

Here, you should plot residuals versus x coordinate, and zoom-in in user coordinate space. That lets you inspect how the fit changes as a function of x (column) for a given wavelength (line). The IRAF key sequence for this is:

```
x x
y r
r
z z
r
```

Then use `l`, `n` to go through all wavelengths. Press `p` to get back to all data points.

Now, we use this solution to transform the data. We want to transform only the objects now, no need to carry the arcs or flats or bias frames through any more processing. For the blue side, the command is simple. First create the output file on the command line. Remember that trimmed files have the "`.trim`" suffix, while transformed files have a "`.tr`" suffix.

```
sed "s/cosmic/transform/; s/cr/tr/;" objects.lst > transform.lst
```

Then do the transformations in `IRAF`.

```
transform @objects.lst @transform.lst fitarcs \
interp=poly3 flux+ logfiles="STDOUT,logfile"
```

For the red side, the procedure is a bit more involved: we have to use each file's own coordinate map for the transformation. To do this, we have to create a script - just using a "`@fitarcs.lst`" causes all the transforms to be averaged first, and then applied to all spectra. **This part is still to be written.**

2.5. Extract the spectra: `APALL`

Let's start with creating the list of spectra:

```
sed "s/transform/spec/; s/tr/spec/;" transform.lst > spec.lst
```

Now, examine the default parameters for the apall package.

```
apdefault
```

Change the parameters to the following:

```
                                        I R A F
                             Image Reduction and Analysis Facility
PACKAGE = apextract
   TASK = apdefault
```

```
(lower  =                  -5.) Lower aperture limit relative to center
(upper  =                   5.) Upper aperture limit relative to center
(apidtab=                     ) Aperture ID table

(b_funct=             chebyshev) Background function
(b_order=                    2) Background function order
(b_sampl=         -40:-20,20:40) Background sample regions
(b_naver=                   -3) Background average or median
(b_niter=                    1) 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
(mode   =                   ql)
($nargs =                    0)
```

Type `:go` to exit the settings. Next, issue the giant `APALL` command. Answer "`YES`" (in capitals) to all questions. Background regions may have to be tweaked based on other stars appearing in that region, etc. Commands to do so are listed below. Note down any manual tweaks you did in the `LOG` file.

```
apall ("@transform.lst", out="@spec.lst", nfind=1, \
        apertures="", format="multispec", references="", profiles="", \
        interac+, find+, recen+, resize+, edit+, \
        trace+, fittrace+, extract+, extras+, review+, \
        line=2070, nsum=15, lower=-10, upper=10, \
        lower=-5., upper=5., apidtable="", \
        b_function="chebyshev", b_order=2, b_sample="-40:-20,20:40", \
        b_naverage=-3, b_niterate=1, \
        b_low_reject=3., b_high_rejec=3., b_grow=0., \
        width=10., radius=5., threshold=10., \
        aprecenter="", npeaks=INDEF, shift+, \
        llimit=-10., ulimit=10., ylevel=0.1, \
        peak+, bkg+, r_grow=0.1, avglimits-, \
        t_nsum=10, t_step=10, t_nlost=3, t_sample="*", \
        t_function="lege", t_order=3, t_naverage=1, \
        t_niterate=1, t_low_reject=3, t_high_reject=3, t_grow=0, \
        background="fit", skybox=1, weights="variance", \
        pfit="fit2d", clean=yes, \
```

```
saturation=65535., readnoise="3.6", gain="1.63", \
lsigma=4., usigma=4., nsubaps=1)
```

There are 3 reasons to tweak `APALL` fits interactively:

Center the correct target: From a bright star (calibrator), note down the position of the trace. If there is another bright star on the slit later, you will have to recenter the aperture onto the trace. First, take the cursor on the correct target trace, and press '`s`'. It will ask you if you want to recenter the trace, say `yes`. Then press `z` to resize the trace. The background regions are still from the previous trace: press `b` to go to background mode, `f` to fit a new background, and `q` to return to the trace again.

Trace too faint: Use the colon command `:line 1234` to change the fitting to line 1234. That may bring up a stronger trace. Use `DS9` to examine which part of the spectrum shows the best trace.

Star in background region: Press `b` to go into background mode. Too often, the window is too small: press `w`, `a` to show the entire plot. Use `w` followed by two `e` to zoom in on a part of the graph. The best way to set background regions, instead of the interactive "`s`" command, is to use the colon command "`:sample`". For example, you may change the background region to "`:sample -40:-20,40:60`". Press `f` to fit the new background, and `q` to return to the trace part. *Note: For the subsequent images, the background automatically resets to the value entered in the command.*

2.6. Extracting seeing

For accurate RV analysis, you need to measure the seeing in the image spectrum. This can be done by fitting a gaussian to the trace in the spatial direction. Doing this manually is tedious, and very sensitive to which part of the spectrum you use. The aperture file in the `IRAF` database stores the width of the trace, not the FWHM. Hence we use a script for this purpose: `specseeing.pl` (Appendix D). This code extracts the aperture trace from the `IRAF` database, then calls the `IDL` routine `specseeing.pro` with appropriate parameters for each file. `specseeing.pro` fits a gaussian at multiple locations along the trace, and reports the sigma clipped mean for seeing. All output values are Gaussian FWHM, as required by `getvel.pro`. Issue the following command on the command line:

```
specseeing.pl transform.lst
```

The code can handle transform.lst whether or not the files in there have been listed with a .fits extension.

2.7. Wavelength correction (Blue side)

The blue side geometric distortions are calculated from arcs, and there can be a linear offset between arcs and sky wavelengths.

2.7.1. Extract a sky-only spectrum

Any spectrum with say a 30 s or longer exposure shows a prominent O_2 line at 5577.340 Å. In DBSP spectra, we see `Hg I` lines at 4046.564(1) Å and 4358.336(1) Å. This can be used from the `multispec band 3` for anchoring the wavelength solution. For shorter exposures (unfortunately, this includes radial velocity standards), we will have to extract the sky spectrum by summing up the background signal. First, let us separate the long and short exposures:

```
hselect @spec.lst $I "exptime >= 30" > long_spec.lst
hselect @spec.lst $I "exptime < 30" > short_spec.lst
```

Make a list for onedspec output files, and set up the "guess" file for `fitprofs` to be used later. On the command prompt, do:

```
sed "s/.spec/.sky/" long_spec.lst > long_sky.lst
sed "s/.sky/.sky.2001/" long_sky.lst > long_skyoned.lst
echo 5577.340 > skyline.dat
```

`skyline.dat` simply provides a starting point for the line-finding algorithm later, to lock onto the sky line. Back in `IRAF`, issue the splitting command:

```
scopy ("@long_spec.lst", "@long_sky.lst", \
        bands="3", format="onedspec", renumber-,
        offset=0, clobber-, merge-, rebin-, verbose+)
```

2.7.2. Fit the sky line location

```
fitprofs("@long_skyoned.lst", reg="5560 5590", pos="skyline.dat", \
        prof="gauss", gfwhm=3, verb-, \
        nerrsample=50, sigma0=3.6, invgain=0.61, \
        output="", plotfile="", logfile="skyfit.dat")
```

Open `skyfit.dat` in a text editor and scan through the `GFWHM` for each fit. Typical values for LRIS blue are 3.2 to 3.6: if you see a value more than 10% outside this range, examine that file. You may have to demote it to short_spec.lst to extract a better sky line. Go back to the command line to calculate the offsets from the `fitprofs` output. Use the script saved in `~/bin/parseskyfit.pl` (See Appendix C for details). The script takes 2 inputs: the file with a list of spectra, and the name of the `firprofs` log file.

```
parseskyfit.pl long_spec.lst skyfit.dat specshift.cl
```

2.7.3. Apply the shift to the multispec spectra

Run the script created above, to apply corrections to the multispec files. Note that the original files are modified changed here. Then plot sky bands of the spectra in `splot` to examine the results.

```
cl < specshift.cl > specshift.LOG
splot @long_spec.lst xmin=5557 xmax=5597 band=3
```

2.7.4. Working through shorter exposures

Look at the skyshift.pro code and the LOG file in /scr/varun/obsred/lris/091017/blue

2.8. Telluric correction

The blue side spectra dont show telluric absorption features, hence this step is not required. **This part is still to be written for red side spectra.**

2.9. Flux calibration

Flux calibration takes place in 3 parts: calibrating from the standard star, calculating the sensitivity function of the instrument, and finally, applying the calibration to the spectra. First, let us make the file lists:

```
sed "s/spec/flux/g" long_spec.lst > long_flux.lst
sed "s/.flux//" long_flux.lst > long_fluxoned.lst
```

Now, run `standard` – be sure to modify the filename appropriately.

```
standard ("spec/b091016_0115.spec", "standard", star_name="eg247", \
        samestar+, interac+)
```

To check the current values, issue the command: `lpar longslit`. You can look at a list of the standard stars by using the page command: `page onedstds$README` The default calibration directory is `onedstds$spec50cal/`, and has the following stars in it:

```
Standard stars in onedstds$spec50cal/  (3200A - 8100 A)

bd284211   eg247      feige34    hd192281   pg0205134  pg0934554  wolf1346
cygob2no9  eg42       feige66    hd217086   pg0216032  pg0939262
eg139      eg71       feige67    hilt600    pg0310149  pg1121145
eg158      eg81       g191b2b    hz14       pg0823546  pg1545035
eg20       feige110   gd140      hz44       pg0846249  pg1708602

Standard stars in onedstds$spec50cal/  (3200A - 10200A)

bd284211   eg247      feige34    g191b2b    hz44
eg139      eg71       feige66    gd140      pg0823546
eg158      feige110   feige67    hilt600    wolf1346
```

Next, we calculate the sensitivity function of the instrument:

```
sensfunc("standard", "sens", graphs="sr", \
        ignoreaps+, func="cheb", order=6, interac+)
```

Finally, we calibrate the spectra and export them as onedspec for `getvel`.

```
calibrate("@long_spec.lst", "@long_flux.lst", sens="sens", \
        extinct+, flux+, ignoreaps+, fnu-)
scopy ("@long_flux.lst", "@long_fluxoned.lst", \
        bands="1,4", format="onedspec", renumber-,
        offset=0, clobber-, merge-, rebin-, verbose+)
```

3. Radial velocity analysis

Typical settings and values used in getvel. But I am going to modify getvel itself, to be able to read fits files.

A. tr2cr.sh

Here is what the command does internally (lines too large to fit on the pdf are wrapped and indented for clarity):

```
#!/bin/tcsh
if (-f "lacosmic.cl") then
        echo "Error - lacosmic.cl already exists ! I'm running away from here !"
        exit
endif

echo -n "CCD Gain? "
set gain = $<
echo -n "CCD Read noise? "
set readnoi = $<
echo " "
echo "Make sure you ran the 3 hselect commands:"
echo 'hselect @trim.lst $I '"'"'ttime = 0'"'"' > bias_tr.lst'
echo 'hselect @trim.lst $I '"'"'ttime > 0 && ttime <= 150'"'"' > short_tr.lst'
echo 'hselect @trim.lst $I '"'"'ttime >150'"'"' > long_tr.lst'
echo "Press return to continue, ^C to cancel"
set junk = $<

echo "Processing using gain=$gain and readnoise=$readnoi"

cat short_tr.lst \
|sed "s:trim/\(.*\)\(.[0-4].trim\):lacos_spec trim/\1\2 cosmic/\1.cr
        cosmic/\1.crbad:"\
|sed "s:\(.*\):\1 xorder=7 yorder=30 sigclip=5.5 sigfrac=0.22 gain=$gain
        readn=$readnoi niter=2:"\
> lacosmic.cl

cat long_tr.lst \
|sed "s:trim/\(.*\)\(.[0-4].trim\):lacos_spec trim/\1\2 cosmic/\1.cr
        cosmic/\1.crbad:"\
|sed "s:\(.*\):\1 xorder=7 yorder=30 sigclip=5.5 sigfrac=0.22 gain=$gain
        readn=$readnoi niter=4:" \
>> lacosmic.cl
```

```
cat bias_tr.lst \
|sed 's:trim/\(.*\)\(.[0-4].trim\):imcopy trim/\1\2 cosmic/\1.cr:' \
>> lacosmic.cl
```

Another method to do this is to use REGEX "black magic" in gvim rather than the command line. Note that it has been split across mutiple lines here, but needs to be given as a single command on the ":" prompt. The drawback is that you have to somehow figure out where to set niter=2, where to 4. And it lacks the simple imcopy for bias frames.

```
%s:trim/\(.*\).fits:lacos_spec trim/\1 cosmic/\1.cr cosmic/\1.crbad
xorder=7 yorder=30 sigclip=5.5 sigfrac=0.22 gain=1.63 readn=3.6:
```

B. Wavelength calibration lines

Here are lines saved in various files, used for specific gratings.

B.1. LRIS blue: 600 grating

The data are saved at:

```
~/work/bin/iraf/lris_600blue.list
```

Contents of the file are:

```
# line list made for well-exposed, properly warmed up Hg+Cd+Zn+Ar lamps
# using the Keck lists for blends
# http://www2.keck.hawaii.edu/inst/lris/txt/blended_line_list.txt
# but taking blends only for close-by lines of same element
# getting wavelengths from NIST Atomic Spectra Database,
# http://physics.nist.gov/PhysRefData/ASD/index.html
# sky line is 5577.340
5460.750 Hg I
# 5181.98  Zn I - removed, peak just 150 counts
# 5154.660 Cd I - can be kept, but peak weak - 350 counts
5085.822 Cd I
```

```
# 4916.07  Hg I - can be kept, but just 400 counts, assymetric
4810.53  Zn I
4799.912 Cd I
4722.15  Zn I
# 4679.135 Zn/Cd I del
# 4629.81  Zn I - can be kept, 500 counts, on wing of 4679 line
# 4412.989 Cd I del
4358.335 Hg I
# 4339.22  Hg I del
# 4292.88  Zn I - removed, peak is barely 100 counts and on wing of 4358 !
4077.837 Hg I
4046.565 Hg I
3663.18  Hg I bl
3650.158 Hg I
# 3610.508 Cd I
# 3611.01 CdIblend better than the single, but not quite there
# 3466.200 Cd I del
3403.652 Cd I
# 3345.57  Zn I del
#3345.04  ZnI/HgIblend milan
# 3302.58  Zn I del
# 3302.75 ZnI milan
3282.33  Zn I
3261.055 Cd I
3131.70  Hg I
3125.674 Hg I
```

C. parseskyfit.pl

```
#!/usr/bin/perl
if ($#ARGV+1 ne 3) {
        print "Usage: parseskyfit.pl long_spec.lst skyfit.dat\n";
        print "\tlong_spec.lst = list of multispec files to be corrected\n";
        print "\tskyfit.dat    = offsets to be applied\n";
        print "\tspecshift.cl  = name of IRAF script to be created\n";
        print "\t                (WILL BE OVERWRITTEN !)\n";
        exit;
```

```
}

open (INFILE, $ARGV[0]);
open (INSHIFT, "grep -E '^ *[0-9]' $ARGV[1] | awk '{print 5577.340-\$1}' |");
open (OUT, ">$ARGV[2]");
#grep -E "^ *[0-9]" skyfit.dat | awk '{print 5577.340-$1}' > skyshift.dat
while (<INFILE>){
        chomp;
        @files = (@files, $_);
}
while (<INSHIFT>){
        chomp;
        @shifts = (@shifts, $_);
        #print "Received $_ \n";
}

for($i=0; $i<=$#files; $i++){
        print OUT "specshift $files[$i] $shifts[$i] verb+\n";
}
print "Done! Created IRAF script $ARGV[2]\n";
```

D. specseeing.pl

```
#!/usr/bin/perl
use Switch;

# use $argv[0] to get a list of files
# open each file and calculate max - min as seeing

open (OUT, ">seeingscript.pro");

open (IN, @ARGV[0]);
while($apfile = <IN>){
        $low = -1;
        $high = -1;
        $xmin = -1;
        $xmax = -1;
```

```
$polytype = -1;
$ncoeffs = -1;
$nlines = -1;
$axis = -1;
$xref = -1;
$yref = -1;

chomp($apfile);
$apfile =~ s/.fits//;
$apfile2 = $apfile;
$apfile2 =~ s/\//_/g;
$apfile2 = "database/ap$apfile2";
open(AP, $apfile2) or next;
while($line = <AP>){
        if ($line =~ /^\s+axis\s+/){
                split(/\s+/, $line);
                $axis = @_[2];
        }
        if ($line =~ /^\s+center\s+/){
                split(/\s+/, $line);
                $yref = @_[2];
                $xref = @_[3];
        }
        if ($line =~ /^\s+curve\s+/){
                split(/\s+/, $line);
                $nlines = @_[2];
                $polytype = <AP>;
                chomp($polytype);
                $polytype =~ s/\s+//g;
                $ncoeffs = <AP>;
                chomp($ncoeffs);
                $ncoeffs =~ s/\s+//g;
                $xmin = <AP>;
                chomp($xmin);
                $xmin =~ s/\s+//g;
                $xmax = <AP>;
                chomp($xmax);
                $xmax =~ s/\s+//g;
```

```
                        $sci = "[";
                        for ($i=4; $i<$nlines; $i++) {
                                $line = <AP>;
                                chomp($line);
                                $line =~ s/\s+//g;
                                @ci = (@ci, $line);
                                $sci = "$sci$line, ";
                        }
                        $sci =~ s/, $/\]/;
                }
        }
        # errors => axis !=1 OR nlines - ncoeffs != 4 OR polytype != 2
        if ( ($axis ne 1) or ($nlines-$ncoeffs ne 4) or ($polytype-2 ne 0) ){
                warn "## Error! $apfile.fits axis = $axis, nlines = $nlines, ";
                warn "ncoeffs = $ncoeffs, polytype = ##$polytype##\n";
                print OUT "print, 'Skipping $apfile.fits'\n";
        } else {
                #print OUT "print, 'Processing $apfile.fits'\n";
                print OUT "specseeing, '$apfile.fits', $xref, $yref, $xmin, ";
                print OUT "$xmax, $sci, xstep=50, xaverage=10, seeguess=12.\n";
        }
}

close(OUT);
system('ln -s ~/bin/idl/mpfit.pro');
system('echo "@seeingscript.pro" |idl');
```