Spectroscopy Reduction Handout

This handout will give you a basic understanding of the many steps and subtleties of processing spectral data. Please note that we will be using IRAF through PYRAF package to perform all spectroscopy reduction. IRAF routines are extremely powerful but often rather fussy, so make sure to carefully follow all of the steps in order, and don't be surprised if things go wrong along the way.

Before You Open PYTHON

1) Make a science directory to perform all of your spectral processing. Make sure to include (for a given night!) master bias, master dark, master dome flat, arc lamp spectra, and object spectra.
2) Copy the file `~/physics100/login.cl` into your science directory.
3) In this directory, make a sub-directory called `pyraf`.
4) Edit the `login.cl` file. The variables `home` and `imgdir` should reflect the working directory you just made, while the `userid` should have a username from your group and not `sehlert`.

Using IRAF

Unlike most routines that we have used in class so far, the IRAF routines are a bit more interactive and include some graphical features. Although these software packages have been tested by TAs and professional astronomers, there is a non-negligible chance of this going wrong in the analysis. Please make sure to read and follow the instructions carefully to avoid trouble.

Overview of Data Flow

As in most cases, it makes sense to try and determine what we truly wish to do with the data. In this case, the pipeline is something as follows (detailed recipe is provided in the next section):

- Process the image for CCD noise and inhomogeneities (bias, dark, flat), using the information in the dome flat images to create the 2D spatial flat image.
- Using spectra taken of the arc lamps, identify the range of wavelengths observed in the star image.
- Add the wavelength information to your science observation.
- Calculate the response of the spectrometer over this wavelength range by comparing the observed and known spectra of a calibration star.
- Remove the spectrometer response from the science observation, resulting in a spectrum with physical units.

You will need to **do this FIRST for calibration star**, and then for your science star. This makes it easy for us to see where things are going wrong.

Step I: Preliminary data calibration

Please note that for spectral data reduction we will be using **ipython and not ipython notebooks**, since notebooks cannot handle interactive displays. To start, simply type `ipython` in your terminal. You also want to use the same IPYTHON terminal for all steps of the analysis.

Before we begin the spectral calibration, we need to correct the images for biases, darks, and flats. Biases and darks are done in the same way as for imaging, but flats require a bit more care.

1) Make sure that you have your MasterBias, MasterDark, and MasterFlat copied to the directory you are working in.

2) Upload all of the relevant and useful PYTHON and IRAF libraries. Make sure to include two `dispaxis` commands, otherwise the program will assume the spatial axis is the spectral one, leading to completely incorrect results.

```
import numpy as np
import complete_process #this is your library
from pyraf import iraf
iraf.noao()
iraf.twodspec()
iraf.onedspec()
iraf.longslit()

iraf.apextract()
iraf.longslit.dispaxis=2
iraf.apextract.dispaxis=2
```

3) Now we will create special flat fields for spectroscopy. We will use a PYRAF routine `response`, which creates a 2D spatial flat image out of the dome flat that has only one dimension of spatial information. A sample call is

```
iraf.response('MasterFlat.fits','MasterFlat.fits', 'SpecMasterFlat.fits',
interactive='no',sample='*',naver=1,function='spline3',
order=1,low_rej=3.0,high_reject=3.0,niterat=1.0,grow=0)

Dispersion axis (1=along lines, 2=along columns, 3=along z) (1:3) (1): 2
```

You want to make sure that the dispersion axis is set to 2. The name of the output spectroscopic flat field image is `SpecMasterFlat.fits` in this case.

4) Process your science image and your arc calibration images from the very beginning: correct for biases, darks and flats. The flat field image you want to use is the output of `iraf.response()` (not your default MasterFlat!). You will want to use your `complete_process.py` routines to do this.

Step II: Calibration star reduction

1) IRAF routine `apall` determines the extraction aperture, models the slit, and extract basic spectrum. You will do this twice: once for the science image, and then again for the arc lamp. The calls to this routine are different, and the science call occurs first. Keep in mind, that in all interactive steps you can type `?` and get a list of all commands. You would then type `q` in the terminal to change back to the original task at hand.

```
  iraf.apall('${image}',interactive='yes')  #$image means your clean, normalized file
  #i.e. star_clean_norm.fits
```

You will then be asked a bunch of yes/no questions which you will answer:

```
  Find apertures for ${image}? yes
  number of apertures to find: 1
  Resize apertures for ${image}? yes
  Edit apertures for ${image}? yes
```

and then a graphical window will show up. There should be a bump on the plot corresponding to the aperture. If you are lucky, IRAF will mark the aperture for you automatically, but you may also need to identify the aperture manually. To do this, place your mouse cursor over bump and press the letter `m`. IRAF will then draw an aperture on the window. Then you want to press the letter `q` to quit. You will be offered a new set of questions in the *terminal*, but you will answer them in the *plotting window*. Also, they tend to stack on a single line with no spaces, so they are hard to read. Simply put, the answer is yes for all of them:

```
  Trace apertures for ${image}? yes
  Fit traced positions for ${image} interactively? yes
  Fit curve to aperture 1 of ${image} interactively? yes
```

The screen will change to the fit for the trace of the aperture. Basically, it is accounting for the fact that the aperture may not be a straight line across the image. Press `q` if you are happy with the fit. It will then ask you a few more questions that you answer yes to:

```
  Write apertures for ${image} to database? yes
  Extract aperture spectra for ${image}? yes
  Review extracted spectra from ${image}? yes

#The following only shows up if file image already exists
  Clobber existing output image ${image}? yes

Review extracted spectrum for aperture 1 from ${image}? yes
```

You will then see the uncalibrated spectrum for the aperture. Press `q` to exit.

Now we will run the same routine on the arc lamp spectrum at the same dial setting, but **we will use the same aperture as for the science observation**. Type:

```
  # Be careful! Do not include the .fits or .ms.fits for your reference file name!
  iraf.apall('${arclamp}',reference='${image}')
```

Please note the answers to the questions which follow:

```
  Recenter apertures for ${arclamp}? ('yes'): no
  Resize apertures for ${arclamp}? ('yes'): no
  Edit apertures for ${arclamp}? ('yes'): no
  Trace apertures for ${arclamp}? ('yes'): no
  Write apertures for ${arclamp} to database? ('yes'): yes
  Extract aperture spectra for ${arclamp}? ('yes'): yes
  Review extracted spectra from ${arclamp}? ('yes'): yes
# The following will show up only if you've already used this arc
  Clobber existing output image arc.ms? ('no'): yes
  Review extracted spectrum for aperture 1 from ${arclamp}? ('yes'): yes
```

After typing that last yes, you will see the arc lamp spectrum. If you are successful with these two steps, then you will see an arc lamp spectrum identical to the traces you saw at the observatory. The output files from this analysis have the same name as the input files, but with a `.ms` before the `.fits` extension, so for the above call the output should be something like `arclamp.ms.fits`.

2) PYRAF routine `identify` will identify the lines from the arc lamp. We use a graphical window to mark a few estimates of line positions, and it figures out the rest:

```
  iraf.identify('${arclamp}.ms')
```

In the GUI, you use `m` to mark a line, and you will then type in your estimate of the line position. Do this for two or three lines, and then type `f` to perform a fit. You will see the residuals from the fit, which you can exit by typing `q`. This will take you back to the original spectrum. You can then have it determine all of the lines by typing `l` (a lowercase L). You will see a bunch of identified lines in the window. Look over some of these lines and make sure that the ones you know are identified as helium/neon lines. You can repeat the steps above (`m`, `f`, and `l` in order to iteratively improve the fit). Close the window, save the fit, and then move on to the next step.

```
Write feature data to the database (yes)? yes
```

3) PYRAF routine `hedit`. Now that you wavelength calibration is all done, we need to add this information to the image, so that it knows which image pixels correspond to which wavelengths. To do this, we use the command

```
  iraf.hedit('${image}.ms.fits',add='yes')
```

in order to alter the header information to keep track of this. You will then be asked to say what key word you are editing, where you will type `REFSPEC1`. For the value expression, include the processed arc lamp file `${arclamp}.ms.fits`:

```
  fields to be edited ('REFSPEC1'): REFSPEC1
  value expression: ${arclamp}.ms.fits
  add ${image}.ms.fits,REFSPEC1 = ${arclamp}.ms.fits
  update ${image}.ms.fits ? (yes): yes
  ${image}.ms.fits updated
```

4) PYRAF routines `dispcor` and `wspectext` are used in the last two steps in the spectral processing. The first step changes the headers and database so that you can analyze the spectrum, while the second converts the spectrum into a text file. Both of these will fail if the output already exists. They are used as follows:

```
iraf.dispcor()

List of input spectra: ${image}.ms.fits
List of output spectra: ${image}_dispcor.fits #hereafter ${imagedispcor}
```

This creates a new FITS file that has the wavelength information along with the source spectrum. You should see an output line that looks like this:

```
RC_dispcorrected_actual.fits: ap = 1, w1 = 3780.056, w2 = 5786.27,
dw = 3.933753, nw = 511
```

The values for `w1` and `w2` correspond to the wavelength range of the spectrum, so if these don't look like your expectations, take note.

```
iraf.wspectext(header='no')
```

This converts the FITS data for the spectrum into a text file that is easily read by PYTHON (e.g. with `numpy.loadtxt`), and easily plotted using `matplotlib`. Typically you will want to place the command `header='no'` into the parentheses so that the header information does not get printed into the text file along with everything else.

5) PYRAF routine `standard`. Compare your ESO standard star spectrum to the known spectrum using this routine. You provide the name of the star and your measured spectrum of it. Here are the interactive options you need to use. Make sure that there is a slash at the end of path to the caldir! And remember, this is only done for the calibration star!

```
iraf.standard(caldir='/afs/ir/users/o/n/ondrej/physics100/archive/specstandards/')

file root name: ${imagedispcor}
output flux file: 'std' #can hit ENTER

${imagedispcor}(1): #nothing needs to be typed here
Observatory Identification: 'lick'
${imagedispcor}: Airmass: 1.0 #make sure to type this in
Star name in calibration list: # name of star from ESO list #if nonexistent,
#IRAF will offer available choices
Edit bandpasses? (no|yes|NO|YES|NO!|YES!) ('no'): yes
```

Press `q` to quit the interactive window.

6) PYRAF routine `sensfunc`. Model the spectral response of the telescope and detector from the standard star spectrum using a `iraf.sensfunc()` routine. (Only done for the calibration star!)

```
iraf.sensfunc()

Input: 'std'
Output: 'sens'
Fit aperture 1 interactively? (no|yes|NO|YES) (no|yes|NO|YES) ('yes'): yes
```

Press `q` to quit the interactive window. You now have a model for how to do "flat-fielding" in the wavelength axis. You now want to correct the science star spectra – although only the ones where you expect the response you modeled to be appropriate.

7) PYRAF routine `calibrate` calibrates your spectrum using the modeled response. This takes the response function you modeled in the previous step and applies it to the input spectrum. (Only done for the calibration star!)

```
iraf.calibrate()

Input: ${imagedispcor}
Output: name of your choosing #e.g. image_calib
```

The output file from this step is now your fully calibrated spectrum of your calibration star. You will probably want to convert it to text using the routine:

```
iraf.wspectext(header='no')
```

You can then read this text file into PYTHON and look at the spectrum. What are the units?

Step III: Science star reduction

Now that we created a sense function using our calibration star, we can properly reduce our science star!

1) From within IPYTHON, copy your sense function:

```
cp sens.fits sens.0001.fits
```

2) Perform points 1–4 from Step II. Note that now `$image` refers to your **science star**.

3) This time we will use a slightly different PYRAF `calibrate` routine, which calibrates your spectrum using the modeled response. This takes the response function you modeled in the previous step and applies it to the input spectrum.

```
iraf.imred()
iraf.specred()
iraf.imred.specred.calibrate(sensitivity='sens')

Input: ${imagedispcor}
Output: name of your choosing
Extinction:   #leave blank
Airmass (1.0:): 1.0
```

The output file from this step is now your fully calibrated spectrum, upon which you will perform all of your science analysis. You will probably want to convert it to text using the routine `iraf.wspectext(header='no')`. You can then read this text file into PYTHON and look at the spectrum. What are the units?

Now you should have two text files with calibrated stellar spectra, as well as at least one calibration spectrum. What are the y-axis units? (Hint: look up your calibrated spectrum fits file header!). Remember to *always* put units on your axis (important for completion of Lab 2).

For submission:

Move your calibrated spectra to ~`/physics100/student_spectra` directory, and upload the path to these spectra via Canvas for credit. Make sure not to overwrite your classmates' files!