1. Pre-Processing

```
--> noao
--> imred
imred/:
 argus/         ctioslit/       hydra/          kpnocoude/      vtel/
 bias/          dtoi/           iids/           kpnoslit/
 ccdred/        echelle/        irred/          quadred/
 crutil/        generic/        irs/            specred/
--> ccdred
ccdred/:
 badpiximage    ccdlist         combine         mkillumcor      setinstrument
 ccdgroups      ccdmask         darkcombine     mkillumflat     zerocombine
 ccdhedit       ccdproc         flatcombine     mkskycor
 ccdinstrument  ccdtest         mkfringecor     mkskyflat
```

NOTE:

(1) Change all the names to ".fits", NOT ".fit"

(2) All the images should have header "BZERO" and "BSCALE". IF THEY DON'T HAVE, type the following:

*** I HAVE ALREADY DONE THIS ***

```
ccdhedit *.fits BZERO 32768
ccdhedit *.fits BSCALE 1
cd bias/
ccdhedit *.fits BZERO 32768
ccdhedit *.fits BSCALE 1
cd ../dark/
ccdhedit *.fits BZERO 32768
ccdhedit *.fits BSCALE 1
cd ../flat/
ccdhedit *.fits BZERO 32768
ccdhedit *.fits BSCALE 1
cd ../ref/
ccdhedit *.fits BZERO 32768
ccdhedit *.fits BSCALE 1
cd ..
```

*** I HAVE ALREADY DONE THIS ***

The real pixel value should be

$$\text{real value} = \text{pixel} * \text{BSCALE} + \text{BZERO}$$

People sacrificed the 15^{th} bit of 16-bit system, and used it as a "sign" bit. Usually, $\mathrm{BZERO} = 2^{15} = 32768$, and $\mathrm{BSCALE}{=}1.0$. If you do not add these header keywords to the image header, you will result in many negative values from bias/dark/flat raw images, which eventually lead you to a wrong result.

1-1. BIAS

bias is also called zero or offset.

```
--> cd bias/
--> ls
bias1.fits  bias2.fits  bias3.fits
--> ls bias*.fits > bias.list
```

You now have a list file. You can open this with gedit bias.list or vi bias.list .

Quiz: Try what happens when you do `ls bias* > bias.list` (delete '.fits'). Open the bias.list file, and you will find something happened. Why?

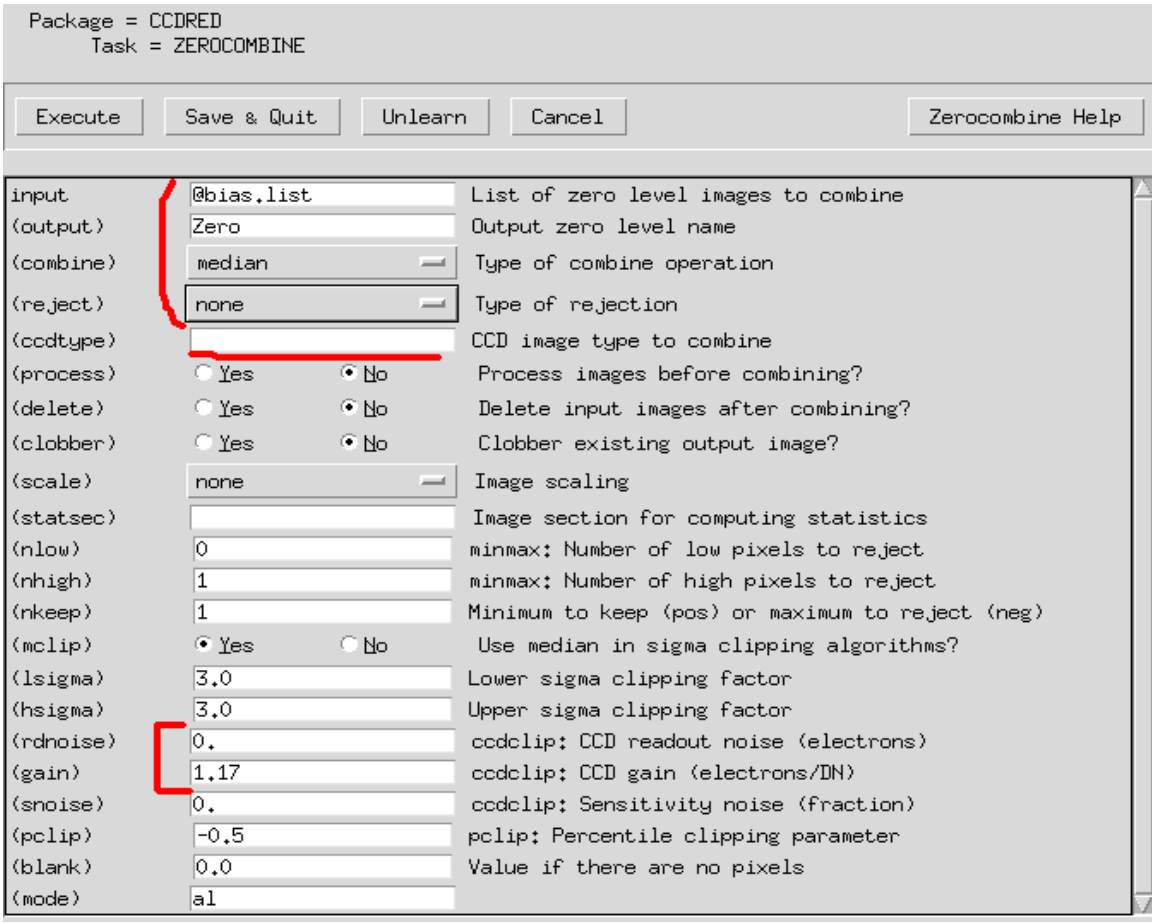
Using ZEROCOMBINE

```
--> epar zerocombine
```

* You can always use IMCOMBINE and IMARITH, or COMBINE (different from IMCOMBINE). ZERO/DARK/FLATCOMBINE tasks are all small variants of COMBINE, and run through CCDPROC task.

Set parameters as in the figure.

```
--> zerocombine
List of zero level images to combine ('@bias.list'): (type enter)
```



```
--> ls
bias1.fits  bias2.fits  bias3.fits  bias.list  logfile     Zero.fits
--> !ds9 Zero.fits &


--> ls
bias1.fits  bias2.fits  bias3.fits  bias.list  logfile     Zero.fits
--> imstat Zero.fits
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
            Zero.fits   3214848     113.6     6.652       87.     6841.
--> imstat bias*
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
Error reading image bias.list ...
           bias1.fits   3214848     114.1     8.653       79.     6841.
           bias2.fits   3214848     112.9     8.715       78.     6872.
           bias3.fits   3214848     113.9     8.673       78.     6838.
```

1-2. DARK

```
--> cd ../dark
--> cp ../bias/Zero.fits .
--> ls
dark-001_20.fits  dark-002_20.fits  Zero.fits

--> epar darkcombine
```
Set parameters as in the figure.
```
--> epar ccdproc
```
Set parameters as in the figure.
```
--> darkcombine
```

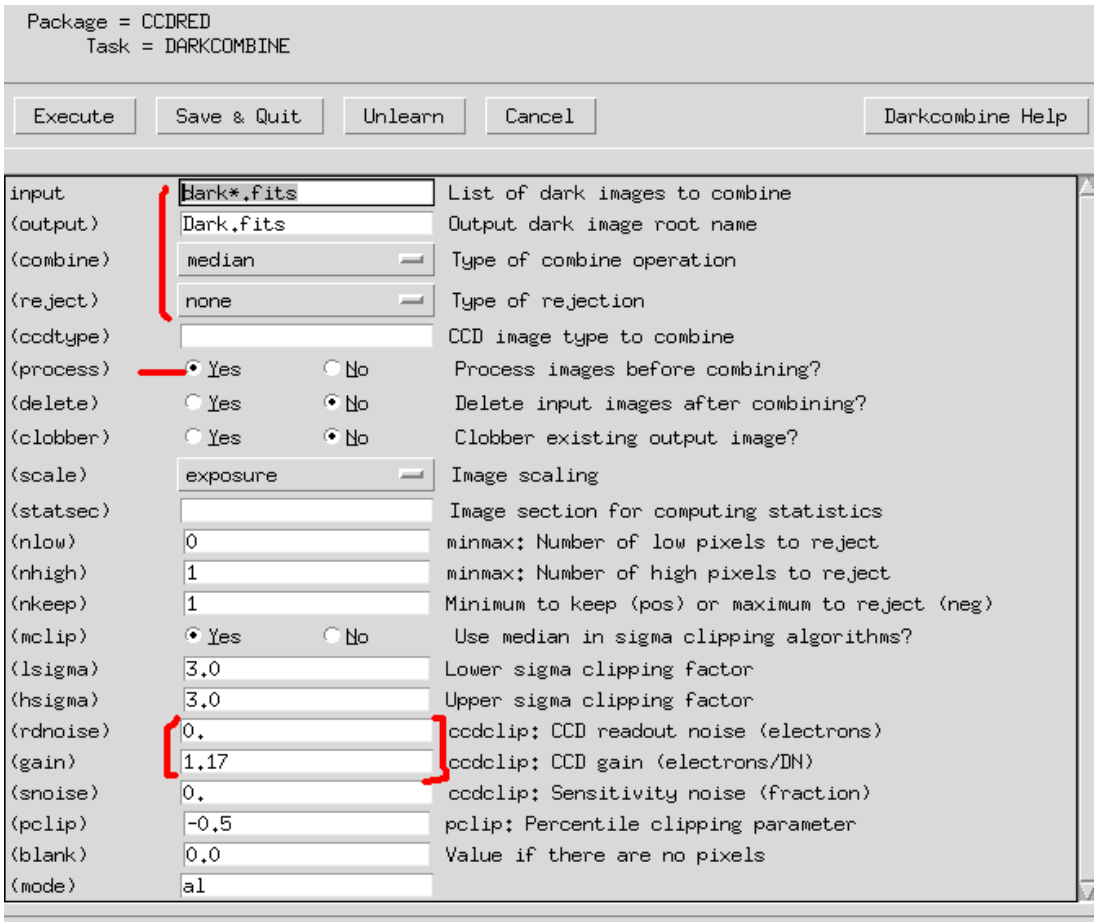
* If you do DARKCOMBINE, the input dark files may be changed (automatically zero subtracted). If you type CCDLIST, you can see [Z] to each dark files, which means the zero correction has already been done. This can be avoided by using CCDPROC:

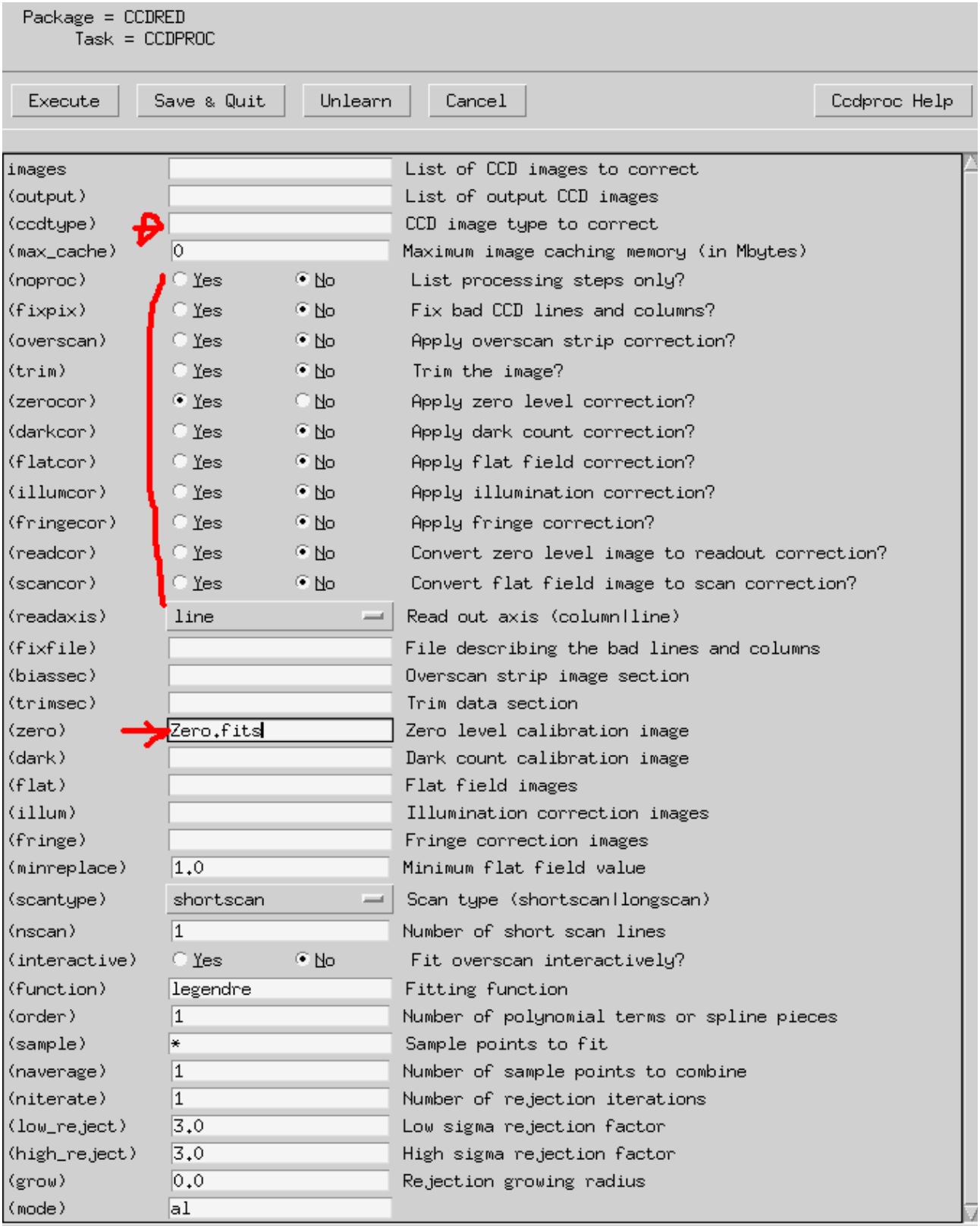
```
epar ccdproc
input=dark*, (output)=Dark.fits, (zerocor)=yes, (zero)=Zero.fits.
```

Or COMBINE or IMCOMBINE to get the median-combined dark, and then subtract Zero.fits by IMARITH:

```
combine dark*.fits tmp.fits combine=median
imarith tmp.fits - Zero.fits Dark.fits
```

Of course (IM)COMBINE and IMARITH all can be controlled by using epar (epar combine, etc) as before. If you have better ways to do this, please let TA know T__T





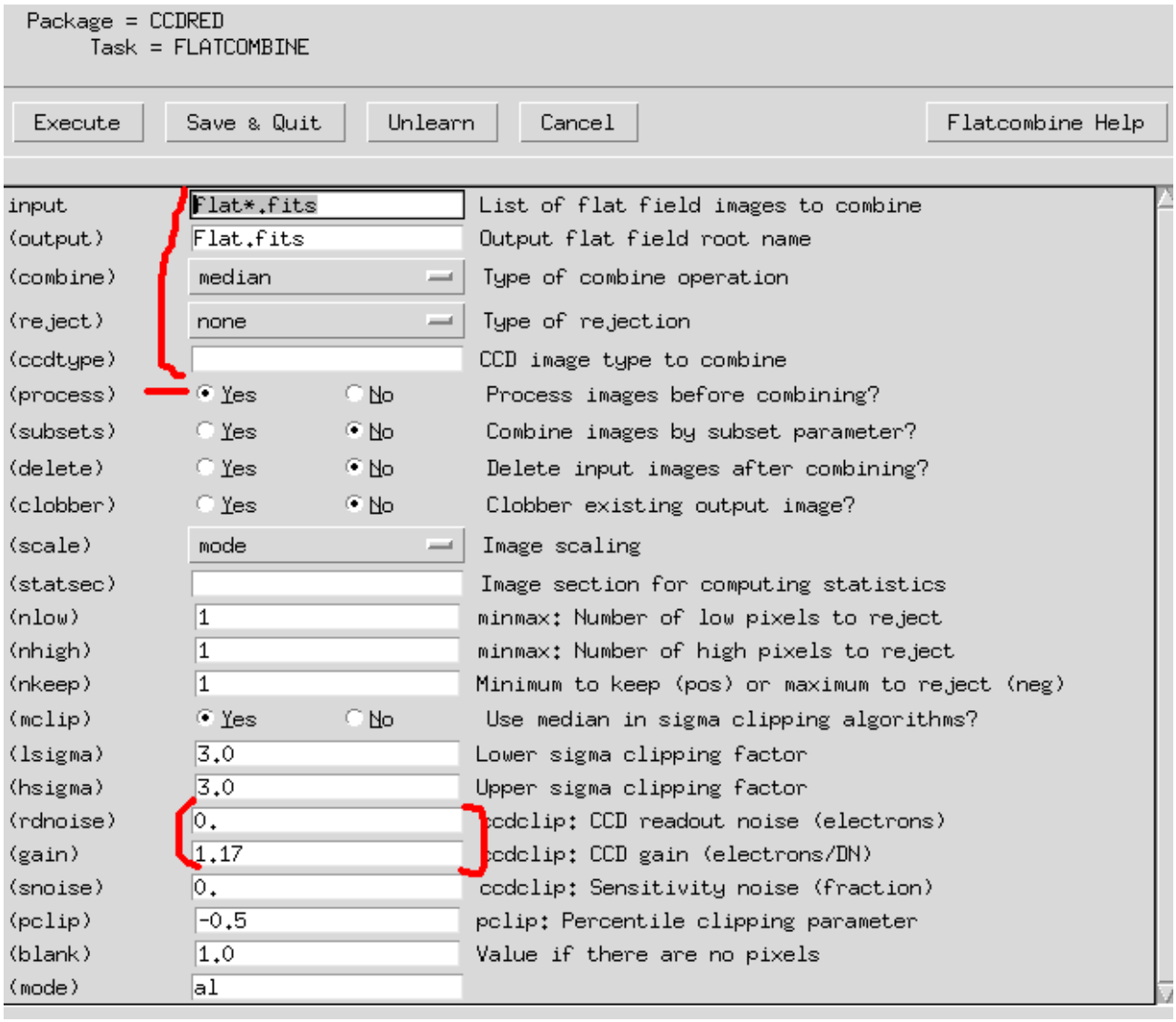

```
--> imstat *
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
           Dark.fits   3214848   -0.8294     11.97     -36.5     13333.
           Zero.fits   3214848     113.6     6.652       87.      6841.
      dark-001_7.fits   3214848   -0.3927     13.14      -46.     13345.
      dark-002_7.fits   3214848    -1.266     13.04      -47.     13322.
```

The zero-corrected dark may have negative values and the mean is around 0.

1-3. FLAT

```
--> cd ../flat/
--> cp ../dark/Zero.fits .
--> cp ../dark/Dark.fits .
--> ls
Dark.fits  flat_30.fits  Zero.fits

--> epar flatcombine
Set parameters as in the figure.
--> flatcombine
```




```
--> !ds9 Flat.fits &
Looks OK

--> imstat Flat.fits
#               IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
            Flat.fits   3214848     360.5     723.4   -18963.     24817.
```

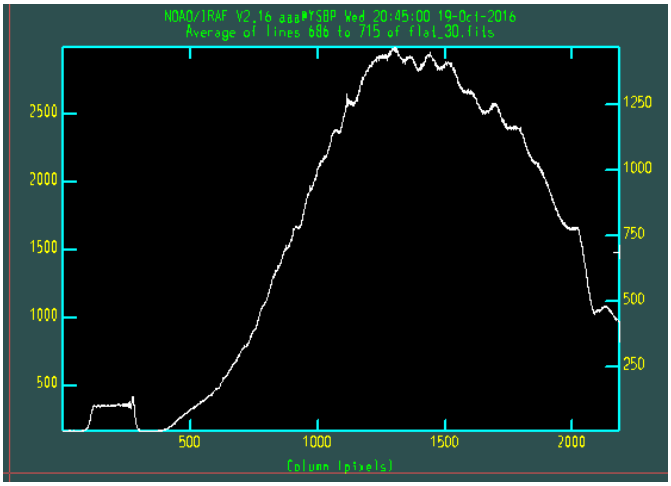
1-4. PRE-PROCESS

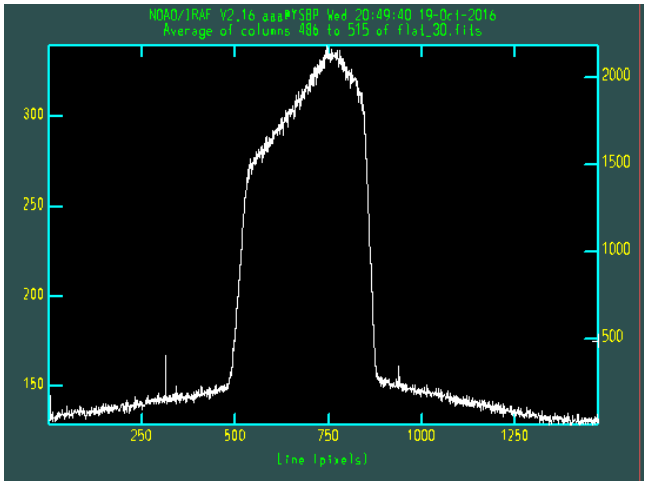
```
--> cd ..
--> cp flat/Flat.fits .
--> cp flat/Dark.fits .
--> cp flat/Flat.fits .
--> implot Flat.fits
```

What we are going to do is to “trim” only the usable part of the image. By doing implot on flat image, you can find some plausible x, y range of the usable region.

Left image: `:a 30` → `:l 700`
Right image: `:a 30` → `:c 500`

On the right y axis, the small yellow tick shows at which column or line you are. Try many different col/line using `:c` and `:l` to get an idea how it works.



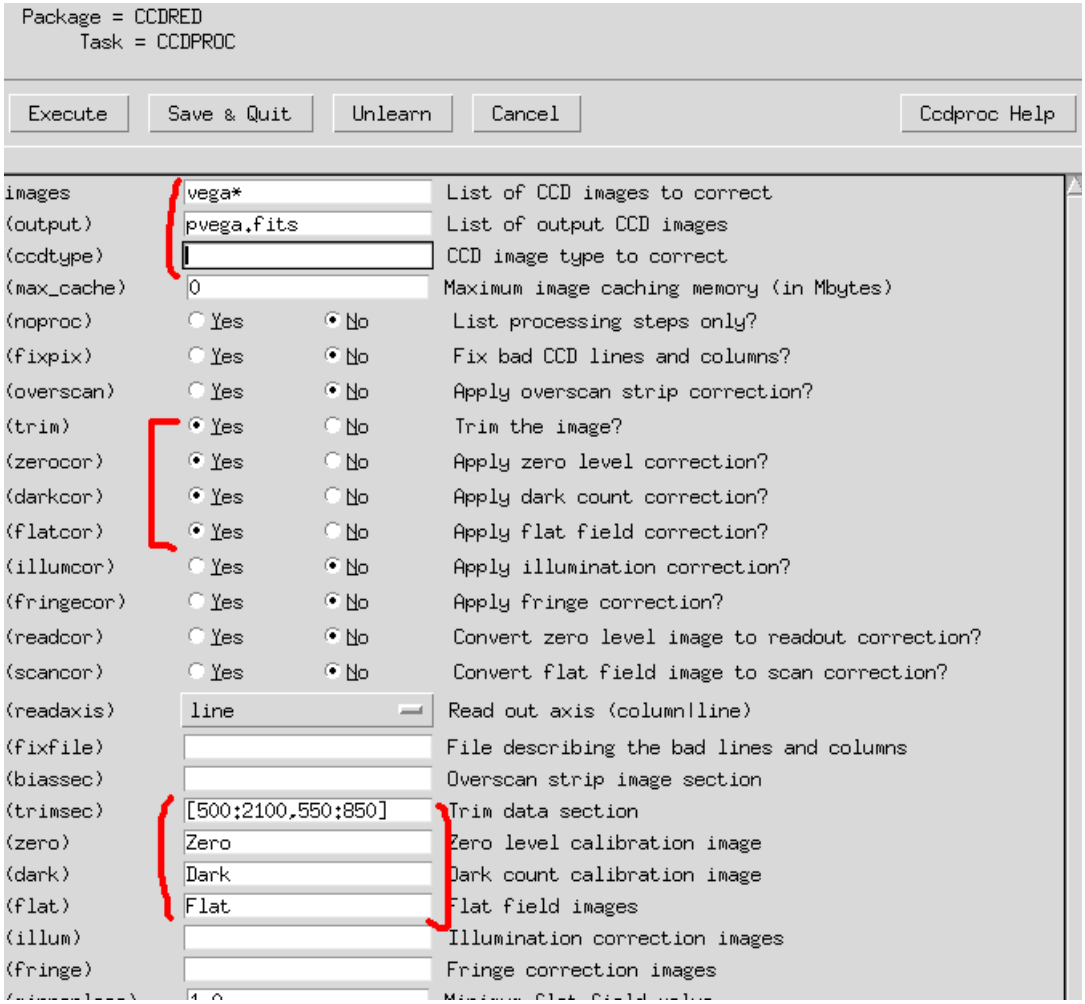


We may say that `x(column): 500~2100` and `y(line): 550~850` is reliable, and other pixels are useless. For smaller x values, there seems to be signal, but it is due to the higher order dispersion and/or due to the order sorting filter.

We can trim all the images (including the object) using the “trimcor” option in CCDPROC. The trim region can be specified at (trim) by `[x1:x2,y1:y2]`, i.e., `[500:2100,550:850]`, in our case.

Now we turn on `zerocor`, `darkcor`, and `flatcor` with `Zero.fits`, `Dark.fits`, and `Flat.fits` as follows:

```
--> epar ccdproc
```
Set parameters as in the figure.



Click on Execute.

```
--> imstat
List of input images ('*.fits'):
#                IMAGE      NPIX      MEAN    STDDEV       MIN       MAX
            Dark.fits    481901   -0.7664     8.627     -33.5     2321.
            Flat.fits    481901     1804.     823.4     73.57    24817.
            Zero.fits    481901     113.3     5.251       90.      829.
           pvega.fits    481901     168.3     2541.    -2224.   133972.
         vega_7s.fits    481901     220.9     1254.       83.    35427.

--> ccdlist
CCD images to listed ('*.fits'):
Dark.fits[1601,301][real][unknown][][TZ]:dark
Flat.fits[1601,301][real][unknown][][TZD]:
Zero.fits[1601,301][real][unknown][][T]:
pvega.fits[1601,301][real][unknown][][TZDF]:
vega_7s.fits[2184,1472][ushort][unknown][]:
```

2. Wavelength Calibration

I want to make a separate directory to start from a clean directory. It is totally up to you.

```
--> !mkdir after_prep
--> cd after_prep/
--> !mv ../pvega.fits .
```

2-1. DISPAXIS

The dispersion axis should be saved in the header for the IRAF to work properly. We can manually correct this information:

check whether the header containing the information:

```
--> imhead pvega.fits long+ | grep DISPAXIS
```

edit header:

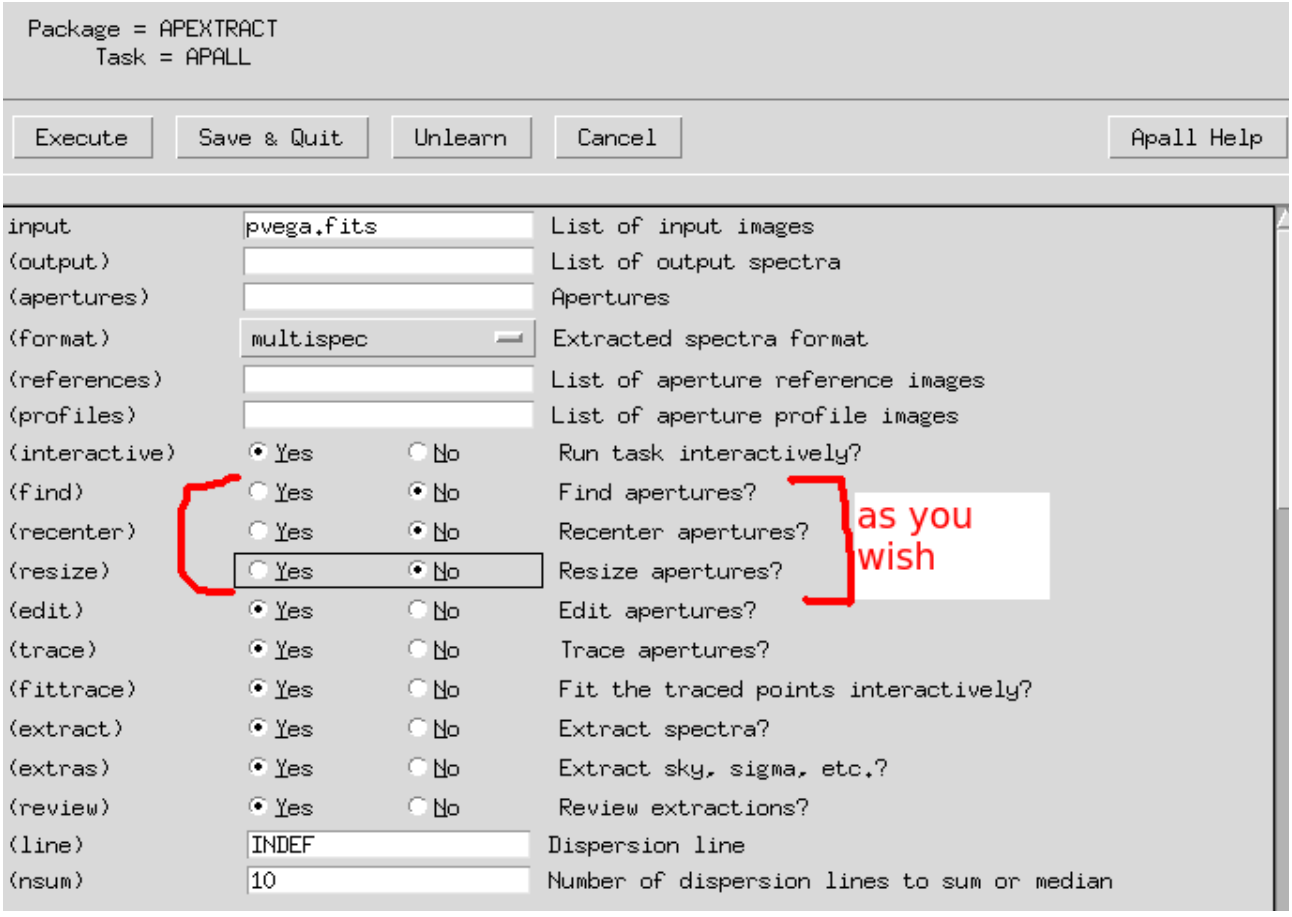
```
--> hedit pvega.fits DISPAXIS 1 add+
add vega.fits,DISPAXIS = 1
update vega.fits ? (yes):
vega.fits updated
--> imhead pvega.fits long+ | grep DISPAXIS
DISPAXIS=                    1
```

Now the image has the DISPAXIS header keyword.

2-2. APALL

```
--> noao
--> onedspec
--> twodspec
twodspec/:
 apextract/      longslit/
--> apextract
apextract/:
 apall           apedit          apflatten       apnormalize     apscatter
 apdefault@      apfind          apmask          aprecenter      apsum
 apdemos/        apfit           apnoise         apresize        aptrace
--> epar apall
```

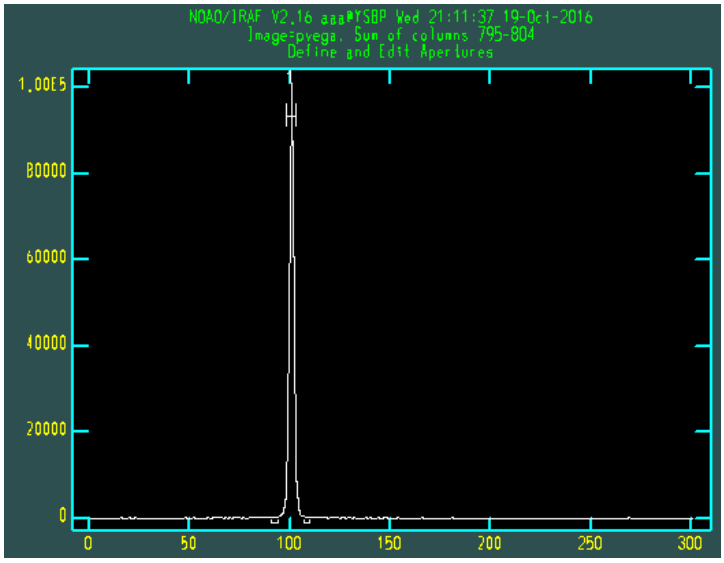
Set parameters as in the figure. If you want the IRAF to automatically find aperture(you can modify it later by yourself, but you can get an idea how it finds the aperture), you can just set everything as default.



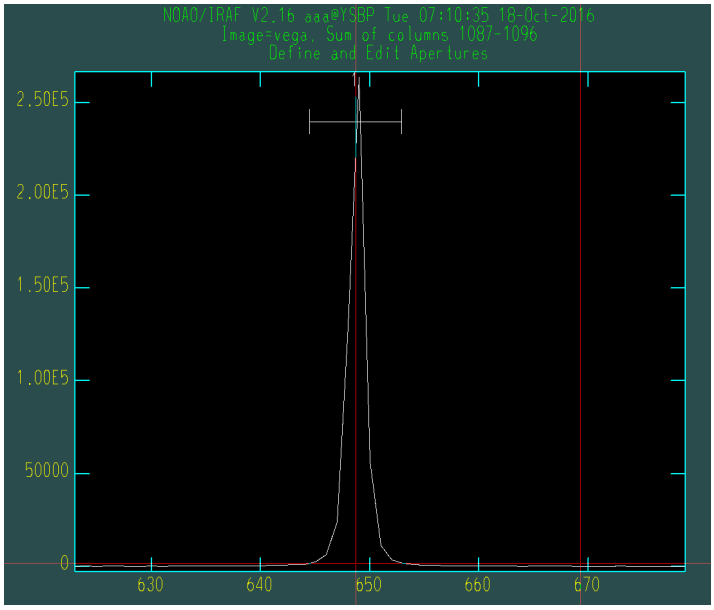

```
--> epar apextract
Set dispaxis = 1

--> apall
List of input images ('vega.fits'):
Edit apertures for vega? ('yes'):
```

If you hit '?', the command will show you the help page. Hit '?' again, and you may be able to get out of the help page.



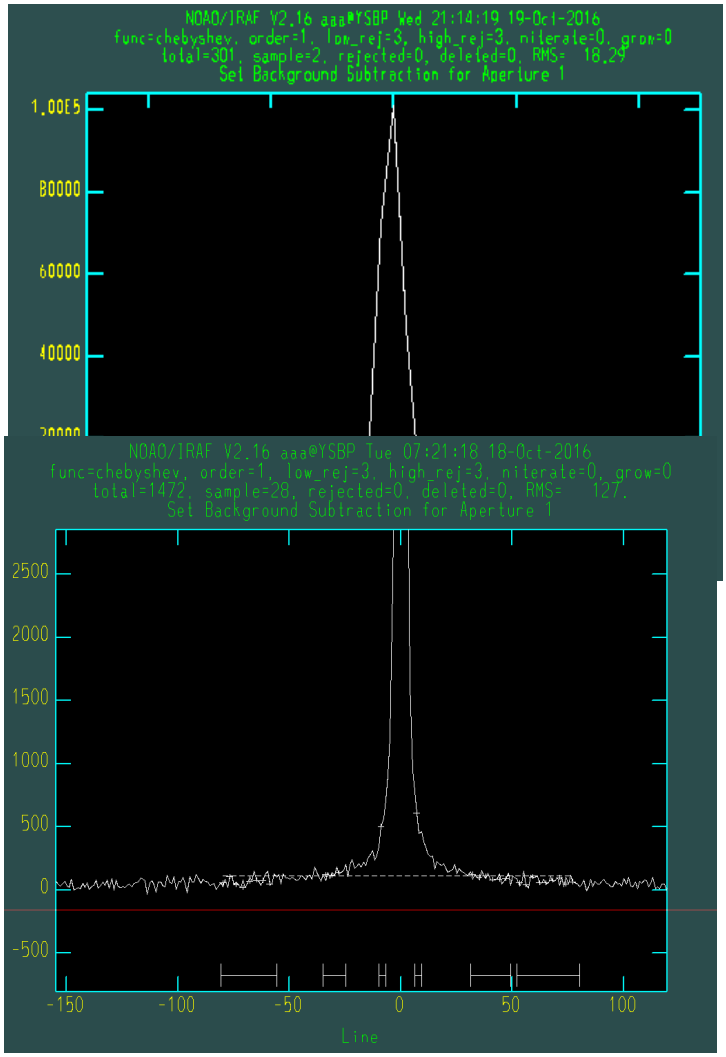
Hit `w` for zoom, and `e` for the lower left and upper right corner to zoom in. hit `w a` to get to the default zoom.



(This screenshot may slightly differ from yours, but the general trand is the same)

Type m after you put the cursor near the peak, and IRAF will automatically find aperture for the profile. Hit `d` to delete the recent ones. Hit `y` key so that you can define the aperture size as the width at the cursor.

Hit `b` → We now will fit the background. This is needed to subtract background from the source. Hit `w a` to go to the original zoom. Then do `w e e` to zoom. It is better to zoom small y range to see the background fluctuation as indicated in the figure (red marks).



The dashed line = fitted sky (background). As you can see, it is too high!

We need to adjust the sky sampling region. Hit `s` at the lower/upper for both sides (total 4 hits). Hit `f` to fit the sky.

Hit `z` key to undo, if you want.
(This screenshot may slightly differ from yours, but the general trand is the same)

Now it seems better…

Hit `q` to get out.

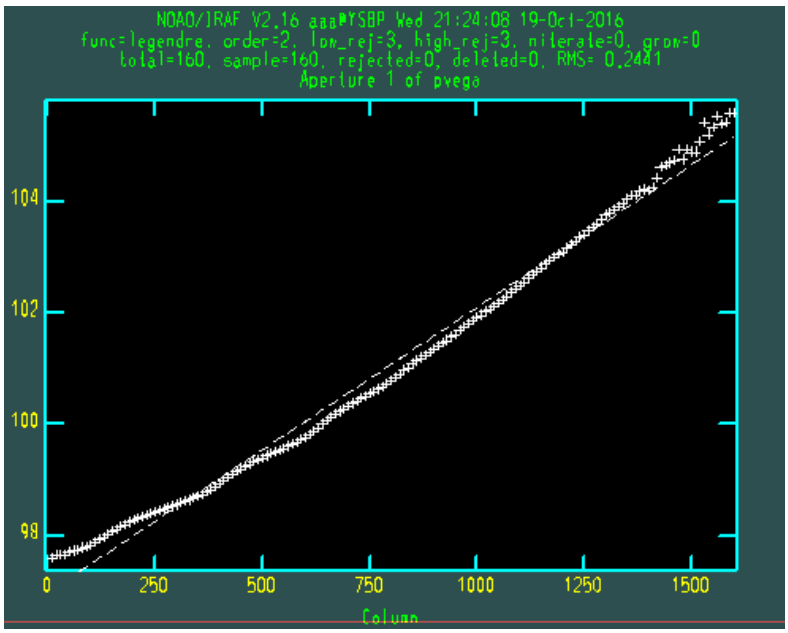
Hit `q` again.

On the terminal, you will see some questions. Type "`yes`" to the GRAPHICS, NOT THE TERMINAL!

You may have to answer 2-3 times, depending on the APALL `epar` setting.

Type return again and again until you see the following figure:

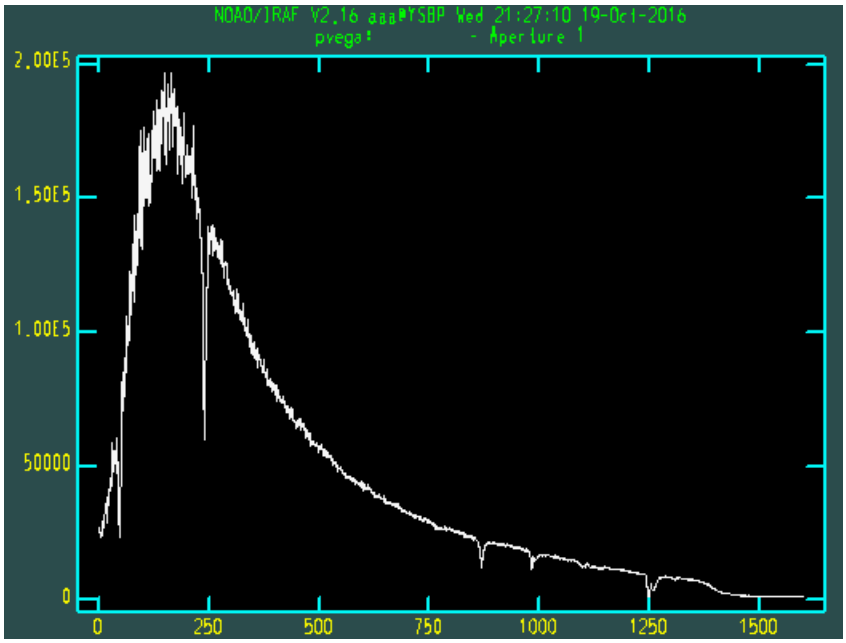
The fitting does not look good, but if you see closely, the error is only few pixels order. But you may still want better fit.



Type `:order` *n* to fit the n-th order fitting line to the + signs. `:order 10` or `5` will be enough for most of the cases.

Hit `f` to see newly fitted dashed line.

* Moving along columns, the center of the aperture should move, due to the imperfection of instruments. This "shift" is called the trace. The above image shows this "*trace*" value along the columns(x-axis).



Now you get this spectrum of Vega.

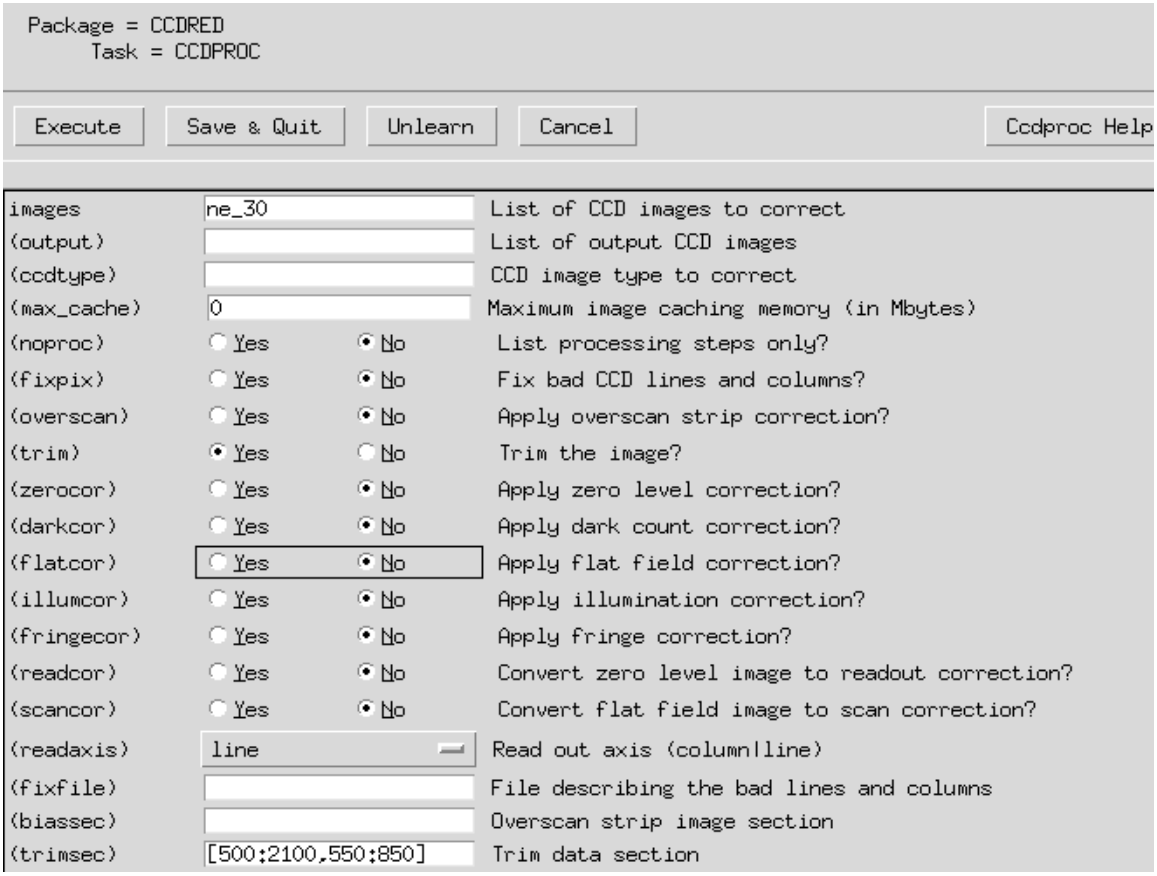
What is remaining?

We have to change the x value to wavelength.

2-3. IDENTIFY

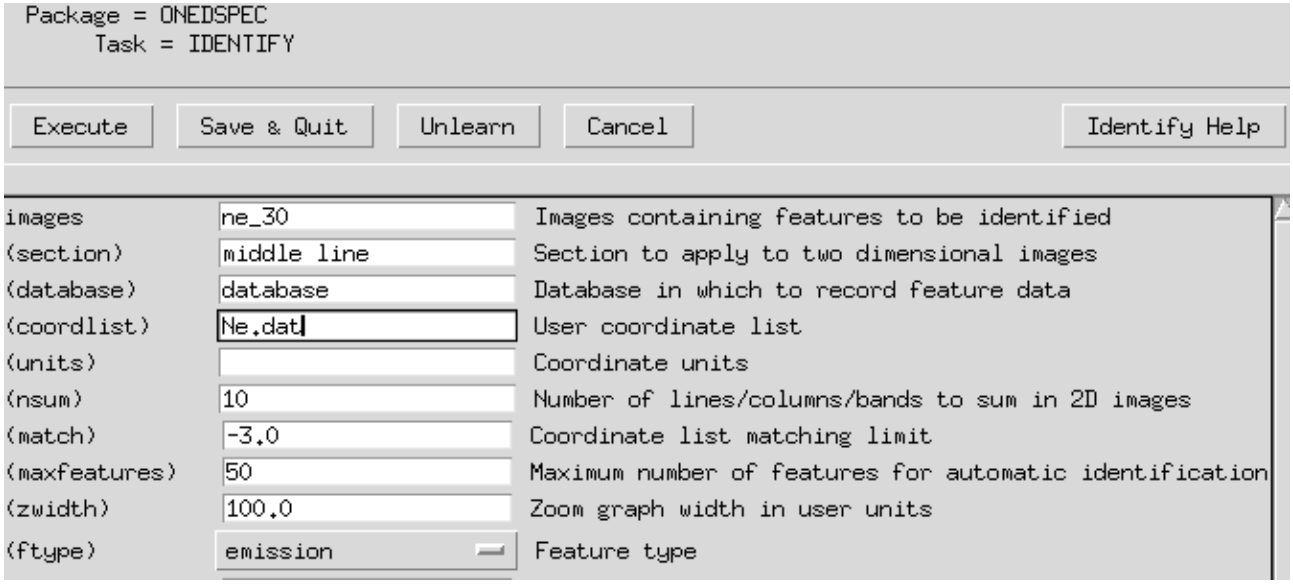
First let's trim the neon lamp file. Turn only `trim` on:

```
--> epar ccdproc
```



Then we have trimmed image for the lamp as any other images.

```
--> epar identify
```
(or `identify ne_30.fits coordlist=Ne.dat`)

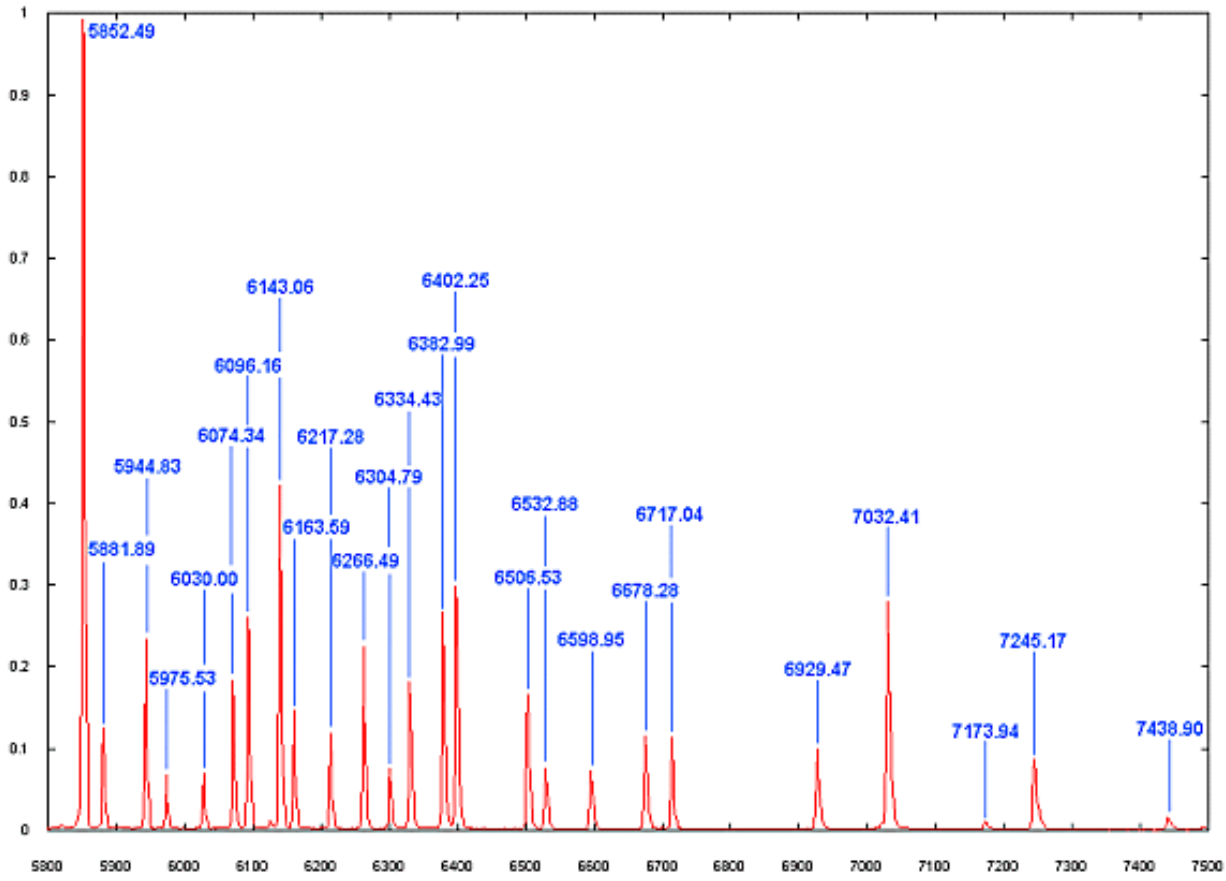


(These screenshots may slightly differ from yours, but the general trand is the same)

This shows the Ne lamp image's spectrum.

The figure at the next page is the Ne spectrum. The numbers are in Angstrom unit.

You can see that the general features are the same as the IRAF (PyRAF) graphics.

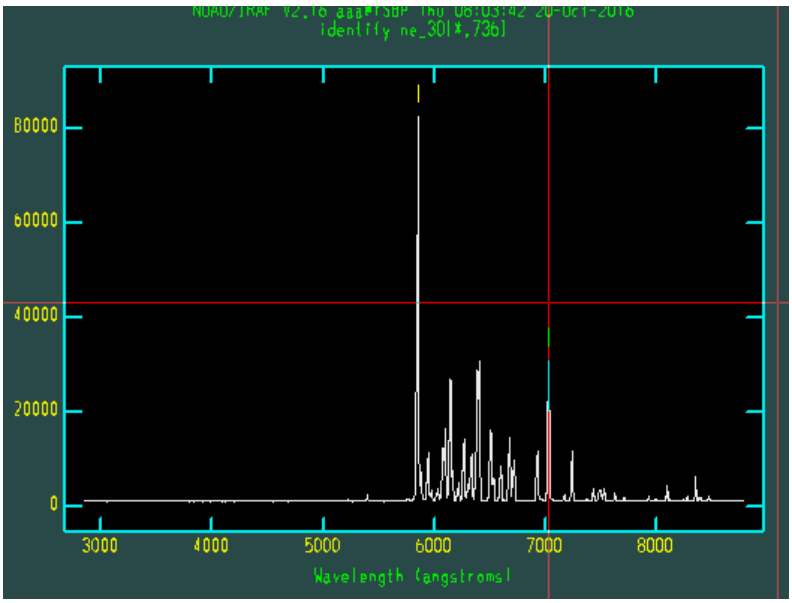


http://www.astrosurf.com/buil/us/spe2/hresol4.htm
(If the image is low resolution, you may find "low resolution lamp spectrum" more similar to your image, such as that in the following link:
http://www.astrosurf.com/buil/us/spectro8/spaude5_us.htm)

Now we have to let IRAF know what the wavelengths of about 2~3 peaks in the graphics. IRAF then automatically use the "peaks" data file to find all other peaks. Let's see how it works:

Use `w e e` and `w a` to zoom the spectrum. Put the cursor near one of the peaks (e.g., the largest peak at 5852.49 Angstrom). Then type `m` (for mark). At the bottom of the graphics, type the rough wavelength value (e.g., "`5852`"). Hit return. At the bottom, there will be the found wavelength value. It is found from the `Ne.dat` file, using your input value.

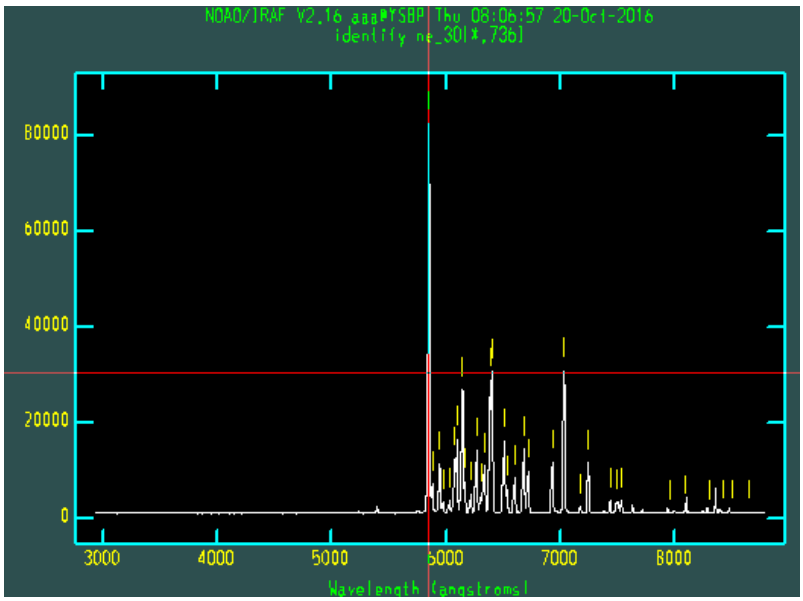
Do the same thing about once more. (total two is enough since we have `Ne.dat` file)



Hit `f` will show you how well the fitting will be done. If it seems to have small errors, you are satisfied and hit `f` again. If you are not satisfied, do the marking again after hitting `f`.

Hit `l`. IRAF will automatically read the `Ne.dat` file (containing all possible line centers) and find all peak values. But this process is not always accurate, so you have to edit them.

After fitting, you can see the x axis is now in wavelength unit.



After hitting `l`, you may see a lot of identified lines as in the figure. The lines, however, are not always the real ones, since very small emission peaks are in the `Ne.dat` file, too.

You have to delete few of them, and re-fit few of them.

You can see the following link for help:
http://stsdas.stsci.edu/cgi-bin/gethelp.cgi?identify

Some are summarized below:

d (D)elete the feature nearest the cursor.
n Move the cursor or zoom window to the (n)ext feature (same as +).
p (P)an to the original window after (z)ooming on a feature.
z (Z)oom on the feature nearest the cursor. The width of the zoom window is determined by the parameter zwidth .
+ Move the cursor or zoom window to the (n)ext feature.

Instead of `p` and `z`, you can of course use `w e e` and `w a` as usual.

If you feel satisfied, hit `f` to see the residual plot. If there is any point with significant residual, hit `d` near it to delete it (X mark will appear). Hit `f` again and you will fit again. Hit `q` twice to quit identify.

2-4. Header and Dispersion Correction

The `ne_30.fits` now include the information of our calibration. We now have to implement the object image file that "*the image fits file has reference spectrum data as this neon fits file!*" To do so:

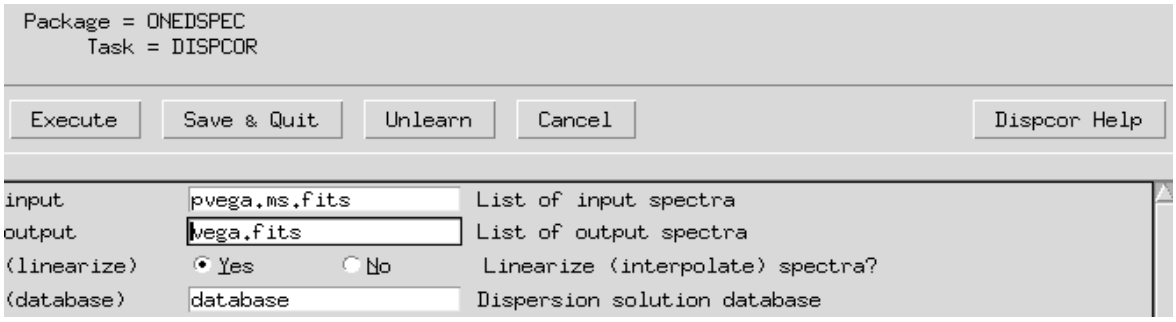
```
--> hedit pvega.ms REFSPEC1 "ne_30.fits" add+
add pvega.ms,REFSPEC1 = ne_30.fits
update pvega.ms ? (yes):
pvega.ms updated
```

For check:

```
--> imhead pvega.ms long+ | grep REFSPEC1
REFSPEC1= 'ne_30.fits'
```

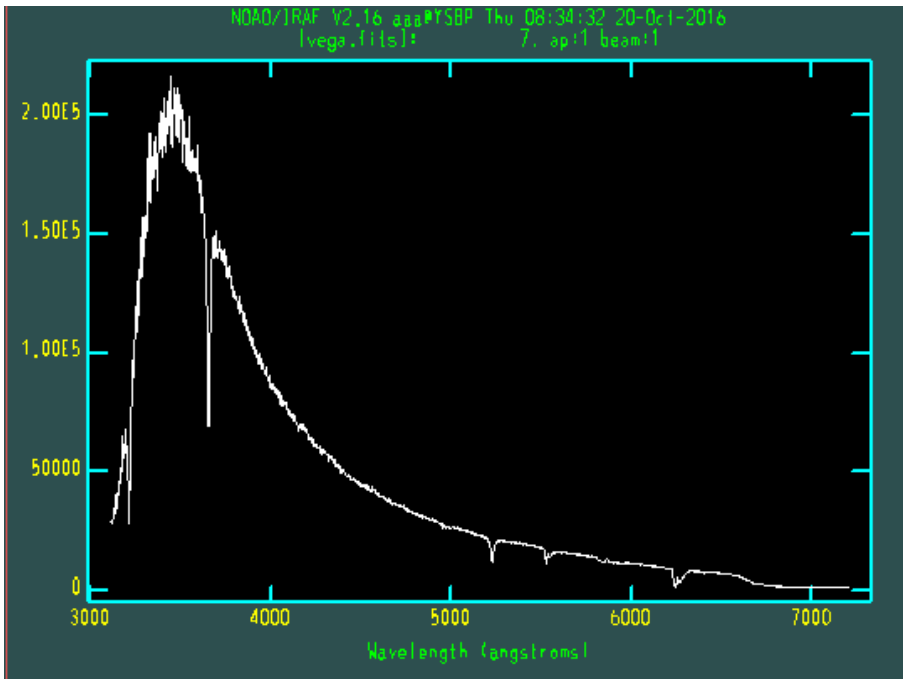
So the IRAF now will be able to change the x-pixels into wavelengths based on this ne lamp fits data. This can be done by dispersion correction:

```
--> epar dispcor
```



The splot task will show us the calibrated plot:

```
--> splot vega.fits
```



Hit `space`: you will see x, y, z values. Y is the cursor's y-value and z is the actual y value of the plot.

You can use `w e e` and `w a` to zoom in/out.

2-5. Deblending

To study about a line (absorption, in our case), zoom into a certain region using `w e e`. We will fit the continuum, and fit the absorption line.

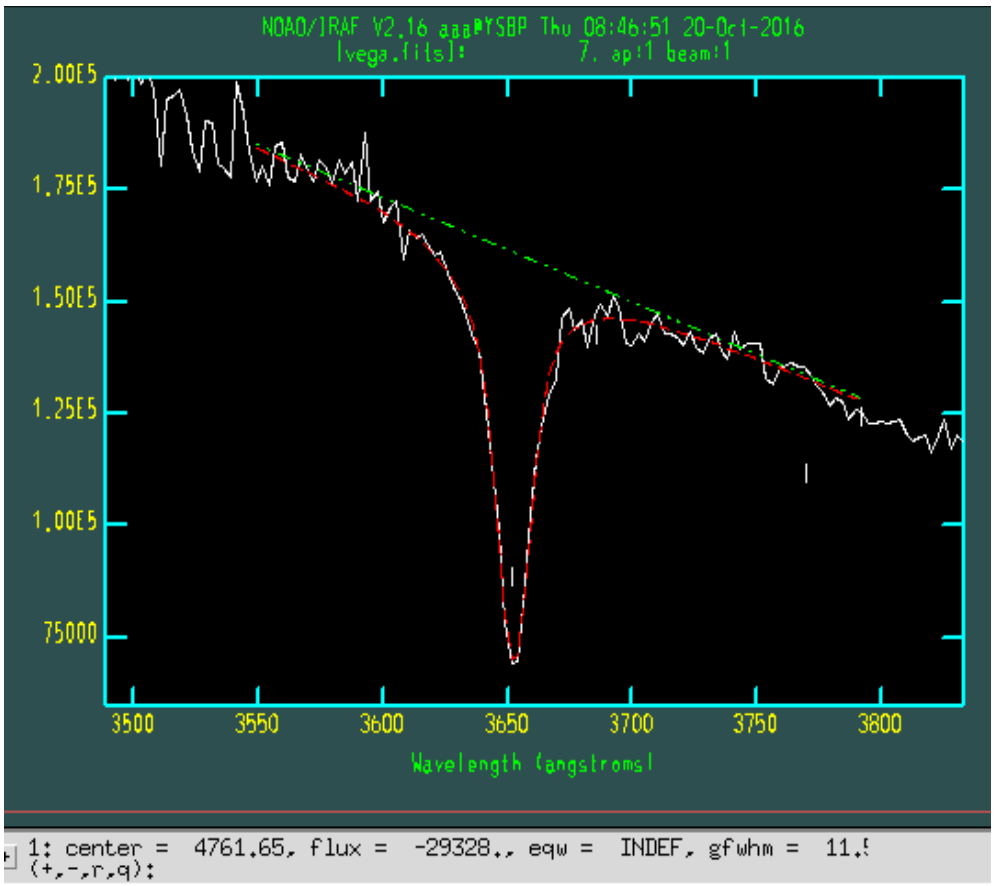
At the left- and right-most positions you want to fit the continuum, hit `d` at each edges.

At the bottom of the graphics, you will be asked about "Lines". This means with which profile you want to fit the absorption. I will select Gaussian. Put the cursor near the absorption line, and hit `g`. There will appear a small vertical tick which indicates the center you selected.

After you selected all the lines you want to calculate within the zoomed window, hit `q`. Another question is Fit positions: just type `a`. The next is Fit Gaussian width question, just type `a`. Fit background? Oh yes: type `y`. Then you will see a green continuum fitting line and red fitting line (absorption fitting line).

You can see the center, flux, equivalent width, and gaussian FWHM value at the bottom of the graphics. `+/-` will show you the results at next/previous peaks. `r` will show you the RMS of the background after fitting.

Hit `q` to exit. The same question will be asked for if you have other things to fit. Just hit `q` to quit completely. You will see "`Deblending Done`" message. The process we've done is called *deblending*, since we "deblended" line from the continuum.


NOAO/IRAF V2.16 aaa@YSBP Thu 08:46:51 20-Oct-2016
[vega.fits]: 7. ap:1 beam:1
2.00E5
1.75E5
1.50E5
1.25E5
1.00E5
75000
3500
3550
3600
3650
3700
3750
3800
Wavelength (angstroms)
1: center = 4761.65, flux = -29328., eqw = INDEF, gfwhm = 11.
(+,-,r,q):