CCD-Lab Clarifications and IDL pointers

0.1 Noise

In the current context noise is taken as the **standard deviation**, σ_{Image}, of an image (or a series of images combined into one).

As mentioned in the lab-text (Exercise 8), for each doubling of the sample size - which in the first case is going from one bias frame to two combined bias frames, one expects the noise/standard deviation to increase by the square-root of two. This can quite easily be shown by the addition rules of **variance** of a sample that is combined with another uncorrelated sample (the variance doubles, which implies the standard deviation increases by the square-root of two). If you show this (google addition of variances), this would be nice to add in the report for example.

Later however you are asked to show that the ”noise decreases as you expect” (Exercise 12). Actually, the standard-deviation of the combined flat images will **increase** (ideally by a factor of square-root of two) for each pair of images added into the composite-flat. However - the normalized noise/standard-deviation per image-pairs will decrease. This means the relevant measure of noise given the combination of images is the standard-deviation of the given composite-flat divided by the number of pairs used for the composite. The normalization is necessary as the combining of images will in effect ”smooth out” the noise, but also amplify some noise-peaks if there is "constructive interference".

For Exercise 12 this means that you can both plot the standard-deviation of the combined flats, together with the theoretical expectation that it should increase by $\sqrt{n_{\text{pairs}}}$ (using the standard deviation of the composite using one pair as a starting reference). You can then plot that the **normalized** noise $= \sigma_{\text{Composite Flat}}/n$ (with $n =$ nr. of pairs used), should decrease by the theoretial prediction $\sqrt{n}/n = 1/\sqrt{n}$ (expected rise of standard-deviation normalised over n-pairs).

0.2 Readout noise

The readout noise in electrons, as asked for in Exercise 11 is simply R.O.N. $= g{\cdot}\sigma_{bias}$, where bias stands for one of the bias frames (which ideally should have the same noise).

0.3 Central-region confusion in the CCD-Lab exercises

Where not stated otherwise, computations are done for the **central region** of images, as defined in the lab text. For the last correction in Exercise 13 you can do the correction for the central region only, or for both the central region and the entire FOV if you want.

0.4 Clarified image correction procedure

The corrected image is found by the following steps

- Find the average flat-field image, $F_{\text{average}} = \dfrac{(F_1 + F_2 + \quad \dots \quad + F_{16})}{16}$.

- Find the average dark-image with the same exposure time as the **flat images**, $D_{\text{F,average}} = \dfrac{(D_{\text{F},1} + D_{\text{F},2} + \quad \dots \quad + D_{\text{F},5})}{5}$.

- Find the master flat-image, $F_{\text{master}} = F_{\text{average}} - D_{F,\text{average}}$

- Find the normalized master flat-image by its scalar mean $\overline{F}_{\text{master}}$ by $F_{\text{norm. master}} = \dfrac{F_{\text{master}}}{\overline{F}_{\text{master}}}$

- Correct the raw image, this time using the dark-image average of the dark-images with the same exposure time as the **science image**, with

  $D_{\text{I,raw,average}} = \dfrac{(D_{\text{I,raw},1} + D_{\text{I,raw},2} + \quad \dots \quad + D_{\text{I,raw},5})}{5}$

  and the correction then being $I_{\text{corrected}} = \dfrac{I_{\text{raw}} - D_{\text{I,raw}}}{F_{\text{norm. master}}}$

The final corrected image is probably not going to look all that different - the process is the important part, as well as discussing why the image may not change all that much.

0.5 Getting indice for min/max values of a matrix in IDL

The min/max functions do not output a 2-D index for the given extremum, they do however output a 1-D index which can be converted to a standard 2-D index in (x,y) using the function *array_indices*. An example snippet is given below.

```
my_matrix = [ [1, 2, 3], [4, 5, 6]] ; An example matrix

; Finding max value of my_matrix with its 1-D index saved in location
max_val_of_matrix = max(my_matrix, location)

; finding the 2-D index of the max value using array_indices and the variable
; location
max_index = array_indices(my_matrix, location)
```

The function *array_indices* is also documented here.

0.6 Making histograms and saving plots as .eps/.pdf files in IDL

Histograms can be generated using the *histogram* function, with documentation here, but a simple snippet example for making a histogram can be found here as well, which looks like below.

```
; A very simple example of a histogram

a = [1, 1, 3, 1, 3, 2, 5, 4]              ; A very simple array for the histogram
hist = histogram(a, locations = values)  ; Creating histogram counts in hist,
                                          ; and hist values in values via locations
plot, values, hist                        ; Plotting the histogram
```

For saving plot and not images as nicer .eps/.pdf files, I have created a sample batch file which you can find here. It details how you can save a plot in the .eps format and then convert it to .pdf for inclusion in the report. It looks like detailed below.

```
; This is a snippet example for saving plots as .eps/.pdf files.

; Example vectors:

x = [1,2,3];
y = x;

xsize = 30       ; Set window size here or below in device, try around!
ysize = 20
set_plot, 'ps'   ; Setting window to ps for output
;!p.font = 0     ; These Parameters set fonts and thickness of font/axes
;!p.thick = 8    ; try these out!
;!x.thick = 5
;!y.thick = 5

device, filename = 'your_filename.eps' , encapsulated = 1, /helvetica

; This is your standard window, but it won't pop up, only write to file.
device, xsize = xsize, ysize=ysize

; Your plot command goes here as normal:
plot, x,y, title = 'A plot!', xtitle = 'An x-axis!', ytitle = 'A y-axis!'

device, /close   ; Closing device

set_plot, 'x'    ; Resetting so that new plots will be displayed only again

!p.font = -1     ; Resetting font

; Converting the generated .eps file to .pdf with same name, by executing the
; Linux program epstopdf in the command line via the $ symbol in IDL

$epstopdf your_filename.eps
```

Take note that that the plot as saved above will **not** be displayed on screen, as it is only output to file. So as when you saved your images, you will have to make sure the plots look as they are supposed to manually, and plot them normally beforehand to view them, and then copy your desired plots into something like the arrangement above for getting the output. And yes - IDL can be a hassle when it comes to displaying/outputting files!