Project 1 – Gain and noise of a CCD camera

Due October 4, 2012

ASTR310 Fall 2012

1 Introduction

In this project you will measure the gain G and read-out noise σ_B for the CCD camera in a lab setup at the Observatory. This document describes the practical part of the project.

The electronics associated with a CCD typically include clocking circuits to move the charge in each pixel over to a shift register, the shift register which reads out the pixels one by one, an analog charge-to-voltage amplifier (both the shift register and the analog amplifier are usually part of the CCD chip itself), and an analog-to-digital converter. The original charge packet consists, ideally, of one electron per detected incident photon. These are subject to standard counting (Poisson) statistics. The amplifier introduces some additional noise, which may vary from pixel to pixel by a very small amount but which is generally independent of pixel and of signal level. The amplifier and A/D converter also introduce offsets of the zero-point in the signal. Proper use of the CCD for later observing projects requires an understanding of the noise in the data.

There are basically two parameters that must be measured. The first is the effective gain, G, of the amplifier plus A/D system. In other words, we want to know the number of A/D output units produced per initial photo-electron. This gain is usually expressed as the reciprocal gain, $K = 1/G$, the number of electrons per digital count of output. The other parameter is the amount of read-noise introduced by the output amplifier, σ_B. This is usually measured in A/D units and then expressed in terms of the number of equivalent electrons.

These concepts were covered in class and in the class notes on CCD theory. If you do not have a copy of that handout download a copy from the class web pages or ask the instructor.

You will use measurements of a step wedge to provide signals covering a well-defined range of intensities to determine these parameters, using the theory of statistical fluctuations to determine the number of electrons.

2 Getting the data

2.1 Preparing the Equipment

After the new Apogee CCD camera was installed on the 20-inch telescope, we moved its old Photometrics CCD camera and electronics to the observatory central bay. We have a crude improvised set-up so that you can carry out this exercise using that old CCD camera.

- Keep a log of everything you do and when you do it. Your log will be helpful in writing the Observations section of your report.
- The CCD cooling system should already be on and stabilized. Turn on the rest of the CCD system, if it is not already on, using the separate instructions.
- The step wedge and lamp should be inside the left-hand end of the cardboard tube on the table. The electronics cart will be set up nearby. Locate the battery power-pack and plug in the lamp. Turn it up so you can see the glow of the step wedge.
- The transfer lens is mounted on an aluminum plate, which is fastened by two thumb screws to the face of the CCD camera, which is at the other end of the cardboard tube.
- With the flip mirror in place, look in the CCD eyepiece and verify that you can see the wedge. You should turn off all the room light – if you leave the door open there will be enough light from outside to see what you are doing. The wedge should be centered and well-focused – if it is not, alert your TA. If you look carefully, you will see a rectangle in the eyepiece field of view, which corresponds to the CCD chip. A centered and reasonably well-focused image is very important.
- Flip the mirror out of the optical path, turn down the lamp brightness, and take some practice images of the step wedge. Adjust the brightness of the lamp and the exposure time to produce a good image.

2.2 Making the Measurements

Be sure the lights are all out. Use the cloth shroud to shield the gap between the tube and the CCD lens.

When you are sure that the CCD has reached equilibrium, you can begin taking data. First readjust the brightness of the lamp and the exposure time for optimum exposures. *Ideally this means digital counts of about 10,000 at the brightest portions of the step wedge with a reasonably short exposure, but not less than about 3 seconds.* The counts in the background (the part of the image away from the wedge, which should be dark) should be several hundred. See the operations manual for the necessary commands to check the counts.

Once all these parameters have been adjusted, your data consist of three images, taken in quick succession. The first and last images are of the step wedge and the middle one is a bias image. To take them in quick succession, use different image caches in memory. *You can stack the commands for all exposures and cache changes on one line.* It is important that nothing changes between the two images of the wedge! If the equipment moves or if the intensity of the light varies, even by a small amount, this will make analysis of the data impossible. It may be worth taking a second set of images to make sure you have one good set. Be careful to note the image caches, exposure times, and start times of each exposure in your log.

Now you should save your images to the hard disk. After you have written them, you can then exit the CCD program by typing "quit".

Next you need to get the images from the Observatory to the Astronomy Department. The first step is to copy the images to a floppy disk. The TA has several floppies on hand for just this purpose. Use the DOS command **copy**:

copy C:\`filename.img` **A:**\

Hint: You can use wildcards in the commands, such as `fi*.img` to copy all .img files starting with `fi`. You may want to leave your images on the hard disk at the observatory, just in case you have a problem with the transfer, but we will erase them all at some time in the future. If you are back at the observatory after you are sure that your data are safely on the network in the Astronomy Department, please delete these images from the hard disk.

The second step is to take the floppy to the newer computers at the Observatory to transfer the images from the floppy disk to a flash drive. Try to have at least one flash drive within each group.

You can work as a group to learn the commands for the CCD program, but you should each take your own individual images for analysis.

Unplug the lamp from the battery pack when everyone in your group finishes taking their images to preserve the battery power.

3 Analyzing the data

3.1 Transferring data to the Astronomy computers

The analysis is done on the PCs in the computer lab in the Astronomy Department on campus. Once you log on to your account, you will want to transfer the images from your flash drive to your directory.

If the PCs are booted into Windows, choose the restart option from the start menu. Watch as the computer reboots, as you need to quickly choose the CentOS 5 option when the screen pops up. If you are not fast enough, the computer will default to Windows and you'll have to wait for it to finish and then restart again.

Once you see the `lab0##` login box, use the username and password given to you. The password is initially set to your student ID number, but you can change it if you choose.

Either select the terminal icon from the top command bar or right-click anywhere on the desktop and select "open in terminal" to get a terminal window with a command prompt.

Plug your flash drive into the USB ports on the side of the monitor or on the front of the computer.

Next make a new directory in your home directory where you would like to place your data, using the **mkdir** command at the command prompt:

mkdir `dirname`

where `dirname` is whatever name you choose (something like `lab1` would be sensible).

Next, read the data in from the flash drive. Start by changing your directory by typing:

cd /media

and list **ls** the contents of the media directory to find your flash drive with:

ls

Then change directory **cd** into your flash drive by typing:

cd `flashdrive`

where `flashdrive` is the name of your drive, and copy your files to the directory you created, using **cp**:

cp `filename.img` **/home/**`yourusername/dirname`

for each file (or use the * wildcard, or **cp -r** `name` to cp an entire directory at once). When you have copied the files, cd into the directory to which you copied your files and type:

umount /media/`flashdrive`

to unmount the flash drive. Alternatively, you can double click on the flash drive's icon on the desktop, and choose "Unmount volume" from the "File" menu. The drive will only unmount if it's not in use, which is why you should cd out of the directory first. Now you can safely remove the flash drive from the USB port.

3.2 MATLAB procedures

Now that you have the .img files in your directory, you will want to do something with them. The first step is to start MATLAB so you can run a function to change the `filename.img` files (a special format valid only for this camera) into `filename.fits` files (FITS is a standard astronomical format). Start MATLAB by typing:

matlab2009

Next, include in the internal MATLAB search path the directory containing a series of useful functions written for ASTR310 by typing the command at the `>>` prompt in the MATLAB command window (don't forget the single quotes and the twiddle):

addpath('~harris/a310')

Then, to convert the file `filename.img` type (still within MATLAB):

img2fits('`filename.img`**')**

You will find a new file with the name `filename.fits` in your directory. Do this for each file you need to convert, choosing a new name for each file to keep from overwriting.

Make a note of the names of your `.fits` files: the two exposures of the step wedge and the bias frame. Read in the images by using the **rfits** (read FITS) procedure. If your file were called `mystep1.fits`, you could type (don't forget the single quotes to keep this a name rather than a variable):

s1=rfits('`mystep1.fits`**')**

A FITS file has both header information and data, so the variable `s1` is a data structure, with the variable named `s1.data` containing the image, and other components of the `s1` structure containing header information. To look at the image, you may type

imagesc(`s1.data`**)**

No quotes here; `s1.data` is a variable.

To display an indication of what values are associated with the different colors use **colorbar**. To change the color map, look at help for the **colormap** function. For a list of predefined colormaps (among other things), type **help graph3d**. For example, try **colormap gray**. The standard colormap is recovered with **colormap jet**. To set the values corresponding to the extremes of the color range use **caxis([lowest highest])**.

For hardcopies, first produce postscript files with the **print** command within MATLAB (look it up in the introduction to MATLAB documentation), then print them out from your terminal window with the **lp** UNIX command. For example: within MATLAB you can issue **print -depsc2** `nicefig.eps` to produce an encapsulated postscript file containing the graphics in this graphics window, `figure 1`. See **help print** if you want to write the image from a different figure window. Then, in the terminal window, issue a **lp** `nicefig.eps` to produce a hardcopy in the lab printer. Similarly, **print -djpeg**`nicefig.jpg` writes a jpeg file.

3.3 Image analysis

Now, suppose you have the image displayed. You can examine the values of the pixels at any point of the image by selecting the "data cursor" tool in the graphical window tool bar. Once the "data cursor" is selected, clicking anywhere in the image will display in a small rectangular region the x,y coordinates of the point selected (corresponding to the column and row of the `s1.data` matrix), the value in that cell (which will be called "index"), and the corresponding RGB values to which it is mapped in the current colormap. Another useful tool is the "zoom in" tool. Select it, then click and drag the cursor to select the region into which to zoom. To unzoom, use the "zoom out" tool.

The mean value of the bias image is the mean offset in the zero point introduced by the amplifier and A/D system. The noise of the amplifier produces random scatter in the value of the bias from one pixel to another. If your bias image is called "bias", then the mean value of the image can be obtained by typing **mean(bias.data(:))** — **mean()** is a MATLAB function that sums all the elements of an an array and then divides the total by the number of elements. You can compute the root mean-square-value of the bias image (the read out noise) by first subtracting the mean bias value from each pixel value (i.e., **del=bias.data - mean(bias.data(:))**) to get the fluctuations about the mean, Δ_B, and then taking the square root of the mean of the square of Δ_B: **sqrt(mean(del(:).^2))**. MATLAB has a built-in function that computes the root-mean-square value (also called the standard deviation, since it is equivalent to that parameter for a Gaussian distribution): **std(bias.data(:))**.

To investigate the Poisson noise due to the photon statistics of the illuminated CCD, we first correct the two exposures of the step wedge by subtracting off the bias image. Thus

sc1 = s1.data - bias.data;
sc2 = s2.data - bias.data;

Next, form two new images:(a) the average of the two bias-subtracted images `sc1` and `sc2`, and (b) the square of the difference between the two uncorrected images — this is the variance. You should in principle use the raw images `s1` and `s2` (not `sc1` and `sc2`) to form the difference, since using the corrected images could introduce extra noise. (Though if the *same* bias frame is used to correct both, it will cancel exactly.) **See the theory write-up for the derivation of the equations you will use.**

You will need to define two separate boxes on **each step** of the step wedge, with each box containing several hundred pixels, and located well away from the edges of the steps. Also try to avoid obvious blemishes, etc. (In principle, one box per step is enough, but the second allows an important check on the results.)

The easiest way to get the boxes is with the **box_cursor** procedure. It is called as follows:

[x0,y0,nx0,ny0]=box_cursor;

When you enter this command, click and drag on the image to define a box. Now, the variables `x0` and `y0` contain the coordinates of the lower left corner of the box, and `nx0` and `ny0` are the width and height of the box. That's all you need! For example, if the entire image were `sc1`, then the sub-image of pixels within the box is just given by

sc1_box0 = sc1(y0:(y0+ny0),x0:(x0+nx0))

Furthermore, the **exact same box** on another image, e.g. the variance **var**, will be

var_box0 = var(y0:(y0+ny0),x0:(x0+nx0))

Note that the reason why x and y are reversed from which you would think is their natural order is because in the manner the data are displayed: x corresponds to the columns of the matrix, and y to the rows. The notation expected by MATLAB is that row is the first index, column the second, as in var(row, column).

You can then take the mean of this new sub-image **var_box0**, etc. If you want to define all the boxes at one go, you just call **box_cursor** repeatedly, with different names each time for the coordinate variables:

[x1,y1,nx1,ny1]=box_cursor;
[x2,y2,nx2,ny2]=box_cursor;, etc.

You should determine the mean and standard deviation for each box, both in the average image and in the variance image.

There is a simple procedure called **box_vals** which will automate the above steps somewhat – see the Appendix. It is in the same special directory as **rfits**, and should be callable as a standard function.

3.4 Analyzing the results

From these data, you are to determine the gain, G [ADU/electron], the reciprocal gain, $K = 1/G$ [electrons/ADU], the readout noise, σ_B [ADU], and the readout noise in equivalent photoelectrons, $K\sigma_B$. The appropriate equations were derived in class by assuming that the noise in the signal itself is due entirely to counting statistics and that the noise in the readout is a fixed value, whether expressed in digital units [ADU] or in equivalent electrons.

Use the MATLAB plot command (**plot(xvec,yvec)**) to graph the variance (y-axis) vs. the mean intensity (x-axis) for your measured (box) data. Fit these data to a straight line. Since a straight line is a polynomial of degree one, you can use the MATLAB **polyfit** procedure

coef=polyfit(xvec,yvec,1)

where **xvec** is a *vector* of x-values and **yvec** the corresponding y-values. The result, `coef`, is the vector of coefficients of the polynomial fit – in this case, just the equation $y(x) = m * x + b$, where slope m is **coef(1)** and intercept b is **coef(2)**.

To get an idea of how reliable your value of G may be, you should fit 3 least squares lines: one from one box from each step for the first linear fit, and the second from the other box from each step. The difference between these two fits is a crude measure of the accuracy of G. Discuss this in your report. Then, the third line will be your best value for G, a least squares fit to all the data taken together.

There are two ways to determine the read-out noise, σ_B. One is from the intercept of the least-square fit. The other is directly from bias image. Compute the mean and the standard deviation of the whole bias image. Now try cutting out a few boxes from the bias image, and compute the mean and σ_B in those boxes. Is there a large difference? *It is the σ from the boxes that gives you a true measure of the read-out noise.* Explain why the value from the boxes is better than the value from the whole image. How does this result compare to the intercept of the least-square fit? Which result do you think is the most reliable? Why?

We want you to write MATLAB scripts to process these data. For example, after testing a data analysis path interactively, create a MATLAB script (an m-file) that reproduces it and attach it to the documentation for your lab. We want to be able to reproduce what you did precisely, running these scripts. We do not want to see a hardcopy of your MATLAB session in your report: just a copy of the scripts you wrote and used. Write the MATLAB scripts using the MATLAB editor (the **edit** command), or your favorite external editor as discussed in the introduction to MATLAB document.

4 Report

Your write-up should have the following sections:

- Abstract: A short (a few sentences) summary of your report. This should state the goal of the lab and your final results.
- Introduction: A few paragraphs discussing why you did the lab. What are read noise and gain of a CCD, and why is important to determine them?
- Observations: Review the steps you followed to acquire your data and note any problems that arose.
- Results: This section should include a clear and concise record of the values measured during the analysis, written as a nice, well documented table with headings, etc. Examples are the means and standard deviations for all the boxes you measured, and the coefficients for the fits you found. Include plots of variance vs. mean intensity, for each of the 3 linear fits you performed, including both the data points and the line from the least squares fit. *Never turn in tables or plots where you have not labeled the quantities tabulated or plotted!* Explain how you analyzed your data to derive G and σ_B, including descriptions and discussions of the equations and methods you used.
- Discussion and conclusions: What do your results mean? How well do they agree with your expectations from theory? How do the values you obtain influence observations? Talk about the uncertainties in your results and answer any questions that are asked in the lab handout.
- Appendix: Append a copy of the MATLAB script or scripts you wrote and used to obtain your results. Just the scripts (`*.m` files), not a MATLAB session log!

Your raw images should be available also; please leave them in your directory along with your MATLAB script or scripts. The instructor may want to look at them.

Appendix

The following MATLAB procedure should be in your directory (otherwise, cut and paste the text into a file named box_vals.m). Read the commands to see how it works. Suppose you have obtained the the average of the bias corrected images and have called called it "sav". Suppose that the variance image is called "sdif2". First display "sav" and then invoke the procedure with "box_vals, sav, sdif2". Select the region of the image in which to carry out the computations. The means and standard deviations of the box pixels of both "sav" and "sdif2" will be printed in the screen. Feel free to modify this to return them into a variable!

```
function box_vals(av,difs)

[x0,y0,nx,ny]=box_cursor;
avbox=av(y0:(y0+ny),x0:(x0+nx));    % Note x corresponds to columns
difbox=difs(y0:(y0+ny),x0:(x0+nx)); % and y corresponds to rows
avm=mean(avbox(:));
dfm=mean(difbox(:));
sigav=std(avbox(:));
sigdif=std(difbox(:));
fprintf(1,'avm=%0.3f, sigav=%0.3f\n',avm,sigav);
fprintf(1,'dfm=%0.3f, sigdif=%0.3f\n',dfm,sigdif);
fprintf(1,'box=%d,%d,%d,%d\n',x0,y0,nx,ny);
```