Clio Linearity Calibration: Draft

Chris Bohlman

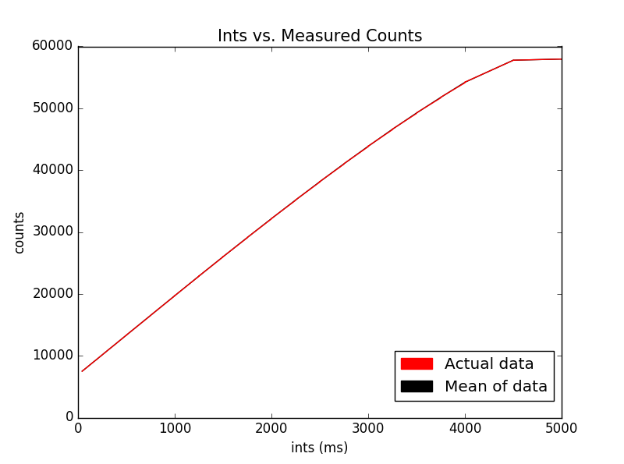
1/12/2017

Context:

The CLIO infrared camera is an instrument used in the Magellan Adaptive Optics system, located at the Las Campanas Observatory located in the Atacama Desert in Chile. Now, infrared light is light that has a wavelength of 800 nm to 1 mm. Since the Earth’s atmosphere is mostly opaque to infrared light, infrared astronomy should be conducted at high elevations, past as much of the atmosphere as possible. Also, adaptive optics are used to gather data and correct for the opaqueness of the atmosphere at those long wavelengths.

The context to this report is to analyze a set of images to calibrate the infrared camera CLIO. To do this, a program must be written to judge how exactly the images should be corrected for linearity. Now, this data has previously been corrected for linearity by Katie Morzinski. However, I chose to also attempt to correct this data, as I want to use what I have done for this specific set of data as a template for correcting data that hasn’t been corrected yet.

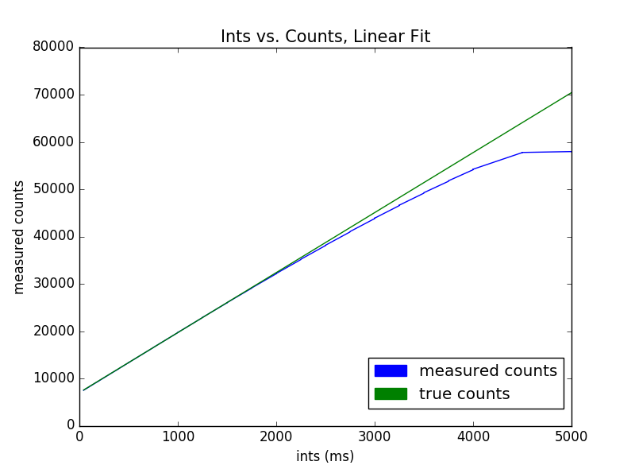
Ints vs. Counts:



To determine the integration time vs the counts of each picture, I read files in through the FITS package from astropy in Python, and adjusted the parameters of the read in’s area equal to original analysis’ parameters (200 to 350 in the x direction and 0 to 200 on the y axis) These parameters are due to the bright side of the images always being on the right half, so we wanted to really capture that part of the image. The program I wrote analyzed counts in that section of the image for each picture, and got the integration time’s measurement from the specific header for each image. In the context of this project, the integration time is best thought of the exposure time for the camera. Therefore, an increasing amount of ints means that the camera was open longer.

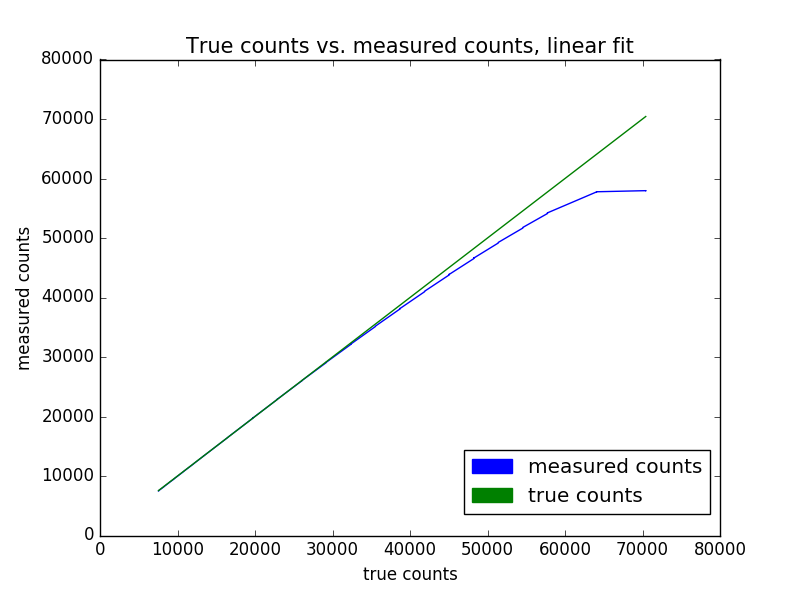
The data was stored in ‘int’ and ‘counts’ arrays, respectively. I graphed the integration time, which were in milliseconds, on the x axis and counts on the y axis.

Ints vs. Counts with linear relationship:



I needed to find a linear relationship with the data so I could find a method of determining the true counts, assuming that for true counts, I wanted it to be synonymous with integration time. I decided that the most linear part of the graph was between 500 ints and 1500 ints. Since there were 5 samples at each integration time, along with slightly different counts for each time in the group, I used polyfit from numpy to determine the coefficients of a line analogous with the area that I had picked out. I then plotted said line next to the data, showing where the data is linear and where it diverges from my calculated line.

True counts vs. Measured Counts with linear:

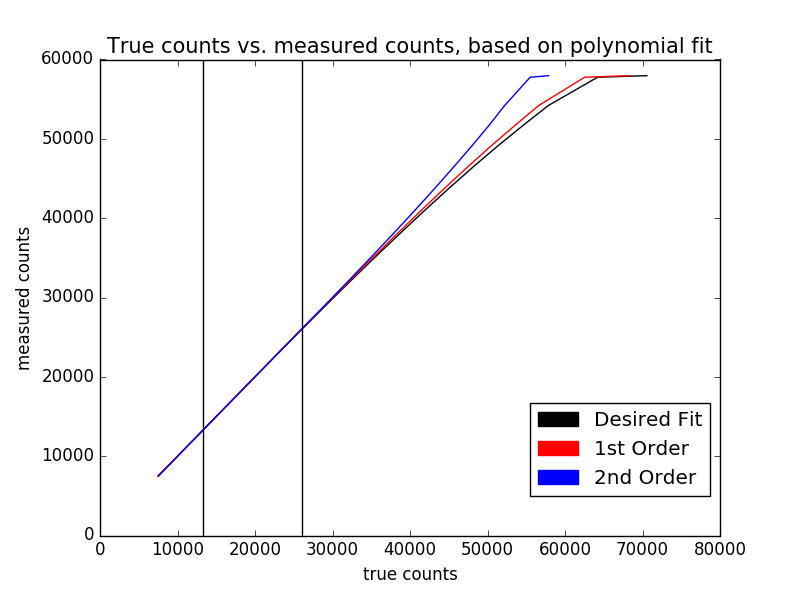


The first step in correcting for linearity was trying to find the straightest part of the data to add a linear fit to demonstrate the supposed linearity we were trying to achieve. I determined that the straightest part was between 300 ints and 750 ints, right at the beginning of the data set. The region of choice is in between the bars above. I chose this region among others because after graphing the linear fit in relation to other regions in the straight-line section of this graph, I zoomed into each of the fits, and found that this fit’s separation was minimal enough to provide a good fit for the rest and was a stellar fit.

The coefficients of the line were m = 14.12325 and b= 4802.21666667, if y = mx + b.

This line shows that the data is only strictly linear up to about 25,000 counts, so the linearity correction must be applied to any values above this.

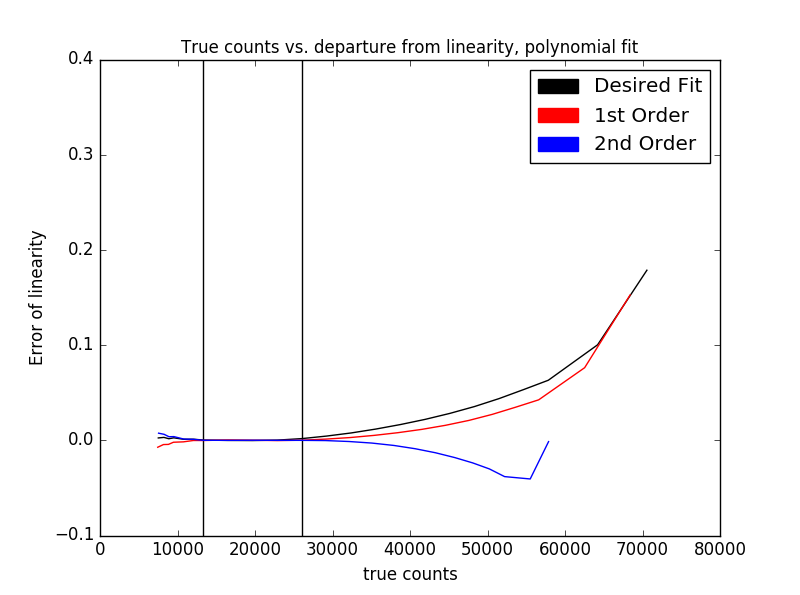
True counts vs. Measured Counts with all:



The next step was to calculate the true counts. The true counts represented what the data would look like if it was perfectly linear, and there was no calibration needed. On the above graph, this is the linear fit represents. However, we also wanted to fix the data for second, third, and fourth order polynomials, so we took the set of data up to about 40,000 counts, and created a function in each of those orders that would convert counts to true counts. By applying these function coefficients that were generated to counts for each order, 3 more true counts graphs were generated. By plotting the original data on the same graph, we could see the third order fit was the most accurate in this case.

The bars in the graph also represent where the linear fit was chosen from for all orders of the fits.

Error of Linearity vs. True Counts



To confirm the results from the previous page that the third order fit was superior, we decided to calculate the error of linearity for each of the different polynomial fits. The error of linearity was calculated by:

Error of linearity =

With true counts referring to the specific true counts of each fit, calculated earlier.

As seen from the picture, 3rd and 4th order come very close to the fit, and diverge at about the same point from the original data. However, only the 3rd order fit still behaves in the same way as the original data. Therefore, from this graph, we can adequately support the claim that either the 3rd order or the 4th order fit is the best fit for this data. Since both were fairly close, I applied both to the raw data separately in order to demonstrate which one corrected the data the best.