Asteroid Light Curves: 354 Eleonora

2205604

April 2019

Abstract

Asteroids are small but important bodies in our Solar System. They are of great scientific interest as the oldest contain material that has not been exposed to the harshness of space for millions of years. Making light curves of these asteroids can give insights into their shape and rotation period, as well as possible correlations between these parameters and their size, position in the Solar System, etc. This report discusses a project that involved making a light curve of an asteroid, 354 Eleonora and writing a Python code that calculated the rotation period of the asteroid, as well as its apparent magnitude. A partial light curve was obtained, meaning a rotation period could not be determined, but 354 Eleonora's apparent magnitude during observations was found to be $m = 10.0945 \pm 1.2819 \times 10^{-9}$. This value is somewhat off the value given by the *Cartes du Ciel* software of $m = 11.0$. Simulated data was also created, and this showed that not all objects were being tracked correctly.

1 Introduction

Asteroids are small rocky objects that orbit our Sun just as the Earth does. They are not planets however, as they do not have a round or nearly round shape and they have not cleared their neighbourhood. The majority of asteroids are located in an area between the orbits of Mars and Jupiter called the Main Belt. The largest asteroid is Ceres, which has a diameter of just under 1000km and a mass of 1.3% of the Moon [1]. Ceres is unusual for an asteroid however, as it is the only one to be classed as a dwarf planet. Asteroids are some of the oldest objects in the Solar System and are classified by a letter that indicates their composition. Most of the asteroids are class C or S [2], indicating that they are of Carbonaceous or Siliceous composition. The oldest asteroids are of great scientific interest, as they contain material that has not been exposed to anything since the dawn of our Solar System. The Japanese Space Agency (JAXA) currently has a spacecraft, *Hayabusa2*, in orbit around 162173 Ryugu, a type C/Cg asteroid, and one of the main objectives of the mission is to collect samples of rock from inside the asteroid. This is to be achieved by detonating explosives to create a crater, exposing material to space for the first time in millions of years that can then be collected by the craft and brought back to Earth. This mission is ongoing at the time of writing, though the explosives detonation had taken place [3].

A light curve is a plot of brightness, photon count or magnitude versus time. Light curves are also often plotted against phase. They show the evolution of the flux received from an astronomical object over time. In the case of an asteroid, its irregular shape means that the light it reflects varies as it rotates on itself. This leads to periodic dips in its measured light curve, if these dips are pronounced enough. An analysis of this light curve can give a value of its rotation period, ie: how fast it spins on itself and an indication of the shape of the asteroid. Finding such light curves for a large number of asteroids can also lead to the

discovery, or not, of any correlation between asteroid size, rotation period, composition and position in the Solar System, which could be an important step in understanding more of our local environment [4] [5]. Other reasons for wanting to obtain these light curves include gathering information for the planning of missions to these asteroids, and the risk assessment these asteroids could pose in a close encounter with the Earth.

In this project, the group I was part of picked an asteroid to observe from the University of Glasgow Observatory, so we could obtain its light curve and determine its rotation period. An analysis code written in Python was then created, to input the collected frames and return a light curve, with the rotation period.

In this report, a brief overview of the apparatus used will be given, as well as a list of equations used for the calculations made during this project. Next, an explanation of the process decided by the group will be provided in detail, followed by the results obtained and a discussion of these. The final conclusions from this project will bookend this report.

2 Apparatus

For this project, the JPL Small Body Database [6] was used to help make a list of potentially observable asteroids. The *Cartes du Ciel* software was also used to determine whether asteroids would be in our night sky at the time of year, and helped pick a suitable background star.

A CCD camera, the Atik 314L+, was connected to the Meade 16″ telescope located in the observatory's main dome to take the frames we wanted to collect. These frames were then downloaded onto a laptop, which could display what the CCD camera was detecting. The *SkyX* software, which was running on a different PC, pointed the telescope and centred it on whatever astronomical object was selected. It would also track the object along its flight path, to keep the telescope centred on it.

The Jupyter Hub was the software used to write to code used to analyse the collected frames. The code was written in Python.

Figure 1: The Atik 314L+ CCD camera connected to the Meade 16″ telescope

3 Equations

Equations 1 and 2 are the equations used to calculate the field of view for each CCD camera [7]. The values of pixel size, number of pixels and focal length were given values and did not need to be calculated.

$$\text{Field of view per pixel} = \frac{\text{pixel size } (\mu\text{m})}{\text{Focal length (mm)}} \times 206.3 \tag{1}$$

$$\text{Field of view for CCD} = \text{Field of view per pixel} \times \text{number of pixels} \tag{2}$$

The following five equations are needed to calculate the exposure time required to observe the chosen asteroid.

$$N = \frac{\eta S_\nu A \Delta\nu \tau}{h\nu_0} \tag{3}$$

$$F = S_\nu \Delta\nu \tag{4}$$

N is the number of incoming photons, ν is the quantum efficiency of the CCD, S_ν is the flux density (in units of Jansky, Jy), A is the collecting area of the telescope (in metres), $\Delta\nu$ is the bandwidth, τ is the exposure time, h is the Planck constant and ν_0 is the central frequency. F is the incoming flux.

Rearranging equation 3 and substituting in equation 4, the formula to calculate the exposure time for our asteroid can be found:

$$\tau = \frac{Nh\nu_0}{\eta F A} \tag{5}$$

The values of F and A first need to be calculated however, and that is achievable through the following equations:

$$A = \pi r^2 \tag{6}$$

$$F_{ast} = F_{Vega} \times 10^{-0.4(m_{ast} - m_{Vega})} \tag{7}$$

$$F_{Vega} = \frac{L_{Vega}}{4\pi d^2} \tag{8}$$

Equation 7 is Pogson's equation, where the m values represent apparent magnitudes. In equation 8, L_{Vega} is Vega's bolometric luminosity and d is the distance to Vega.

The last three equations are used for error analysis. Poisson statistics were applied, as it was assumed incoming photons on the CCD arrived independently in time at a constant rate. These assumptions lead to equation 9 being the definition of the errors on each pixel. These errors could then be propagated using equations 10 and 11.

$$\Delta\text{pixel count} = \sqrt{\text{pixel count}} \tag{9}$$

If $z = x \pm y$:

$$\Delta z = \sqrt{\Delta x^2 + \Delta y^2} \tag{10}$$

If $z = x \times y$ or $z = \frac{x}{y}$:

$$\Delta z = z \times \sqrt{\left(\frac{\Delta x}{x}\right)^2 + \left(\frac{\Delta y}{y}\right)^2} \tag{11}$$

4 Method

We first decided how to proceed with the project. Our initial main goals were to choose the best asteroid and the best CCD camera for observations. We then figured out how to proceed with observations: which star to use as a reference, calculating exposure times, in what order to take frames, naming conventions, etc. We could then build the code around these decisions.

4.1 Choosing a CCD camera and an asteroid

First of all, the equipment needed to be picked. We had a choice of two CCD cameras to use: the Starlight Xpress SXV-H16 or the Atik 314L+. Field of view calculations could be made using equations (1 and 2) to help choose between the two. The necessary data values were found in the CCD cameras' respective manuals [8] [9].

Next, we needed to pick a suitable asteroid. We defined a 'suitable' asteroid as one that has a rotation period of less than 6.5hours with a magnitude change that we could detect and that we can find in the *Cartes du Ciel* software. We wanted to pick a short period asteroid, as that would enable us, in ideal conditions, to take data for at least one full rotation period in one observing night. *Cartes du Ciel* is a software that shows us what the sky looks at at our coordinates and altitude ($55^\circ\ 51'\ 26''$, altitude $\simeq$ 50m [10]) and displays whatever bodies you want to see at whatever time. It is an important tool, as it can show us when an asteroid is above the horizon, and so can help us remove asteroids from our consideration if they are not going to be observable at this time of year. The JPL Small Body Search Engine was used to make a shortlist of asteroids to choose from. This database contains a wealth of data (such as orbit eccentricity, orbital inclination, geometric albedo, rotation period, etc) about over half a million numbered asteroids. The *Cartes du Ciel* software contains a database of the first 100 000 numbered asteroids, so a filter is used in the JPL search engine to only consider the first 100 000 numbered asteroids. Other filters were applied to narrow down the possibilities: the rotation period must be less than 6.5 hours and a magnitude change must be known. We also applied other filters to display certain data: the name of the asteroid, and its apparent magnitude. This should give a relatively short list of asteroids that fit these criteria, that can be narrowed again. The list was copied into an Excel file where we selected the asteroids that had a low apparent magnitude and a high dip in brightness. The asteroids in this shortlist were then searched in *Cartes du Ciel* to see if they could be observed at night at this time of year, and if so, for how long. Combining this with the asteroids' apparent magnitudes, a ranking could be made, with the asteroid placed in first being the one we would focus our efforts on.

4.2 Observations

Our next step was to decide how to observe. This consisted of calculating, first of all, how long our exposures should be. This required using equations 6, 7 and 8 to calculate the incoming flux from our asteroid. The remaining values to use in equation 5 could then be found and the exposure time determined.

We next needed to pick a a star to use as a reference; this star needed to be of a similar type to the Sun, ie: a G type star, to be of roughly the same apparent magnitude as the chosen asteroid and to not be a variable star. If we were to plot the light curve of a non-variable star (such as the Sun), we should obtain a straight line. However, as the star drops in our field of view, we need to look through more atmosphere, so more absorption occurs, hence its light curve would drop with time. The incoming flux is also affected by seeing and by upper cloud cover. Taking frames of a background star and analysing them can thus help us correct the asteroid's flux. We assume that this is the only process affecting the incoming flux from the star and that it affects the flux from the asteroid in the same way. The reason why we want

the star's apparent magnitude to be similar to the asteroid's is so the exposure time need not be altered, particularly if the star and the asteroid are close enough in our field of view to be seen in the same frame.

The last details to decide before writing the analysis code were to pick the naming conventions for our frames, determining in which order to take frames, ie: when to take asteroid/star frames and dark frames, and how the dark frames should be processed: whether they should be combined into an average dark frame, or whether certain dark frames would only be used for certain asteroid/star frames.

We needed data to put into our analysis code, and this needed to be taken on a clear night, so our results could be as accurate as possible. The data was taken by using the Meade 16" telescope in the main dome of the University of Glasgow Observatory. The telescope had our chosen CCD camera attached to it at the focus and it was connected to a laptop and to a PC. The laptop controlled the data collection from the CCD camera: the number of exposures in a loop and the length of these exposures, as well as the naming of the files resulting from the collection. On the laptop screen, we could also see a preview of a frame just after it had been taken. This is also where the data was saved, before it could be copied and shared on the Cassegrain server. The PC was running a software called *SkyX*, which was similar to *Cartes du Ciel* in the fact it showed us the night sky at our location and contained data regarding a vast amount of astronomical bodies. The *SkyX* software however, when linked to a telescope, could point it and centre it on an object. It would also track the object, keeping the telescope centred on it at all times, following its path along our night sky.

Before we could begin taking data, the setup first needed to be calibrated. With the CCD correctly placed at the focus and the rest of the setup ready, we were first to point the telescope at a large, bright source, such as a bright star. We could then focus the setup and check if the source was centred in our field of view, either on the laptop or by using an eyepiece. If the source was not centred, we were to put the source in the centre by manually adjusting the telescope, and adding the correction to the *SkyX* software on the PC. Once done, we were to slew to another slightly dimmer source, and repeat the previous process until the the light source would be centred immediately after slewing to it.

4.3 Analysis

The most time-intensive part of this project was the writing of an analysis code, which would take the frames collected during observations as inputs, and output the pixel counts for the asteroid and the star, asteroid pixel counts corrected by the star's evolution and time values for the asteroid and star frames, all with calculated errors. Four plots were also outputted, a light curve for the asteroid, another for the star, a corrected light curve for the asteroid and another corrected light curve for the asteroid in terms of apparent magnitude, not pixel counts. Finally, a value for the apparent asteroid magnitude was calculated and outputted. The analysis was split into seven sections: flat frame analysis, dark frame analysis, analysis of the asteroid data, then of the star data, correcting the asteroid data, finally plotting the outputted data and displaying the apparent magnitude value.

For the flat frame analysis, the flat frames are read in one by one, added together and averaged to give an average flat frame.

We came up with two options for the dark frame analysis, we called them 'full average' and 'selective average'. The full average analysis works in the exact same way as the flat frame analysis: the dark frames are added one by one and averaged together. The selected average averages groups of ten dark frames together to create a series of dark images. The asteroid/star frames collected in a sequence are then corrected by the averaged dark frames taken before the sequence and by the averaged dark frames taken afterwards. The reason for this is due to a

hypothetical evolution of the thermal noise present in dark frames due to the drop in ambient temperature during observations.

Steps 3 and 4 both call the same function, called *analysis*, but use different inputs: step 3 uses appropriate inputs for the asteroid, and step 4 for the star. The *analysis* function first undertakes a stage called frame cleaning, where an asteroid/star frame is called, the appropriate dark frame removed and the result divided by the average flat frame. This process is looped through every asteroid/star frame. Then, the function runs through each frame and picks out all the coordinates of all the light sources, and it calculates the sum of the pixel counts for the light sources. It then inputs these results into another function which picks the asteroid or the star (whichever is desired) from the list of centroids (light sources) and adds its pixel count to a list. Groups of ten asteroid counts are averaged together, to form a list of average counts. The last step of the *analysis* function is to obtain the times at which observations were taken. In this process, the timestamps from the files are extracted and converted to seconds, then an average time for each average image is calculated. The starting time is removed from every time value to give a list of times that starts at 0. The *analysis* function's outputs are the average counts for the asteroid/star and the observation times.

The next step is the correction of the asteroid's counts from the analysis of the star's data. Here, the first star count value is used as a reference. The reference value then goes through the list of star counts, where it is divided by each element. This gives a list of ratios, called multipliers, which represent how much brighter or dimmer a star image is in comparison to the first star image. Each asteroid count element is then multiplied by the corresponding element in the multipliers list. This gives an array of corrected asteroid counts.

The sixth step in the code is the plotting of the data. Three light curves are plotted: the first is a light curve of the chosen asteroid, the second is a light curve of the star and the third is the light curve of the asteroid corrected by the star's evolution. Colours and line styles have also been assigned for clarity.

The last step of the code consists of converting the pixel counts for the corrected asteroid into apparent magnitude values. These values are then plotted against the same time values as above. Next, an average value for the apparent magnitude is calculated and displayed.

Error analysis was also added to the code and propagated through every function so it could be displayed with the final results, including on all four graphs. We assumed that Poisson statistics were applicable in our situation, meaning equation (9) was valid. The error propagation followed the simple rules of equations (10, 11). These equations were added to the functions that dealt with pixel counts where appropriate, and the corresponding modifications to inputs and outputs made. However, in the *centroids* function, the error on the sum of the counts from the light sources were calculated by the *aperture_photometry* function, which is a function that is imported. This function calculates errors on the sum of the counts for each light source it detects, as long as the errors for each pixel is inputted. The errors for each pixel have been calculated in the previous functions. The error on the timestamps incorporated into each file is assumed to be $\pm 1s$. The same error equations as earlier were applied to the *get_time* function, which allowed errors on each time value to be calculated, and later to the rotation period.

The full code has been added to this report in Appendices A and B.

5 Results and Analysis

5.1 CCD Choice

The field of view was calculated for each CCD camera using equations (1 and 2). This could be done by finding the number of pixels and the pixel size in the CCD cameras' manuals.

	Atik 341L+	Starlight Xpress SXV-H16
Number of pixels	1391×1039	2048×2048
Pixel size	$6.45\mu m$	$7.4\mu m$
Field of view per pixel	$0.489''$	$0.561''$
Field of view for CCD	$680.20'' \times 508.07''$	$1148.93'' \times 1148.93''$

Table 1: Field of view values for our two CCD cameras

The Starlight Xpress has a wider field of view, so this was the CCD camera we picked.

However, the drivers for the CCD camera would not install on our laptops so we decided to switch to the Atik camera. Also, one of the other groups tested the Starlight Xpress camera and found the data would save in an unusable format.

5.2 Finding the best asteroid

The filters described in the above section were added in the JPL Small Body Search Engine, and gave a long list of 32 numbered asteroids. From these 32, nine were shortlisted, by selecting the asteroids which had an apparent magnitude of less than 11.5 and a 'magnitude slope', the drop in brightness, of more than 0.2. The shortlist is shown in the table below.

Full Name	Rotation Period (hours)	Change in magnitude	Apparent Magnitude
4 Vesta	5.34212766	0.32	8.0
433 Eros	5.27	0.46	9.3
354 Eleonora	4.277	0.37	10.6
349 Dembowsa	4.701	0.37	10.7
15 Eunomia	6.083	0.23	10.8
29 Amphitrite	5.3921	0.2	10.9
216 Kleopatra	5.385	0.29	10.9
44 Nysa	6.422	0.46	11.2
129 Antigone	4.9572	0.33	11.3

Table 2: Shortlist of asteroid candidates, in order of increasing apparent magnitude

From these nine asteroids, we found using the *Cartes du Ciel* software, that 4 Vesta, 15 Eunomia and 29 Amphitrite would not be observable at the right time, and that 44 Nysa would not be observable for very long, so obtaining a full light curve would require several observing nights. These five asteroids were removed from our consideration. The remaining five asteroids were then ranked through a combination of absolute magnitude and observability. The ranking is shown in the table below.

Full Name	Rotation Period (hours)	Apparent Magnitude
433 Eros	5.27	9.3
349 Dembowska	4.701	10.7
216 Kleopatra	5.385	10.9
354 Eleonora	4.277	10.6
129 Antigone	4.9572	11.3

Table 3: The best 5 asteroids to observe in ranking order

Therefore, 433 Eros was the asteroid we decided to focus on and observe.

5.3 Setting up observations and Change of focus

With the asteroid chosen, the next step was to calculate the exposure time needed to take a frame of 433 Eros. At the time these calculations were made, the Starlight Xpress SXV-H16 was the CCD camera we were going to use.

In the CCD camera's manual (*REFERENCE*), it was recommended not to try and collect more than 20 000 photons in an exposure, so we decided to set out N value to be half that, ie: $N = 10000$. A quantum efficiency graph was also present at the end of the manual, this gave us a value for η and for the central frequency. The collecting area of the telescope could then be found by setting $8''$ as the radius and converting it into metres. The luminosity, apparent magnitude and distance values for Vega can be researched and used in equations 7 and 8 to give the asteroid flux. The exposure time could then be calculated using equation 5. Calculating and finding the necessary values, we obtained an exposure time of 30s for 433 Eros.

At this point in the project, our plans for the project were forced to change. There had been very few clear observing nights, so all the groups working on this project collectively decided to focus on 354 Eleonora, as one of the other groups had already collected data on this asteroid. This decision did not change much for our group, all that needed to change was the exposure time and the background star used for the correction stage. Note: 354 Eleonora was our fourth choice.

Unlike for 433 Eros, where we had calculated the required exposure time using equations 5 to 8, for 354 Eleonora, we decided on the exposure time while calibrating the equipment for night observations (more details on the calibration later in this section). We were told that another group had observed 354 Eleonora with exposure times of 6s. We tested this before starting and found the resulting frames to be too noisy. We then tried again with an exposure time of 12s and found the results much more appealing.

The group that had already taken data had used a star named HD 287493 as their reference star. For consistency, we also used that star, which is a G0 type star and had an apparent magnitude of 9.53 on the observing night, but was not in the same field of view as the asteroid. This meant that when wanting to take data from the star, we would need to slew the telescope from the asteroid to the star.

We also decided on an observing pattern, ie: the order in which we would capture frames. We started from the basis that we needed to collect four types of data: asteroid frames, star frames, dark frames and flat frames. The decision was taken to take frames in loops of ten exposures, so that averages of these ten exposures could be made later on in the analysis. Flat frames were to be taken at the end of our observations, as the setup needed to be focused and we did not want to waste observing time at the beginning. We also assumed that the pixel counts from the star would not vary as much as the counts from the asteroid would, so we took half as many star frames as asteroid frames. As a result, every two sets of ten asteroid frames taken would be followed by the collection of one set of ten star frames. Taking into account the time taken to collect the data for each set and the time for the telescope to slew between the asteroid and the star and back again, we figured that in one half-hour of observations, it was possible to take six sets of asteroid frames and three sets of star frames, with a set of dark frames taken at the start of the half-hour and another set at the end. Graphically, this sequence looks as follows:

DF A A S A A S A A S DF

DF, A and S represent the collecting of 10 dark, asteroid and star frames each respectively.

This sequence was to be repeated as many times as necessary, until either data for a full rotation period was collected (4 hours 17 minutes for 354 Eleonora, see 5.2) or until it is no longer possible to observe in good enough conditions.

5.4 Observations and Results

It turned out that no data for any asteroid was present on the *SkyX* software, so 354 Eleonora's orbital flight path had to be manually added to the software's database before observing could begin.

The setup calibration took a long time, as it was difficult to keep the object in the centre of the field of view without having to manually adjust it. There were also three stars in the field of view, about which we tried to find information, in particular their spectral type. If one of them were to be a G type star, it would have been more convenient to use it for the correction stage, rather than have to slew numerous times to the star we had decided on previously. Unfortunately, we could not find any such information, so we kept HD 287493 as our correction star. However, the time it took to make this decision meant that we lost around an hour of observing time.

The observing sequence described in the subsection above was completed six times, giving us a total of 360 asteroid frames, 150 star frames and 70 dark frames. 40 flat frames were also taken. The asteroid frames were named 'object_', the star frames 'star_', the dark frames 'dark_' and the flat frames 'flat_'. Each file name was followed by a number, with this number representing the order in which it was taken. It was also made sure that there were no gaps in the numbers, as the code used to analyse this data would run through each file in number order, with no breaks encoded.

We stopped early as the star had dropped too far down in our field of view for us to observe. Light pollution became so great that we were not able to resolve the star at all, even if we had continued. This is why there are 30 fewer star frames than expected. Each observing sequence took slightly longer to complete than originally planned, due to having to manually adjust the telescope's orientation to try and keep objects centred (more on this shortly). This meant we had acquired data over a period of just under four hours, so the majority of a light curve should have been obtained.

We started to find issues with our data when first analysing it. It turned out that the object in our field of view that we thought was the asteroid was in fact one of the background stars, and that the asteroid had drifted out of our frames by around object 160. The rest of the frames are therefore just background.

This happened for two reasons: the first was mistakenly identifying the asteroid in our field of view, and the second was that the *SkyX* software was not tracking the objects accurately enough. We wanted the object we were observing (the asteroid or the star) to be in the middle of our field of view, but it was also possible to manually adjust the telescope's orientation if the object was not in the middle, and add this correction to the software. As we slewed the telescope between the asteroid and the star, the tracking became more and more offset, displacing the objects in our field of view more and more, meaning we had to continuously adjust the telescope, despite adding the corrections to the software. This made keeping objects in the middle of the field of view very difficult and that resulted in some of the objects disappearing from our field of view altogether. Unfortunately, as we had mistaken one of the background stars as being an asteroid, the actual asteroid was one of the objects to disappear from our frames. An illustration of the asteroid leaving our field of view is shown in figures 2, 3 and 4.

The asteroid 354 Eleonora is the object in the top left part of the first frame. It is clear to see that it has disappeared from our field of view by frame 170. The brightest object in these frames was a background star we thought was 354 Eleonora

We took our 40 flat frames at the end of our observations, keeping the setup in the same focus. To take the flat frames, we covered the detector with a blanket that had a white side to it and shined a headlight on it. However, we could see the collected frames appear as a preview on a laptop we were using, and we could see that the flat frames did not look as they should. The flat frames could change drastically from one frame to the next, so we decided

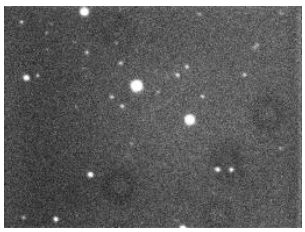


Figure 2: object_150

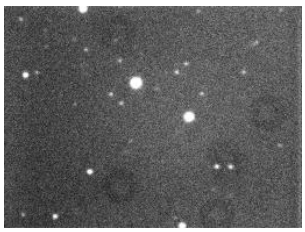


Figure 3: object_160

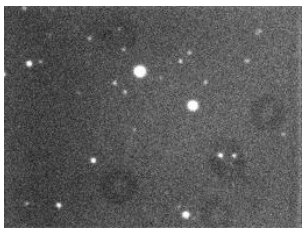


Figure 4: object_170

to discard them, with a view of potentially taking more at a later date, more rigorously. The reason for the odd-looking frames was likely due to the way we took our frames, if we'd pointed the telescope at a lit side of the inside of the dome, our results would likely have been better and usable.

Two of these flat frames are displayed in figures 5 and 6.

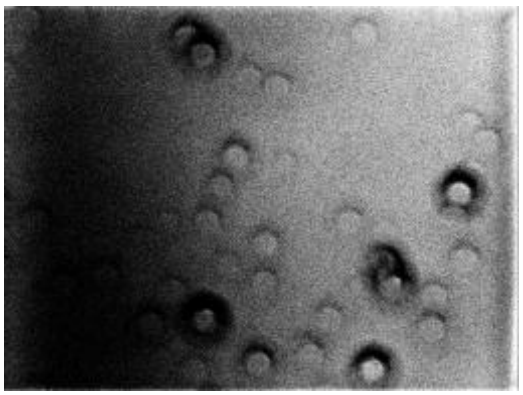


Figure 5: flat_5

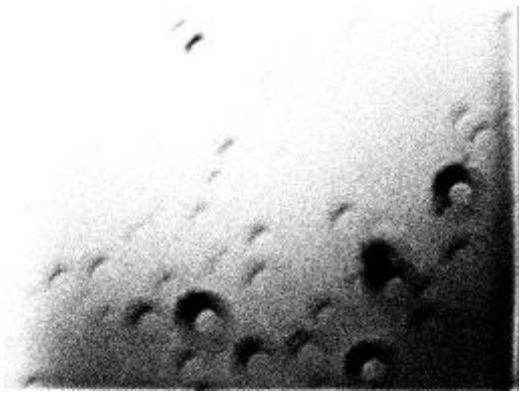


Figure 6: flat_17

Dark frames were taken by simply putting the cover for the detector on, and taking exposures of the same length as for the star and asteroid frames, ie: 12s.

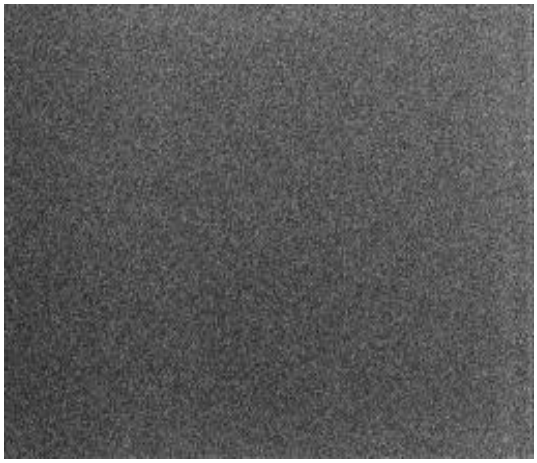


Figure 7: Example of a dark frame taken during observations

Not having the asteroid in the centre of the field of view had another consequence on our analysis, but this time a minor one. The function *new_coord*, which is a part of the *analysis* function, picks the asteroid or the star from all the light sources (centroids) in a frame, by picking the centroid that is closest to the asteroid coordinates from the previous frame. It also pinpoints the corresponding pixel count and pixel count error. However, this is not possible

for the first frame. If the asteroid or the star were in the middle and/or the brightest object in the field of view, the initial asteroid or star coordinates would have been the position of the closest centroid to the middle of the frame and/or the centroid with the highest pixel count. With the data at our disposal, this could not be guaranteed, so we have had to manually find which centroid corresponds to the asteroid or star in the first frame and add these findings as initial inputs to the *new_coord* function.

To conserve our observing sequence, we ran our code on the first 120 asteroid frames, the first 60 star frames and the first 30 dark frames. The selective dark frames were chosen as inputs for the *analysis* function. We still did the flat frame analysis with our 40 flat frames, but it was not taken into account in the frame cleaning. This would give us a section of the light curve, though two thirds of our data from the night had to be discarded. Below are the results obtained from running our code over these frames.

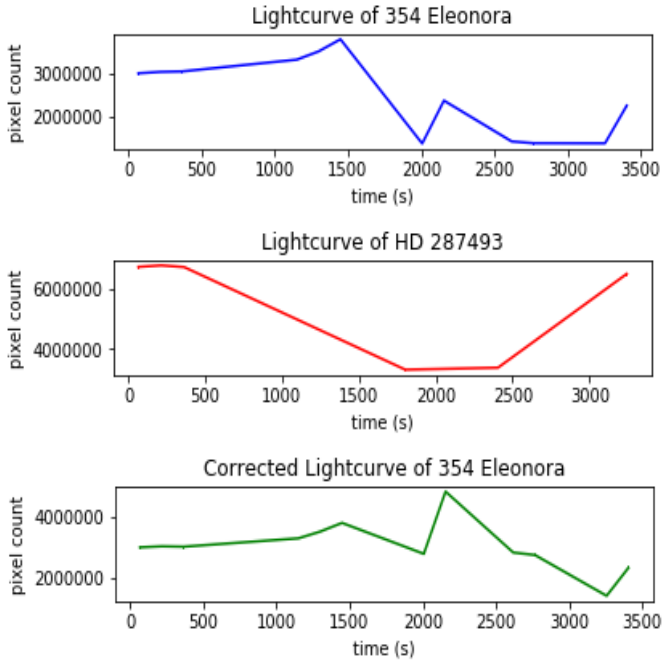
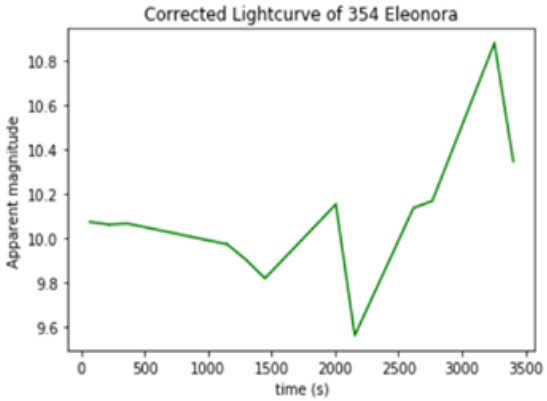


Figure 8: The three light curves obtained from observations of 354 Eleonora



Average Apparent Magnitude = 10.09459763880401+/-1.281943785873694e-09

Figure 9: Corrected light curve in terms of apparent magnitude and an average apparent magnitude

We decided to create some simulated data, which we could use to show the kind of results

we would expect if we had some good data of our own to use. It would also confirm that the code used to analyse inputted data works correctly and outputs the values and plots we want to obtain.

Among the simulated data were 400 frames, which each contained a simulated asteroid, a simulated star and other background stars. The frames all contained random noise that became more important in each frame. The asteroid would also travel and its brightness would vary to try and replicate what we would observe with a real asteroid; its 'rotation period' was set to be 1.5 hours. The star's brightness was also made to vary so we could test the correction step in the code. Dark frames (110 of them) were also simulated so they could be used in the frame cleaning step, however flat frames were not. Each frame also had a timestamp attached to it, that could be extracted in the exact same way as in real frames. As such, the simulated files could be processed by the same code in a very similar manner. Similar to the analysis of the real data, the selective dark frame analysis was used. The results we obtained from this data are shown below (figure 13), along with examples of the simulated frames (frames 10 to 12):

Figure 10: First simulated frame

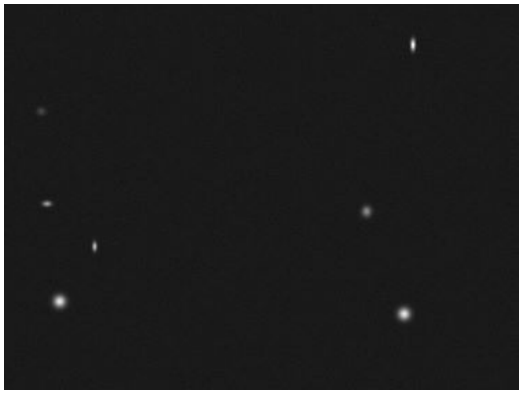


Figure 11: Last simulated frame

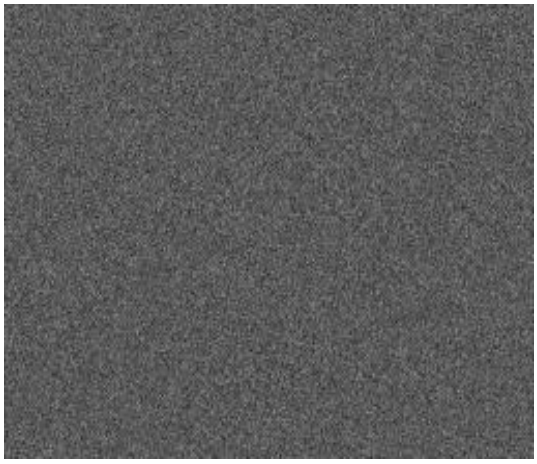


Figure 12: Example of a simulated dark frame

The code used to create these simulated frames is in Appendix C.

5.5 Discussion of Results

5.5.1 Results

We have very few results to show for all the effort put in by the group as a whole. Because of the mishaps that occurred during the observing night, two thirds of our data from that

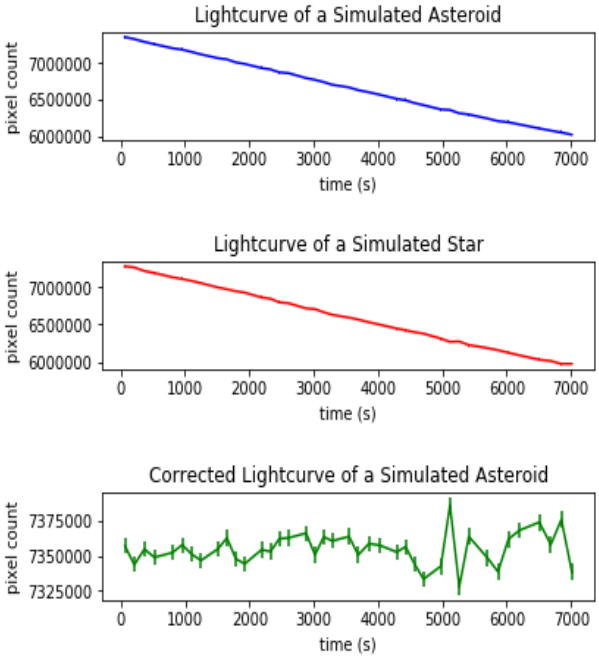


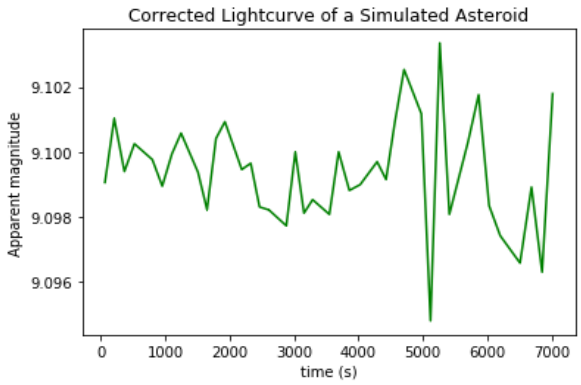
Figure 13: The three lightcurves obtained from the simulated data

night could not be used for our light curves. There were also no other clear nights to retake data after that one night, so there were no opportunities to obtain better, more usable data that could have produced some results. We were not able to determine 354 Eleonora's rotation period, but we were able to plot its light curve in pixel counts and in magnitude, as well as determine an average apparent magnitude during observations. This value was found to be $m = 10.0945 \pm 1.28 \times 10^{-9}$ and is not far off the value on the night of $m = 11.0$ given by *Cartes du Ciel*.

We were also able to obtain results for the simulated data: four light curves and an average apparent magnitude for the object that was tracked. (See figures 13 and 14)

Error bars are present on all the light curves, but are very small in value, so might not always be visible. Basic error analysis was performed, and efforts were made to make sure errors were correctly propagated throughout the code, nonetheless, the error values seem perhaps too small. The only values obtained with error were the average apparent magnitude for 354 Eleonora and for the simulated data (whichever object was tracked).

However, obtaining these light curves was not straightforward. Getting the *new_coord* function to track the correct object proved to be a lot more difficult than expected. The initial coordinates needed to be manually inputted, as the function requires the object's coordinates in the previous frame as inputs so it can pick out the correct object from the light sources in the current frame. To find these initial values, the *centroids* function was run on the first frame and that frame displayed alongside the results. By knowing where the object was in the frame, the coordinates of the corresponding light source could then be identified among the results and added to the code. However, the *centroids* function proved to be inconsistent, particularly the imported *CircularAperture* or *aperture_photometry* functions. Depending on the number of frames inputted into the function, the number of light sources detected would vary and their positions change. Also for real data, no light sources would be found in any frame if they had not had the dark frames removed first. For the simulated data, even when run over all cleaned frames and the correct values added to the code, the correct object would not be tracked. This is visible in figure 13, where the asteroid light curve should be a sinusoidal wave, but is instead a straight line. It seems one of the background stars was tracked, instead of the asteroid. We



Average Apparent Magnitude = 9.099372458668691 +/- 7.027927833977879e-10

Figure 14: Corrected lightcurve in terms of apparent magnitude and an average apparent magnitude value

could have manually added the object coordinates from each frame into an array, but that would have been too time consuming and inefficient as, for example, there are 400 simulated frames.

We had a will to make the code run as automatically as possible, with as few inputs as we could manage. This was not completely achieved, the code could have been made more automatic, but it would not have made much difference, either in terms of convenience or in terms of run speed. One major modification could have been made: in cases where the asteroid and the star were in the same frame (such as in the simulated data), the frame cleaning process would have occurred only once, rather than be completed twice. We observed the frame cleaning process to be the most time-consuming stage when running the code, so only calling it once would have saved a considerable amount of run time. Had we not had as many issues in trying to get the code to work, making this modification would have been achievable.

5.5.2 Possible Improvements

Certain adjustments would have made this project better and more efficient. On the observational side, taking better flat frames would have made our analysis more complete, and this would have happened if we had been able to take more data. Nonetheless, if we had correctly identified the asteroid from the objects in our field of view, we would have had much more data to analyse, regardless of the accuracy, or not, of the tracking. Had we also been able to start observing earlier in the night, we could have collected enough data to cover an entire rotation period.

On the computing side, a lot of time was spent waiting for results, as the entire code would often take hours to run, often leading to errors. Running the code in one go was a decision taken to make the process as automatic as possible, but we possibly grouped all the steps too early. Spending more time testing all the functions individually might have helped speed up the completion of the code, which could then be rendered automatic.

Another possibility for improvement was to write our own *CircularAperture* and *aperture_photometry* functions, though this would have taken a long time to achieve. However, if successful, these new functions would have been considerably more reliable, and would have helped produce the results we were looking for.

This would have been less of an issue if we would have been able to run the code on hardware with more computing power, particularly when running large amounts of data. For example, running the simulated data would take over five hours to complete.

6 Conclusions

To conclude, we picked an asteroid and a CCD camera, 433 Eros and the Starlight Xpress SXV-H16 respectively, but we had to change both our choices. The asteroid had been chosen through a careful selection of filters, depending on the parameters we required, and the CCD camera was selected by calculating the field of view for our two choices of camera and choosing the one with the largest value. The drivers for the CCD camera would not install correctly and, according to another group, it would save the files in an unusable format anyway. This meant we used the Atik 314L+ camera. A lack of clear observing nights meant all the groups working on this project had to pool their resources together and focus on one common asteroid. As one of the groups had already collected data, we all switched to their asteroid, 354 Eleonora, which was our fourth choice. Eleonora is an S type, Main Belt Asteroid, with a rotation period of 4 hours 17 minutes and an apparent magnitude on the observing night of 11.0, according to the *Cartes du Ciel* software.

We decided on an observing sequence, once the exposure time required for our asteroid was calculated. To keep consistency, we used the same background star as the other group, HD 287493, as the star to use to correct the asteroid's light curve for any atmospheric effects (absorption, seeing, upper cloud cover). This star is a G0 type star, with an apparent magnitude of 9.53, again according to the *Cartes du Ciel* software.

The calibration of the setup delayed the start of our observations, due to the *SkyX* software, which centred the telescope on a selected object, not centring correctly, meaning manual adjustments had to be made several times. This delay meant we could not gather data for the entire rotation period, however we did record frames over a period of close to four hours.

We began to realise we had made several mistakes during the observation session when we first started to analyse the frames we had gathered. Our flat frames were not taken correctly, so were unusable and we had mistakenly identified a star as 354 Eleonora. Also the telescope tracking became progressively more offset the longer we observed. This meant manual corrections had to be repeatedly made and added to the software, slowing down our observations. We also struggled to keep the telescope pointing at the piece of sky, meaning some of the objects originally in our field of view disappeared. 354 Eleonora was, unfortunately, one of the objects to disappear from our frames. The asteroid moved out of our field of view at around the 160th asteroid frame taken. There were no more clear observing nights after our observing session, meaning we could not gather better data to analyse.

A large analysis code was written, but there was one area of it that behaved inconsistently and had a serious effect on our analysis, despite our best efforts. This meant, in some cases, that the wrong object was tracked. It also made getting the full code ready to run very difficult, causing a significant delay in obtaining results.

Simulated data was created to try and replicate real data as much as possible, at least in terms of the important parameters for this project. Frames were created that included a simulated asteroid, a star to use to correct the asteroid's light curve, and other background stars. Dark frames were also simulated. This simulated data was created to show the functioning of the analysis code. When we were finally able to run the code in its entirety, with error propagation, we realised that the wrong object was being tracked. Instead of the simulated asteroid being tracked, the code was picking out a simulated star. We were unable to fix this issue. Four light curves were still outputted and an average apparent magnitude was calculated, all including errors.

All of these factors combined mean that we only obtained a partial light curve for 354 Eleonora, so we could not determine its rotational period. We were able to determine its apparent magnitude ($m = 10.0945 \pm 1.28 \times 10^{-9}$), which was relatively close to the actual value on the night ($m = 11.0$).

Certain improvements could have been made with some better planning: each function used in the code could have been more rigorously individually tested, before being combined into one larger code. Better flat frames could have been taken during the observing session, but that would have happened if we had had a chance to take more data. Had we had more time, we could have possibly created our own *CircularAperture* and *aperture_photometry* functions, which would not have been as inconsistent, and would have helped provide the results we were looking for.

Overall, we were hampered by a lack of clear observing nights, so we could not observe the asteroid we wanted to, nor gather as much data as we wanted. However, we made several mistakes on the one observing night we had. With a bit more luck regarding the weather, these mistakes would not have been so costly.

References

[1] C. T. et al. Russell. Dawn explores ceres: Results from the survey orbit. Available at https://nesf2015.arc.nasa.gov/sites/default/files/downloads/pdf/05.pdf (2019/04/15).

[2] Gradie et al. Asteroids ii; proceedings of the conference, tucson, az, mar. 8-11, 1988. *Asteroids II.*, 12, 1989.

[3] Meghan Bartels. Japan's hayabusa2 shoots copper bomb at asteroid ryugu to create artificial crater. Available at https://www.space.com/hayabusa2-made-crater-on-asteroid-ryugu.html (2019/04/15).

[4] NASA. Light curve analysis. Available at https://www.nasa.gov/content/asteroid-grand-challenge/characterize/light-curve-analysis (2019/04/15).

[5] britastro.org. Asteroid lightcurves part i. Available at http://www.britastro.org/asteroids/Asteroid%20light%20curves%20Part%20I.htm (2019/04/15).

[6] JPL. Available at https://ssd.jpl.nasa.gov/sbdb$_q$*uery.cgix*(2019/04/15).

[7] Wilmslow Astro. Useful formulae. Available at http://www.wilmslowastro.com/software/formulae.htm (2019/04/15).

[8] Starlight Xpress. Sxvr-h16 ccd camera user manual. Available at https://www.sxccd.com/handbooks/SXVR-H16%20handbook.pdf (2019/04/15).

[9] Atik. Atik 314l+. Available at https://www.atik-cameras.com/product/atik-314l-plus/ (2019/04/15).

[10] Glasgow Astrophysics Group. The university observatories. Available at http://www.astro.gla.ac.uk/observatory/ (2019/04/15).

Statement of Work

At the start of the project, we figured out what tasks needed to be completed and split them up amongst us. I looked up ways of narrowing down the list of asteroids in order to make a shortlist. I found the JPL Small Body database and learned how to use it. I applied the appropriate filters and made a long list, a short list and a top 5 ranking, making sure the rest of the group was happy with the progress. They calculated the field of view to choose a CCD camera and calculated the exposure time once the asteroid was chosen.

We decided as a group what observing pattern to follow. At the observing night, I was in charge of checking the naming of the files and keeping track of where in the observing sequence we were. This was very important, as the analysis code requires the file numbers to be in order, without any gaps. The name of the file (object, star, dark or flat) had to be consistent too. My other group members helped calibrate the entire setup, compare what we could see in the field of view to what we could see on *Cartes du Ciel* on their laptop. They also had the keys to the dome so we could observe. There were several groups observing at the same time, so we all had small tasks.

We all took active parts in writing the code. I personally wrote the *frame_cleaner* and *correction* functions. I also structures and wrote the *command* code, which was the master notebook where the entire code was compiled. The other group members wrote the rest of the functions and created the simulated data. I did the majority of the correcting of any errors that came up, and making notes of what modifications to make to the code when switching to different data sets. I also added all the error analysis to all our functions.

After the lab sessions were finished, we regularly met up to complete the project, making sure we all had the necessary results and code.

Appendix A

Here attached is the command code, which is the master code that runs the entire analysis code in notebook.

```
import math as m
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits as fits

from photutils import DAOStarFinder
from photutils import aperture_photometry, CircularAperture
from astropy.stats import mad_std

from list_of_functions import *

## Step 0: Initialisation

frames_in_image = 10
object_image_number = 12
star_image_number = 6
dark_image_number = 7
object_frame_number = frames_in_image * object_image_number
star_frame_number = frames_in_image * star_image_number
dark_frame_number = frames_in_image * dark_image_number
flat_frame_number = 40

l_x = 1039
l_y = 1391

# Path to use to access the files ontaining the data to be processed
path = "/local/examples/a345/lab_projects/shared_data_on_cassegrain/
   asteroids2019/354_Eleonora_HD_287493_26022019/"

## Step 1: Flat Frame analysis
# Input all our flat frames and average them together, to obtain a
   single average flat frame

[sum_flat, error_sum_flat] = sum_frames(path, 'flat_', 1,
   flat_frame_number, l_x, l_y)
[avg_flat, error_flat] = avg_frames(sum_flat, error_sum_flat,
   flat_frame_number)

print('Flat␣Frame␣analysis␣complete')

## Step 2: Dark Frame analysis:
# Decide which to use

# Step 2.1: Full average:
# Input all the dark frames and average them all together

[sum_full_dark, error_sum_full_dark] = sum_frames(path, 'dark_', 1,
   dark_frame_number, l_x, l_y)
```

```
[avg_full_dark, error_full_dark] = avg_frames(sum_full_dark,
   error_sum_full_dark, dark_frame_number)

# Step 2.2: Selective average:
# For every 60 object frames and every 30 star frames, average the
   previous 10 and next 10 dark frames together

dark_image = []
error_dark_image = []
avg_sel_dark = []
error_sel_dark = []

# Create an array of 'dark images', ie an array in which every element
    contains 10 averaged dark frames

for d in range(dark_image_number):
    n = d * frames_in_image + 1
    [sum_sel_dark, error_sum_sel_dark] = sum_frames(path, 'dark_', n,
       frames_in_image, l_x, l_y)
    [avg_dark, error_avg_dark] = avg_frames(sum_sel_dark,
       error_sum_sel_dark, frames_in_image)
    dark_image.append(avg_dark)
    error_dark_image.append(error_avg_dark)

# Make an array where each element is the average of a dark image
   which the next

for e in range(dark_image_number - 1):
    avg_sel = (dark_image[e] + dark_image[e + 1]) / 2
    avg_sel_dark.append(avg_sel)
    Error_avg_sel_dark = np.square(error_dark_image[e]) + np.square(
       error_dark_image[e + 1])
    error_avg_sel_dark = 0.5 * np.sqrt(Error_avg_sel_dark)
    error_sel_dark.append(error_avg_sel_dark)

print('Dark␣Frame␣analysis␣complete')

## Step 3: Asteroid analysis:
# Run the analysis function, which cleans up each frame, then finds
   the coordinates of the asteroid in each frame, then averages
# the pixel counts for every 10 frames and obtains a time value for
   each of these average pixel counts

FWHM = 25 # Ask High Res groups for typical value, ~2-12 ish, ask
   later # 12 for real data, 25 for sim
THRESH = 22 # An estimation of how much brighter the object should be
   than the background, ~5-10

[ast_mid_times, error_ast_mid_times, avg_ast_count, error_ast_count] =
    analysis(path, 'object_', avg_sel_dark, error_sel_dark, avg_flat,
   error_flat, object_frame_number, object_image_number,
   frames_in_image, FWHM, THRESH, l_x, l_y, asteroid = 'yes')
```

```
print('Asteroid analysis complete')

## Step 4: Star analysis
# Run the analysis funciton again, but this time for the star

FWHM = 25 # Ask High Res groups for typical value, ~2-12 ish, ask
    later
THRESH = 22 # An estimation of how much brighter the object should be
    than the background, ~5-10

[star_mid_times, error_star_mid_times, avg_star_count,
    error_star_count] = analysis(path, 'star_', avg_sel_dark,
    error_sel_dark, avg_flat, error_flat, star_frame_number,
    star_image_number, frames_in_image, FWHM, THRESH, l_x, l_y,
    asteroid = 'no')

print('Star analysis complete')

## Step 5: Asteroid correction from background star
# This function corrects the asteroid pixel counts, thanks to the
    variations in the star pixel counts due to it descending in our
# field of view

[corrected_ast_count, error_corrected_ast_count] = correction(
    avg_ast_count, error_ast_count, avg_star_count, error_star_count,
    object_image_number, star_image_number)

## Step 6: Lightcurve plotting

plt.subplot(311)
plt.errorbar(ast_mid_times, avg_ast_count, xerr = error_ast_mid_times,
     yerr = error_ast_count, ecolor = 'b', linestyle = '-', color = 'b'
    )
plt.xlabel('time (s)')
plt.ylabel('pixel count')
plt.title('Lightcurve of 354 Eleonora')
plt.show()

plt.subplot(312)
plt.errorbar(star_mid_times, avg_star_count, xerr =
    error_star_mid_times, yerr = error_star_count, ecolor = 'r',
    linestyle = '-', color = 'r')
plt.xlabel('time (s)')
plt.ylabel('pixel count')
plt.title('Lightcurve of HD 287493')
plt.show()

plt.subplot(313)
plt.errorbar(ast_mid_times, corrected_ast_count, xerr =
    error_ast_mid_times, yerr = error_correted_ast_count, ecolor = 'g',
     linestyle = '-', color = 'g')
plt.xlabel('time (s)')
plt.ylabel('pixel count')
```

```
plt.title('Corrected␣Lightcurve␣of␣354␣Eleonora')
plt.show()

## Step 7: Apparent Magnitudes

h = 6.626 * 10**(-34)
nu_0 = 4.44 * 10**(14)
eta = 0.28
A = 0.125
CST1 = (h*nu_0)/(eta*A)

F_Vega = 2.178 * 10**(-8)
err_F_Vega = 0.0244 * 10**(-8)
CST2 = (err_F_Vega/F_Vega)**2
m_Vega = 0.026

tau = 12.09
err_tau = 0.13
CST3 = (err_tau/tau)**2

m_ast = -2.5 * np.log10(CST1 * corrected_ast_count/(tau * F_Vega)) +
   m_Vega

err_m_ast = -2.5 * (tau * F_Vega / m.log(10)) * np.sqrt(np.square(np.
   divide(error_corrected_ast_count,corrected_ast_count))+CST2+CST3)


plt.errorbar(ast_mid_times, m_ast, xerr = error_ast_mid_times, yerr =
   err_m_ast, ecolor = 'g', color = 'g', marker = '+', linestyle = '
   none')
plt.xlabel('time␣(s)')
plt.ylabel('Apparent␣magnitude')
plt.title('Corrected␣Lightcurve␣of␣354␣Eleonora')
plt.show()

sum_m = 0
Error_sum_m = 0

for j in range(len(m_ast)):
    sum_m += m_ast[j]
    Error_sum_m += err_m_ast[j]**2

avg_m = sum_m / (len(m_ast))
error_avg_m = np.sqrt(Error_sum_m) / (len(m_ast))

print('Average␣Apparent␣Magnitude␣=␣' + str(avg_m) + '␣+/-␣' + str(
   error_avg_m))
```

Appendix B

Here is the contents of the *list_of_functions* notebook, which contains all the functions called in the command code.

```
import math as m
import numpy as np
import matplotlib.pyplot as plt
from astropy.io import fits as fits

from photutils import DAOStarFinder
from photutils import aperture_photometry, CircularAperture
from astropy.stats import mad_std

def sum_frames (path, object_name, n, frames_in_image, l_x, l_y):
# Sums inputted frames into one variable. n represents the starting
   number to be called

    summation = 0
    Error = 0

    for j in range(n, n + frames_in_image):
        data = fits.getdata(path + object_name + str(j) + '.fit')
        Error += data
        summation += data

    error = np.sqrt(Error)

    return summation, error

def avg_frames(summation, error, frames_in_image):
# Averages an inputted array

    average_frame = summation / frames_in_image
    error_average = error / frames_in_image

    return average_frame, error_average

def analysis(path, object_name, dark_frame, error_dark,
   object_frame_number, object_image_number, frames_in_image, FWHM,
   THRESH, l_x, l_y, asteroid):

    # Frame cleaning:
    # For each frame, remove the appropriate dark frame and divide by
       the average flat frame
    object_frame_clean = []
    error_object_frame_clean = []
    a = 0

    for o in range(1, object_frame_number + 1):
        [cleaned_frame, error_cleaned_frame] = frame_cleaner(path,
           object_name, o, dark_frame[a], error_dark[a], l_x, l_y)
        object_frame_clean.append(cleaned_frame)
```

```
    error_object_frame_clean.append(error_cleaned_frame)
    if o % 60 == 0:
        a += 1

print('Frame cleaning complete')

# Find the coordinates of the object in each frame:
object_x = []
object_y = []
object_count = []
error_object_count = []

# Make an array of x coordinates, y coordinates and corresponding
   pixel counts for each light source detected, then pick the
# correct light source for the object (asteroid or star)

for o in range(1, object_frame_number + 1):
    [x_coordinates, y_coordinates, pixel_count, error_pixel_count]
        = centroids(object_frame_clean[o-1],
       error_object_frame_clean[o-1], FWHM, THRESH)
    if o == 1:
        if object_name == 'object_':
            x = 400.00
            y = 600.00
            [correct_x, correct_y, correct_count,
               error_correct_count] = new_coord(x, y,
               x_coordinates, y_coordinates, pixel_count,
               error_pixel_count)
        if object_name == 'star_':
            x = 300.00
            y = 420.00
            [correct_x, correct_y, correct_count,
               error_correct_count] = new_coord(x, y,
               x_coordinates, y_coordinates, pixel_count,
               error_pixel_count)
    else:
        [correct_x, correct_y, correct_count, error_correct_count]
            = new_coord(object_x[o-2], object_y[o-2],
           x_coordinates, y_coordinates, pixel_count,
           error_pixel_count)

    object_x.append(correct_x)
    object_y.append(correct_y)
    object_count.append(correct_count)
    error_object_count.append(error_correct_count)

print('Object extracted')

# Average pixel counts for every 10 frames to get an array of
   average pixel counts:
avg_object_count = []
error_avg_object_count = []
```

```
    for h in range(object_image_number):
        sum_count = 0
        error_sum_count = 0

        n = h * frames_in_image
        for j in range(n, n + frames_in_image):
            sum_count += object_count[n]
            error_sum_count += error_object_count[n]

        avg_count = sum_count / frames_in_image
        avg_object_count.append(avg_count)
        error_avg_count = error_sum_count / frames_in_image
        error_avg_object_count.append(error_avg_count)

    # Extract Timestamp from each image and obtain an "average" time
       for each 10-frame interval:
    [object_mid_times, error_mid_times] = get_time(path, object_name,
       object_frame_number)

    print('Timestamps extracted')

    return object_mid_times, error_mid_times, avg_object_count,
       error_avg_object_count


def frame_cleaner(path, object_name, n, dark_frame, error_dark, l_x,
   l_y):
# Removes the inputted dark frame from the inputted object (asteroid
   or star) frame and divides by the average flat frame

    data = fits.getdata(path + object_name + str(n)+ '.fit')
    object_frame = data
    error_object_frame = np.sqrt(data)

    object_frame_clean = np.zeros((l_x,l_y))
    error_object_frame_clean = np.zeros((l_x,l_y))

    for j in range(l_x):
        for k in range(l_y):
            object_frame_clean[j,k] = object_frame[j,k] - dark_frame[j
               ,k]

            Error_object_frame_clean = np.square(error_object_frame[j,
               k]) + np.square(error_dark[j,k])
            error_object_frame_clean = np.sqrt(
               Error_object_frame_clean)

    return object_frame_clean, error_object_frame_clean


def centroids(object_frame_clean, error_object_frame_clean, FWHM,
   THRESH):
```

```
# This function takes the inputted cleaned-up object (asteroid or
   star) frame, the FWHM and a threshold value as inputs and
# returns two tables: one with a list of centroids and their
   characteristics, and the other with a list of centroids with
   the
# sum of pixel counts within the circle

# Centroids:

Im = object_frame_clean

bkg_sigma = mad_std(Im)                 # Calculates a robust
   standard deviation, whatever that means
daofind = DAOStarFinder(fwhm = FWHM, threshold = THRESH*bkg_sigma)
       # Finds sources, FWHM determined by the user (units of
   pixels), the threshold determined by the user as a multible of
   the standard deviation of the matrix
sources = daofind(Im)                   # not sure what's happening
   at this step...

for col in sources.colnames:             # yeah... don't know what
   that's for either, so don't touch it I guess.
      sources[col].info.format = '%.8g'  # For consistent table
         output

#Circles:

r = FWHM  # Radius of the aperture circles around our light
   sources
error = error_object_frame_clean

positions = (sources['xcentroid'], sources['ycentroid'])       #
   Creates a list of arrays, where the first elmt is the x coords
   of the centroids, and the 2nd is the y coords
apertures = CircularAperture(positions, r)              # Creates
   apertures (the blue circles) at the position of the detected
   sources, with a radius defined by the user (use same as FWHM)
phot_table = aperture_photometry(Im, apertures, error)
                 # Takes an image and apertures as inputs, counts
    the number of photons contained within each aperture, and
   returns the results in a table
for col in phot_table.colnames:
     phot_table[col].info.format = '%.8g'  # For consistent table
        output

# Table Extraction:
# Create three arrays, one with the x coordinates of all the
   centroids, another with all the y coordinates and the third one
# with the sum of the pixel counts inside each circle around a
   centroid

x_coordinates = np.array(phot_table['xcenter'])
y_coordinates = np.array(phot_table['ycenter'])
```

```
    pixel_count = np.array(phot_table['aperture_sum'])
    error_pixel_count = np.array(phot_table['aperture_sum_err'])

    return x_coordinates, y_coordinates, pixel_count,
       error_pixel_count


def new_coord(previous_x, previous_y, x_coordinates, y_coordinates,
   pixel_count, error_pixel_count):
# Inputs are the coordinates of the object in the previous picture,
   and the array with the centroid positions and counts in the
# new picture

    coord_i = [previous_x, previous_y]
    array = [x_coordinates, y_coordinates]
    n = x_coordinates.shape[0]
    coord_f = [0,0]              # set the new coord with a value sure
        to be too fare away to the new object position

    for i in range(n):                    # loop over the number of
       centroids
        d1 = m.sqrt(np.square(coord_i[0]-coord_f[0]) +np.square(
           coord_i[1]-coord_f[1]))          # distance between the old
            coord and the new coord
        d2 = m.sqrt(np.square(coord_i[0]-array[0][i])+np.square(
           coord_i[1]-array[1][i]))  # distance between the old coord
           and the ith centroid
        if d2<d1:                # check if the ith centroid is closer
           to the object than the current
            coord_f = [array[0][i], array[1][i]]  # if so replace
            correct_count = pixel_count[i]
            error_correct_count = error_pixel_count[i]

    correct_x = coord_f[0]
    correct_y = coord_f[1]

    return correct_x, correct_y, correct_count, error_correct_count


def get_time(path, object_name, n):
# The function finds the timestamp for each object (asteroid or star)
   frame and extracts the time converted into seconds, with
# the first file given time 0. Then the function determines a median
   time for every packet of 10 frames

    array = []

    for j in range (1, n + 1):

        objectt = fits.open(path + object_name + str(j) + ".fit")
        hdr = objectt[0].header['DATE-OBS'] #reading the date of the
           observation for the 360 images of the object
```

```
            seconds = int(hdr[17:20])
            minutes = int(hdr[14:16])
            hours = int(hdr[11:13])

            # Correction for change of day during observation

            if j == 1:
                reference_date = int(hdr[8:10])
            elif int(hdr[8:10]) != reference_date:
                hours += 24

            time = 3600 * hours + 60 * minutes + seconds
            array.append(time)

    error_time = 1
    error_time_avg = 0.5 * m.sqrt(2) * error_time

    initial_time = time[0]  # Save the starting time value

    m = n / 10
    mid_times = []
    error_mid = []
    error_mid_times = []

    for i in range(int(m)):
        ti = array[i * 10]
        tf = array[i * 10 + 9]
        t_avg = (tf - ti) / 2

        mid_time = ti + t_avg
        Error_mid_time = np.square(error_time) + np.square(
           error_time_avg)
        error_mid_time = np.sqrt(Error_mid_time)
        mid_times.append(mid_time)
        error_mid.append(error_mid_time)

    for t in range(len(mid_times)):
        mid_times[t] -= initial_time
        Error_no_reference = np.square(error_mid[t]) + np.square(
           error_time)
        error_no_reference = np.sqrt(Error_no_reference)
        error_mid_times.append(error_no_reference)

    return mid_times, error_mid_times


def correction(avg_ast_count, error_avg_ast_count, avg_star_count,
   error_avg_star_count, object_image_number, star_image_number):
# The function corrects the array of asteroid pixel counts thanks to
   the variations in the star pixel counts which shouldn't be
# be present if there was no astmosphere to look through

    reference = avg_star_count[0]
```

```
error_reference = error_avg_star_count[0]
multipliers = []
error_multipliers = []
corrected_ast_count = np.zeros(object_image_number)
error_corrected_ast_count = []

for j in range(len(avg_star_count)):
    ratio = reference / avg_star_count[j]
    multipliers.append(ratio)

    Error_ratio = np.square(error_reference / reference) + np.
       square(error_avg_star_count[j] / avg_star_count[j])
    error_ratio = ratio * np.sqrt(Error_ratio)
    error_multipliers.append(error_ratio)

for m in range(star_image_number):
    corrected_ast_count[2*m] = avg_ast_count[2*m] * multipliers[m]
    corrected_ast_count[2*m + 1] = avg_ast_count[2*m + 1] *
       multipliers[m]

    Error_correction = np.square(error_avg_ast_count[2*m] /
       avg_ast_count[2*m]) + np.square(error_multipliers[m] /
       multipliers[m])
    error_correction = corrected_ast_count[2*m] * np.sqrt(
       Error_correction)
    error_corrected_ast_count.append(error_correction)

    Error_correction = np.square(error_avg_ast_count[2*m + 1] /
       avg_ast_count[2*m + 1]) + np.square(error_multipliers[m] /
       multipliers[m])
    error_correction = corrected_ast_count[2*m + 1] * np.sqrt(
       Error_correction)
    error_corrected_ast_count.append(error_correction)

return corrected_ast_count, error_corrected_ast_count
```

Appendix C

In this Appendix is the code used to create the simulated data.

```
import math as m
import numpy as np
import matplotlib.pyplot as plt
import random
from astropy.table import Table
from photutils.datasets import make_gaussian_sources_image
from collections import OrderedDict
from photutils.datasets import make_random_gaussians_table
from astropy.io import fits as f

from photutils import DAOStarFinder
from photutils import aperture_photometry, CircularAperture
from astropy.stats import mad_std
import scipy.misc
import datetime
import time

def sim_object(Dim,Coord,Amplitude,d_object):

# This function takes as input the dimensions of the picture the
   object is going to be in, its coordinates, its amplitude and the
   standard deviation in pixels.
#  It returns a matrix containing the values for the object, and zeros
    elsewhere

    table_object = Table()
    table_object['amplitude'] = [Amplitude]
    table_object['x_mean'] = [Coord[0]]
    table_object['y_mean'] = [Coord[1]]
    table_object['x_stddev'] = [d_object]
    table_object['y_stddev'] = [d_object]

    Objet = make_gaussian_sources_image(Dim,table_object)

    return Objet

def sec_to_h(sec):
    if sec<3600:
        h=0
        if sec<60:
            m=0
            s= sec
        else:
            m = sec//60
            s = sec%60
    else:
        h=sec//3600
        m = (sec-h*3600)//60
        s = (sec-h*3600)%60
```

```
def set_of_pics():

    Dim = [0,0]
    ast_coord_i = [0,0]
    ast_coord_f = [0,0]
    star_coord = [0,0]

    # Pic
    Dim[0] = int(input('Hello␣User\nVertical␣dimension␣of␣the␣picture␣
       (px):\n'))
    Dim[1] = int(input('Horizontal␣dimension␣of␣the␣picture␣(px):\n'))
    N_frames = int(input('Number␣of␣frames␣in␣pictures:\n'))
    N_pics = int(input('Number␣of␣pictures:\n'))
    N_dark_frames = int(input('Number␣of␣dark␣frame␣pictures:\n'))

    # Background stars
    N_stars = int(input('Number␣of␣stars␣in␣the␣background:\n'))
    max_amp_stars = int(input('Maximum␣amplitude␣of␣the␣centre␣of␣the␣
       background␣stars␣(count):\n'))
    max_d_stars = int(input('Maximum␣width␣of␣the␣background␣stars␣(px
       ):\n'))

    # Noise
    SNR = int(input('Ratio␣of␣the␣maximum␣amplitude␣of␣the␣background␣
       stars␣over␣the␣mean␣of␣the␣noise:\n'))
    Stdev_i = float(input('Standard␣deviation␣of␣the␣noise␣at␣the␣
       beginning␣of␣observation,␣expressed␣as␣a␣fraction␣of␣the␣mean:\
       n'))
    Stdev_f = float(input('Standard␣deviation␣of␣the␣noise␣at␣the␣end␣
       of␣observation,␣expressed␣as␣a␣fraction␣of␣the␣mean:\n'))

    # Asteroid parameters
    T = float(input('Rotation␣period␣of␣the␣asteroid␣(hours):\n'))
       *60*60

    ast_coord_i[0] = int(input('X␣coordinate␣of␣the␣asteroid␣at␣the␣
       beginning␣of␣observation:\n'))
    ast_coord_i[1] = int(input('Y␣coordinate␣of␣the␣asteroid␣at␣the␣
       beginning␣of␣observation:\n'))
    ast_coord_f[0] = int(input('X␣coordinate␣of␣the␣asteroid␣at␣the␣
       end␣of␣observation:\n'))
    ast_coord_f[1] = int(input('Y␣coordinate␣of␣the␣asteroid␣at␣the␣
       end␣of␣observation:\n'))

    max_amp_ast = int(input('Amplitude␣of␣the␣centre␣of␣the␣asteroid␣
       when␣its␣luminosity␣is␣at␣its␣maximum␣(count):\n'))
    d_ast = int(input('Width␣of␣the␣asteroid␣(px):\n'))
    delta_amp_ast = float(input('Amplitude␣of␣the␣variation␣in␣
       luminosity␣of␣the␣asteroid,␣expressed␣as␣a␣fraction␣of␣its␣
       maximum:\n'))

    # Star parameters
```

```
    star_coord[0] = int(input('X coordinate of the star:\n'))
    star_coord[1] = int(input('Y coordinate of the star:\n'))

    d_star = int(input('Width of the star (px):\n'))
    amp_star_i = int(input('Amplitude of the centre of the star at the
       beginning of observation(count):\n'))
    amp_star_f = int(input('Amplitude of the centre of the star at the
       end of observation(count):\n'))
    # this is the end of user input

    # Making the flat frames
    # for i in N_frames:

    # Make the background stars
    param_ranges = [('amplitude', [0.5*(max_amp_stars/SNR),
       max_amp_stars]),
                    ('x_mean', [0, Dim[1]]),
                    ('y_mean', [0, Dim[0]]),
                    ('x_stddev', [1, max_d_stars]),
                    ('y_stddev', [1, max_d_stars])]
                    # ('theta', [0, np.pi])]
    param_ranges = OrderedDict(param_ranges)
    table_sources = make_random_gaussians_table(N_stars, param_ranges)
             # these lines create the BG stars usingthe parameters
       defined by the user

    Sources = make_gaussian_sources_image((Dim[0],Dim[1]),
       table_sources)  # the mat sources contains the BG stars

    print('Background stars generated...\n')

    tot_frames = N_frames*(N_pics + N_dark_frames)  # determining the
       total amount of frames for the observation, not including the
       flat fields
    DF_index = 1  # initialising the pic indices for the naming format
    Pic_index = 1

    Counter = 1

# Determine the number of loops of {DF Pic ...Pic} to be repeated

    if N_pics%(N_dark_frames-1)==0:          # best case scenario
        N_loops = N_dark_frames-1
        N_frames_in_loop = N_frames * (1+N_pics/N_loops)

        for i in range(int(N_loops)):        # looping through sets of
           {DF Pic Pic.... Pic}
            for j in range(int(N_frames_in_loop)):  # now looping
               through the number of frames in a loop
                percent = (Counter/tot_frames)*100
                my_str = f"{percent}%..."
                print(my_str)
                Counter+=1
```

```
Stdev = ((Stdev_f-Stdev_i)/tot_frames) * (i*
   N_frames_in_loop + j) + Stdev_i # the further we go
    the more noise there will be as  we assume the
   temperature of the CCD increases
Noise = np.random.normal(max_amp_stars/SNR,Stdev*(
   max_amp_stars/SNR),Dim)  # dark frames will be used
    in every frame

if j<N_frames:  # this if loop takes care of creating
   and saving dark frames
    DF_name = 'dark_frame_' + str(DF_index) + '.fit'
    DF_index+=1
    hdu = f.PrimaryHDU()
    hdu.data = Noise
    thetime = sec_to_h(12*(i*N_frames_in_loop + j))
    currentDT = datetime.datetime(2019,4,15,thetime
       [0],thetime[1],thetime[2])
    hdu.header['DATE-OBS'] = currentDT.strftime("%Y-%m
       -%dT%H:%M:%S")
    hdu.writeto(DF_name, overwrite=True)


else:
    thetime = sec_to_h(12*(i*N_frames_in_loop + j))
    currentDT = datetime.datetime(2019,4,15,thetime
       [0],thetime[1],thetime[2])
    t =  currentDT.strftime("%Y-%m-%dT%H:%M:%S")
    seconds = int(t[17:20])
    minutes = int(t[14:16])
    hours = int(t[11:13])


    temps = (i*N_frames_in_loop + j)*12

    ast_coord = [ast_coord_i[0] + (abs((ast_coord_f
       [0]-ast_coord_i[0]))/tot_frames)*(i*
       N_frames_in_loop + j), ast_coord_i[1]+(abs((
       ast_coord_f[1]-ast_coord_i[1]))/tot_frames)*(i*
       N_frames_in_loop + j)]  # the coord of the ast
       change with time

    amp_ast = max_amp_ast - max_amp_ast *
       delta_amp_ast *(1- m.cos(2*m.pi*temps/T))
       # the amplitude of the ast varies with time

    amp_star = amp_star_i - (abs((amp_star_f-
       amp_star_i))/tot_frames)*(i*N_frames_in_loop +
       j)  # the star dims as we go along
    amp_ast_dimmed = amp_ast * (amp_star/amp_star_i)
             # the ast is dimmed as much as the star

    Ast = sim_object(Dim,ast_coord,amp_ast_dimmed,
       d_ast)        # woohoo we've got the asteroid
```

```
                Star = sim_object(Dim,star_coord,amp_star,d_star)
                            # and the star

                Pic = Noise + Sources + Ast + Star      # create
                   mat for the picture

                Pic_name = 'Pic_' + str(Pic_index) + '.fit'
                Pic_index+=1
                hdu = f.PrimaryHDU()
                hdu.data = Pic
                hdu.header['DATE-OBS'] = t
                hdu.writeto(Pic_name, overwrite=True)



    for i in range(N_frames):      # this is the last set of dark
       frames
            Stdev = ((Stdev_f-Stdev_i)/tot_frames) * (N_loops*
               N_frames_in_loop + i) + Stdev_i
            Noise = np.random.normal(max_amp_stars/SNR,Stdev*(
               max_amp_stars/SNR),Dim)

            DF_name = 'dark_frame_' + str(DF_index) + '.fit'
            DF_index+=1
            hdu = f.PrimaryHDU()
            hdu.data = Noise
            thetime = sec_to_h(12*(i*N_frames_in_loop + j))
            currentDT = datetime.datetime(2019,4,15,thetime[0],
               thetime[1],thetime[2])
            hdu.header['DATE-OBS'] = currentDT.strftime("%Y-%m-%dT
               %H:%M:%S")
            hdu.writeto(DF_name, overwrite=True)



else:           # worse case scneario, the last loop will have
   less pics in it
    N_loops = N_dark_frames-2
    pics_in_last_loop = N_pics%(N_dark_frames-2)  # /!\ /!\ /!\
       this line needs to be corrected

    N_frames_in_loop = N_frames * (1+(N_pics-pics_in_last_loop)/
       N_loops)

    for i in range(int(N_loops)):       # looping through sets of
       {DF Pic Pic.... Pic}
        for j in range(int(N_frames_in_loop)):  # now looping
           through the number of frames in a loop
            Stdev = ((Stdev_f-Stdev_i)/tot_frames) * (i*
               N_frames_in_loop + j) + Stdev_i # the further we go
                the more noise there will be as  we assume the
               temperature of the CCD increases
```

```
Noise = np.random.normal(max_amp_stars/SNR,Stdev*(
   max_amp_stars/SNR),Dim)  # dark frames will be used
    in every frame

if j<N_frames:  # this if loop takes care of creating
   and saving flat frames
    DF_name = 'dark_frame_' + str(DF_index) + '.fit'
    DF_index+=1
    hdu = f.PrimaryHDU()
    hdu.data = Noise
    thetime = sec_to_h(12*(i*N_frames_in_loop + j))
    currentDT = datetime.datetime(2019,4,15,thetime
       [0],thetime[1],thetime[2])
    hdu.header['DATE-OBS'] = currentDT.strftime("%Y-%m
       -%dT%H:%M:%S")
    hdu.writeto(DF_name, overwrite=True)

else:
    thetime = sec_to_h(12*(i*N_frames_in_loop + j))
    currentDT = datetime.datetime(2019,4,15,thetime
       [0],thetime[1],thetime[2])
    t =  currentDT.strftime("%Y-%m-%dT%H:%M:%S")
    seconds = int(t[17:20])
    minutes = int(t[14:16])
    hours = int(t[11:13])


    temps = (i*N_frames_in_loop + j)*12

    ast_coord = [ast_coord_i[0] + (abs((ast_coord_f
       [0]-ast_coord_i[0]))/tot_frames)*(i*
       N_frames_in_loop + j), ast_coord_i[1]+(abs((
       ast_coord_f[1]-ast_coord_i[1]))/tot_frames)*(i*
       N_frames_in_loop + j)]  # the coord of the ast
       change with time

    amp_ast = max_amp_ast - 2*delta_amp_ast * m.cos(2*
       m.pi*temps/T)      # the amplitude of the ast
       varies with time

    amp_star = amp_star_i - (abs((amp_star_f-
       amp_star_i))/tot_frames)*(i*N_frames_in_loop +
       j)  # the star dims as we go along
    amp_ast_dimmed = amp_ast * (amp_star/amp_star_i)
             # the ast is dimmed as much as the star

    Ast = sim_object(Dim,ast_coord,amp_ast_dimmed,
       d_ast)        # woohoo we've got the asteroid
    Star = sim_object(Dim,star_coord,amp_star,d_star)
                # and the star

    Pic = Noise + Sources + Ast + Star      # create
       mat for the picture
```

```
            Pic_name = 'Pic_' + str(Pic_index) + '.fit'
            Pic_index+=1
            hdu = f.PrimaryHDU()
            hdu.data = Pic
            hdu.header['DATE-OBS'] = t
            hdu.writeto(Pic_name, overwrite=True)



for i in range(int(N_frames*(2+Pics_in_last_loop))):
    if i<N_frames:
        Stdev = ((Stdev_f-Stdev_i)/tot_frames) * (N_loops*
           N_frames_in_loop + i) + Stdev_i
        Noise = np.random.normal(max_amp_stars/SNR,Stdev*(
           max_amp_stars/SNR),Dim)

        DF_name = 'dark_frame_' + str(DF_index) + '.fit'
        DF_index+=1
        hdu = f.PrimaryHDU()
        hdu.data = Noise
        thetime = sec_to_h(12*(i*N_frames_in_loop + j))
        currentDT = datetime.datetime(2019,4,15,thetime[0],
           thetime[1],thetime[2])
        hdu.header['DATE-OBS'] = currentDT.strftime("%Y-%m-%dT
           %H:%M:%S")
        hdu.writeto(DF_name, overwrite=True)


    elif i<N_frames*(2+Pics_in_last_loop)-N_frames:
        thetime = sec_to_h(12*(i*N_frames_in_loop + j))
        currentDT = datetime.datetime(2019,4,15,thetime[0],
           thetime[1],thetime[2])
        t =  currentDT.strftime("%Y-%m-%dT%H:%M:%S")
        seconds = int(t[17:20])
        minutes = int(t[14:16])
        hours = int(t[11:13])

        temps = (i*N_frames_in_loop + j)*12

        ast_coord = [ast_coord_i[0] + (abs((ast_coord_f[0]-
           ast_coord_i[0]))/tot_frames)*(N_loops*
           N_frames_in_loop + i), ast_coord_i[1]+(abs((
           ast_coord_f[1]-ast_coord_i[1]))/tot_frames)*(
           N_loops*N_frames_in_loop + i)]

        amp_ast = max_amp_ast - 2*delta_amp_ast * m.cos(2*m.pi
           *temps/T)

        amp_star = amp_star_i - (abs((amp_star_f-amp_star_i))/
           tot_frames)*(N_loops*N_frames_in_loop + i)

        amp_ast_dimmed = amp_ast * (amp_star/amp_star_i)
```

```
            Ast = sim_object(Dim,ast_coord,amp_ast_dimmed,d_ast)
            Star = sim_object(Dim,star_coord,amp_star,d_star)

            Pic = Noise + Sources + Ast + Star

            Pic_name = 'Pic_' + str(Pic_index) + '.fit'
            Pic_index+=1
            hdu = f.PrimaryHDU()
            hdu.data = Pic
            hdu.header['DATE-OBS'] = t
            hdu.writeto(Pic_name, overwrite=True)



        else:
            Stdev = ((Stdev_f-Stdev_i)/tot_frames) * (N_loops*
               N_frames_in_loop + i) + Stdev_i
            Noise = np.random.normal(max_amp_stars/SNR,Stdev*(
               max_amp_stars/SNR),Dim)

            DF_name = 'dark_frame_' + str(DF_index) + '.fit'
            DF_index+=1
            hdu = f.PrimaryHDU()
            hdu.data = Noise
            thetime = sec_to_h(12*(i*N_frames_in_loop + j))
            currentDT = datetime.datetime(2019,4,15,thetime[0],
               thetime[1],thetime[2])
            hdu.header['DATE-OBS'] = currentDT.strftime("%Y-%m-%dT
               %H:%M:%S")
            hdu.writeto(DF_name, overwrite=True)



print("Done.")
```