PHYC30170: Advanced Laboratories

Astronomical Image Analysis

January 17, 2021

Author: Owen Johnson
Student ID: 18305971
Supervisor: Prof. Morgan Fraser

UCD School of Physics
Scoil na Fisice UCD

Abstract

A review of scientific data provided by the IAC80 telescope on Messier 91 and NN Serpentis. Calibration and reduction were performed to each image before the analysis took place. The apparent physical size of the Messier 91 galaxy was found to be $(24.8 \times 19.1) \pm 0.5\ kpc$ and the duration of an eclipse of the binary NN Serpentis system was found to be 11.6 ± 2.2 minutes.

Contents

1 Introduction

This investigation aims to observe and calibrate a set of scientific images taken by IAC80 telescope at Teide Observatory in Tenerife. The telescope took images of two astronomical objects, Messier 91 (M_{91}) and NN Serpentis (NNSer), after the calibration or 'reduction' of the scientific images analysis, can be carried out on various attributes of each object.
In the case of M_{91} this investigation uses combined images from multiple colour filters to determine the angular size and for NNSer photometry is used to determine the time duration of the eclipse in the binary system by plotting of a light-curve.

2 Theory

2.1 Messier 91 (M_{91})

Messier 91 is a barred spiral galaxy located in the Coma Berenices constellation. It is approximately 63 million light-years away and lies in the Virgo cluster. The apparent dimensions (angular size) of the galaxy is approximately 5.4 x 4.4 (arc min), and it has a linear diameter of approximately 100,000 light-years.
The galaxy was discovered by French astronomer Charles Messier in 1781 along with numerous other messier objects. (9) (10) (14)

2.2 NN Serpentis

NN Serpentis is an eclipsing post-common envelope binary system. The system consists of a white dwarf and a red dwarf, with the stars orbiting each other every 3.12 hours. It is approximately 1670 light-years away. The 'NN' used in the name denotes the variability in the star's brightness over time; this is due to the eclipse that occurs ever 3 hours between the two stars.
It has also been thought that there is an inferred planetary system present in NN Ser by various researchers' teams. However, further observations need to occur before this can be deemed conclusive, and the system itself better understood. (7) (17) (?) (?)

2.3 Astronomical Techniques

2.3.1 Telescope Filters

When making observations with telescopes colour filters are used to obtain different areas of the visible spectrum when looking at the same object. Each filter observes a specific range

of wavelengths. This is due to how the CCD sensor operates; it counts the incident photons but does not register the photon's energy, and thus colour is not obtained. Multiple filters can be used in conjunction to create scientific images such that colour information can be analysed. (5)

In the case of this investigation of M_{91} uses B, $H\alpha$ and V filters and a 'clear' filter is used for NN Ser. Each filter corresponds to a different range of optical wavelengths. Filters are chosen based on their characteristics and the wavelengths they reveal on the spectrum. They are often used against each other to obtain a colour index. (10)

The reason for varying the filters when observing M_{91} is how galaxies emit their light. Their sources vary to create an almost continuous spectrum. This can be from objects such as stars and nebulae. Using a filter, the various parts of the spectrum are isolated, so different parts of the galaxy are more vivid and distinct and thus easier to observe. This is best illustrated in fig. 7. (18)

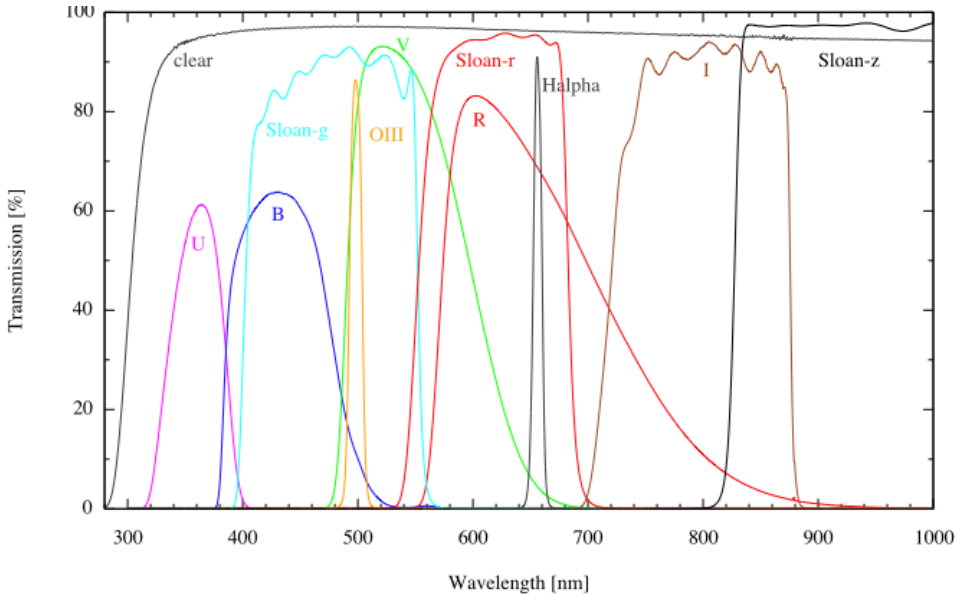


Figure 1 *Comparison of various filters transmission against wavelength* [1]

Filter	**Effective Wavelength** (λ_{eff})	**Bandwidth** (λ_b)	**Description**
B	445 nm	66 nm	Blue light.
V	551 nm	88 nm	Visual light.
H Alpha	656 nm	0.07 nm	Deep-red visible.

2.3.2 Photometry

To determine the length of an eclipse in the NN Ser system, photometry needs to be first used. Photometry is a scientific field specifically dedicated to the measurement of light. This is done with a comparison of bright-stars calibrated against standard stars.
Throughout this investigation, aperture photometry is used to measure the counts of sources. This is done with using the following equation. (8)

$$m_{std} = -2.5 \log_{10}(F) + ZP \tag{1}$$

Where m_{std} is the calibrated magnitude of a source in the standard system, F is the background=subtracted counts from the source, and ZP is the zero-point for the image. In calculating zero points, each image is calibrated to a standard system compared with other images of the same object taken by different telescopes.

The counts for the source in each image are determined using astropy and photoutils (see section 4.3). An aperture and annulus must be chosen. An aperture is selected to contain the most light from the chosen source. The annulus is selected to represent the local background and isolated from the chosen light source (NN Ser). Using photoutils in conjunction with eq. (1) counts and magnitudes can be determined for the chosen source in a particular image.

However, for zero points to be obtained, it is necessary to have sources of known magnitude present within the image, these sources are known as a local photometric sequence. Then eq. (1) can be arranged so that a zero-point can be obtained. Then eq. (1) can be arranged so that a zero-point can be obtained.

$$ZP = 2.5 \log_{10}(F) + m_{std} \tag{2}$$

For each reference source in each image, a zero-point can be found, such that an average zero-point for each image can be calculated and applied to the source of interest. This is illustrated and discussed further in section 4.3. (8)

2.4 Using Light Curves to find Eclipse Duration

A light-curve is the plotting of a star's brightness over time. When plotted when observing a binary system like NN Ser, variation in the star's brightness will dip periodically as the smaller star is eclipsed by, the bigger star (see fig. 17).

Transit photometry can be used to determine things like size, orbit shape, temperature and surface features. When observing a binary system, the stars are too close together to have distinct, separate brightness so the light curve forms from the variation in the system's total brightness.

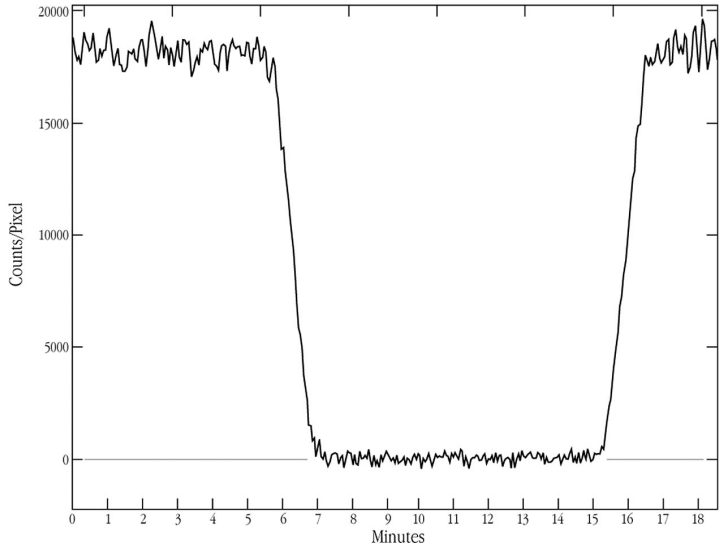


Figure 2 *Light curve when observing NN Ser produced by the European Southern Observatory (12)*

As seen from the above figure (fig. 2) there is a distinct dip. The width of this trough is an estimate of the duration of the eclipse. (13)

2.5 Determining the Angular Size for Messier 91 (M_{91})

When calculating the angular size of M_{91} , multiple factors need to be considered first. The first is the shape of M_{91} , as previously mentioned it is a barred spiral galaxy so its elliptical shape is irregular and non-uniform compared to a regular ellipse like the one seen in fig. 3. This needs to be considered when calculating the angular size which is discussed further in section 5.2

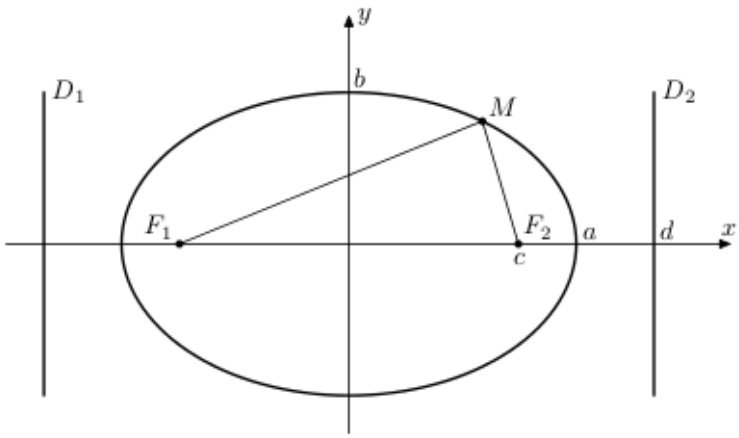


Figure 3 *Diagram of conventional ellipse.*

The angle which an object is viewed at in the sky affects the angular size perceived. This means that the angle of inclination that the object is viewed needs to be considered when calculating the angular size. This is illustrated in fig. 4.

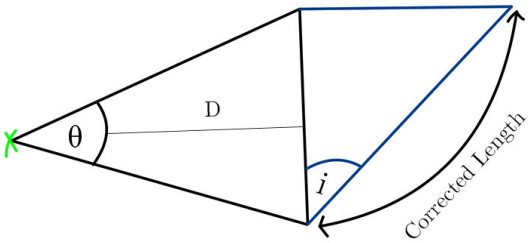


Figure 4 *Diagram illustrating the effects of viewing angle on an object.*

The correction can be performed using simple trigonometry if the angle of inclination (i) and distance (D) to the object is known. Given that angular size, (θ) is calculated, the correction is performed using the following sets of equations where the semi-major or semi-minor axis is denoted by, ω.

$$\omega = \frac{\theta}{2} \tag{3}$$

$$\mathscr{D} = \frac{D \tan(\omega)}{\cos(i)} \tag{4}$$

Where $\mathscr{D}$ is the corrected semi-major or semi-minor axis. (1) (8)

3 Apparatus

All images were taken using the IAC80 telescope in Teide Observatory in Tenerife. All raw data, including the calibration images, was provided in a .fits image format. It was then calibrated and analysed using python and an array of associated packages—the code for which can be found in appendix B.

The IAC80 telescope developed by the Instituto de Astrofísica de Canarias uses a CAMELOT[2] sensor. The CAMELOT sensor is a wide-angle, so it has a pixel array of 2048 $\times$ 2048. The CAMELOT also has four readout channels and is highly sensitive within the optical range. (2)

4 Experimental Methodology

4.1 Calibration and Reduction

Before analysis can take place on the science images, they must be calibrated to account for various noise or imperfections that may cause the image to be obscured. For this investigation, both electronic noise and imperfections in the CCD sensor must be accounted. This is done by constructing a master bias (electronic noise) and normalised master flat (sensor imperfection).

This is done by taking all calibration images for each respective calibration frame and averaging them. The example code for how this done can be found in listing 4 in appendix B.2. Once each calibration frame (bias and flat) have been averaged, the master frames are saved.

[2]CAmara MEjorada LIgera del Observatorio del Teide (CAMELOT)

In the case of the master flat it must be normalised. This is due to imperfections in the sensor having a multiplicative effect on each pixel the sensor detects. Therefore the scientific images must have the master bias subtracted and are then divided by the normalised master flat to negate the noise and imperfections. An example of the code used to complete this task can be found in listing 6 in appendix B.2. It is better illustrated by the operations that occur to the counts in the scientific images in the equation below (15) (11) (8),

$$\text{reduced image} = \frac{\text{orginal image} - \text{master bias}}{\text{normalised master flat}} \tag{5}$$

4.1.1 M_{91} Calibration and Reduction

In the case of M_{91} , three filters were used when obtaining scientific images: a B filter, $H\alpha$ filter and a V filter.

As outlined in the previous section, each set of flat frames for each filter were averaged and saved. They were then normalised by dividing the master flat by the average count from the frame's middle region.

4.1.2 NN Ser Calibration and Reduction

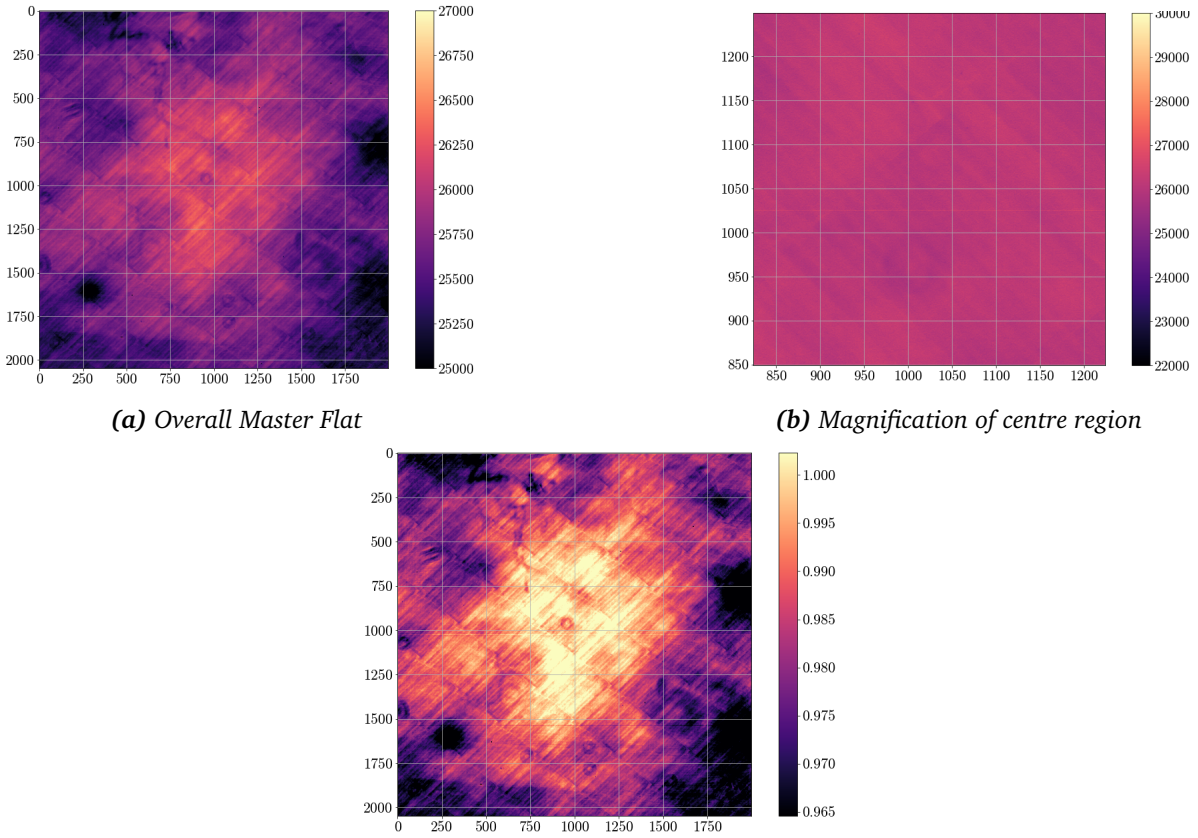
As outlined in the previous sections, the same calibration process needs to be completed for the NN Ser system. Unlike M_{91} the NN Ser images were taken with only a clear filter. Thus only one set of object images need to be reduced. This is done with the same code which can be found in listing 6 in appendix B.2.

4.2 Combining reduced images

Following the previous section, there are now two sets of reduced images for both M_{91} and NN Ser. In the case of M_{91} , each filter must be combined to make a master image. This is done by loading each of the reduced saved files for each filter and averaging them for the object images. The code used to complete this process can be found in listing 6 in appendix B.2. This process was also completed for NN Ser, expansion on this in section 5.4

4.3 Photometry & Light curve for NN Ser

Photometry can be performed on the reduced object images of NN Ser using photutils package. Using the DAO star finder portion of the package, it is possible to identify the object image sources. This package would create a background annulus and aperture for



(a) *Overall Master Flat*

(b) *Magnification of centre region*

(c) *Normalised master flat.*

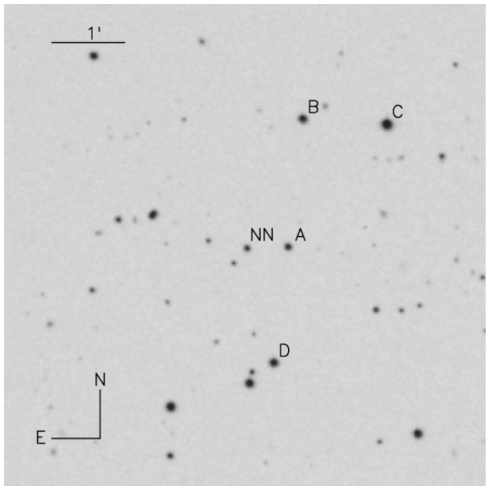
Figure 5 *M_{91} flats images for 'B' filter.*

each specific source. This allowed for the creation of subtracted background counts from each source to be determined. Thus a final table of background-subtracted counts within each aperture was created for each source.

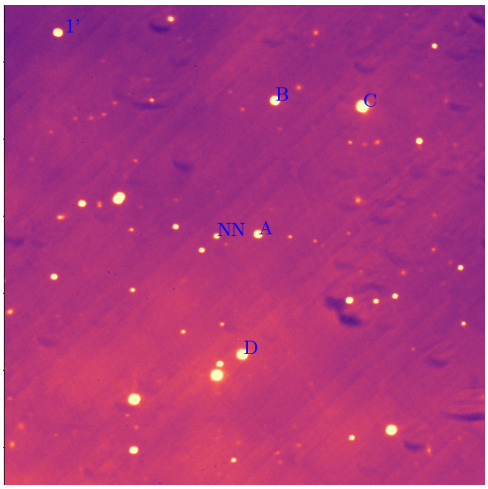
4.3.1 Application of the r' filter

As the clear filter does not have a catalogue of magnitudes calibrated to the sources in the object images application of an r' filter is performed so the magnitudes obtained for NN Ser can be compared against a standard system. This is completed by use of zero points as discussed in section 2.3.2.

Using the sky survey (fig. 6a) each comparison star was found and marked on the reduced master image (fig. 6b). Using the code in listing 22 in appendix B.4 the marked comparison stars were plotted onto each object image to ensure source indexing was correct.



(a) *Digital Sky Survey finding chart (POSS II, blue) for NN Ser. Comparison stars are marked (8)*



(b) *Comparison sources and NN Ser marked on combined master image.*

Figure 6 *Object images marked with comparison stars.*

Then as discussed in section 2.3.2 the zero points were calculated for each comparison star using table 1 and eq. (2). The zero-points average was then taken and added to the clear filter magnitude of NN Ser approximating the magnitude to an r' filter calibration. This was carried out using the code in listing 26 in appendix B.4.

The light curve is then plotted with the newly calibrated magnitudes of NN Ser against the universal time each object image was taken. (8)

Star	u'	g'	r'	i'
A	17	15.6	15.8	15
B	16	14.7	15.1	14.3
C	14.6	13.4	13.7	12.8
D	16.7	14.6	13.7	-

Table 1 *Comparison star magnitudes from NN Ser. (8)*

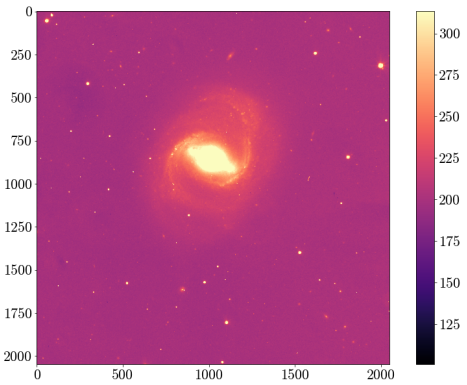
5 Results

5.1 Messier 91 (M_{91})

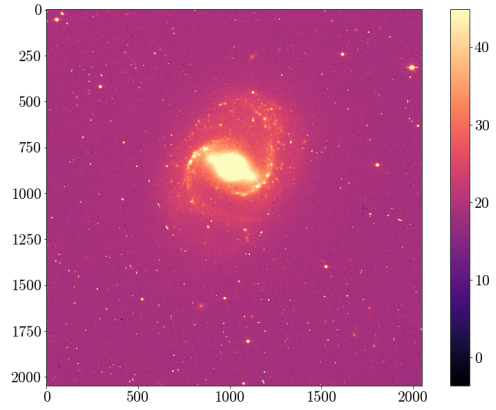
5.1.1 Combined and Reduced Images

As outlined in section 4.1.1 each scientific image must be reduced before analysis is performed. This is done as described by eq. (5), in the case of M_{91} as there were three filters used for the final combined image. Thus three flats were needed for each filter (see appendix A for all images).

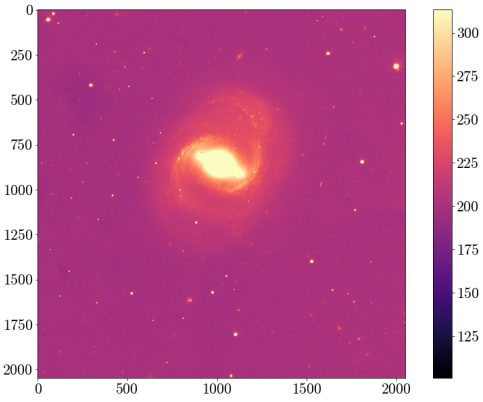
For each object image for each filter, the reduction using the bias and flat was applied, and each image was then saved in a reduced format. For a code example of this see listing 6 in appendix B.2. Each reduced object image was then averaged to produced three combined images (See fig. 7).



(a) *Combined object images for B filter*



(b) *Combined object images for Hα filter*



(c) *Combined object images for V filter.*

Figure 7 *M_{91} combined object images for each filter.*

Each reduced object image for each filter was then averaged to give a 'master' combined image used for analysis. The code example used for this can be viewed in listing 9 in appendix B.2. The master combined image can be viewed in fig. 8.

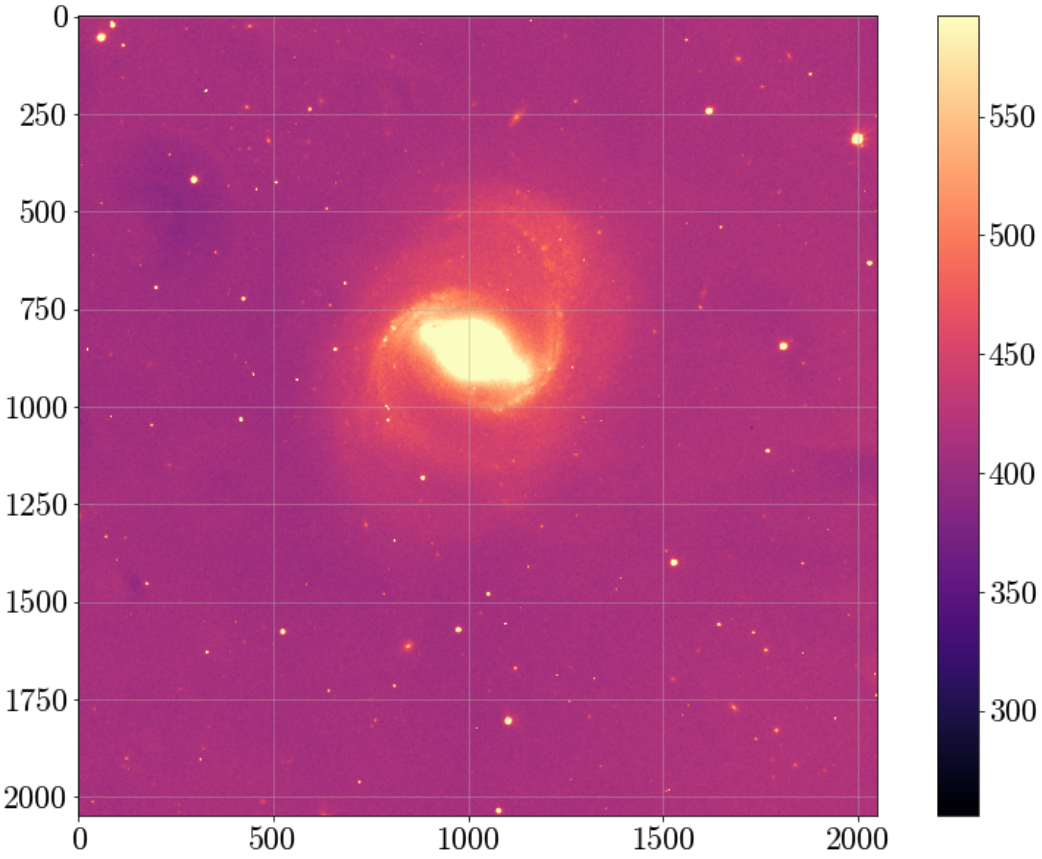


Figure 8 *Combined master image of* M_{91}

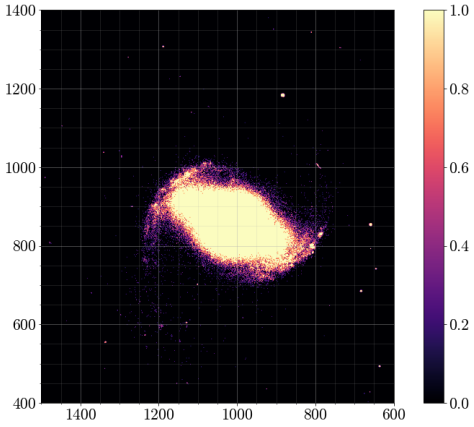
5.2 Determining the Angular Size of M_{91}

With the use of the obtained master image (See fig. 8) and some analysis of the image, it is possible to determine the angular size of M_{91} . While there are numerous ways of approaching this problem, in this analysis, an algorithm was used.
The algorithm first separated the bright sources in the master image by comparing each pixel against the mean counts of an area of the image which was sparsely populated. This area used for the analysis of M_{91} was taken as 200 x 200 region on the image's bottom right.

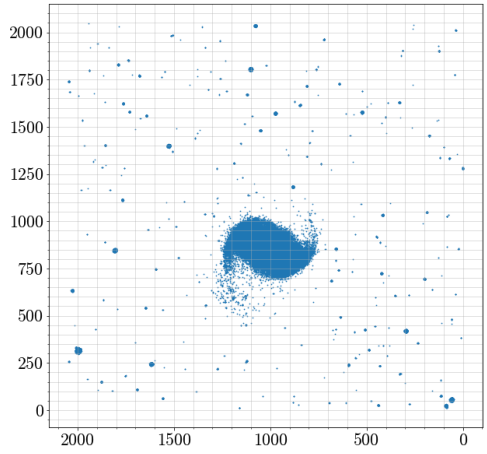
A baseline was calculated using the following expression,

$$\text{baseline} = \overline{\text{counts}} + n\sigma_c \tag{6}$$

Where n is an integer multiplier for the standard deviation of counts from the 200 x 200 region.
In the case of this analysis, five standard deviations ($n = 5$) was added to the base-line. This meant any source that wasn't cut from the data set had to meet a threshold of 5 standard deviations above the average counts of the 200 x 200 region. This produced a base-line of 480 counts.



(a) *M_{91} after pixel by pixel cutting*



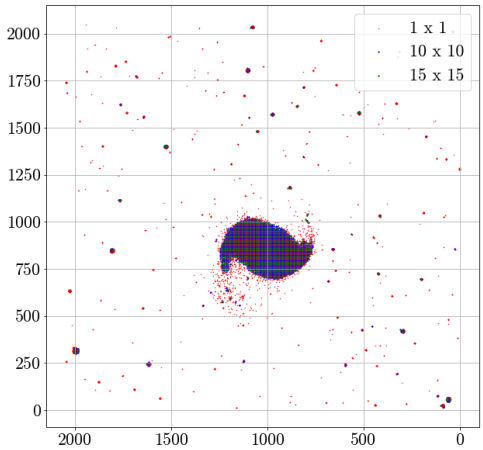
(b) *Full frame plotted on a clear background. After pixel by pixel cutting.*

Figure 9 *Results of comparing each pixel to a baseline.*

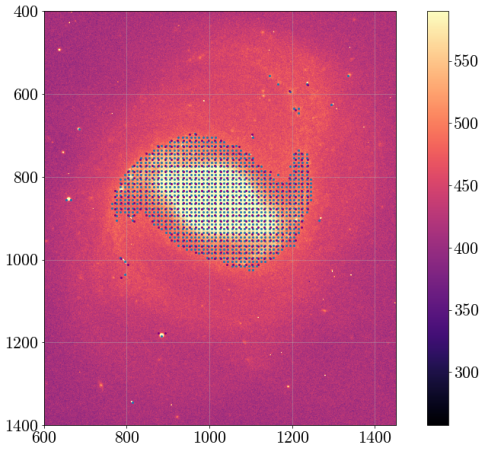
Now that the image had been further reduced to sources of interest, more isolation was needed before calculation of angular diameter. As seen in the above images (fig. 9), there are still sources that are not part of the galaxy present.

To account for this analysis on grids of pixels was used in a similar method to the pixel-by-pixel isolation. This entailed comparing varying grids of a n x n size to be compared against a base-line. This would allow the tightly packed cluster to remain in the data source while smaller sets of isolated sources to get cut (i.e. single sources). The grid-size was determined by overlaying varies grid sizes over each other and the master image to determine which provided the most efficient result (see fig. 10 below).

It was found that a grid size of 10 x 10 and 15 x 15 produced similar results from observation. However, 10 x 10 grid size produced more isolated sources than 15 x 15, which is expected as tighter clusters of sources can meet the threshold.



(a) *Comparison over-layed on master image.*



(b) *Comparison over-layed on master image.*

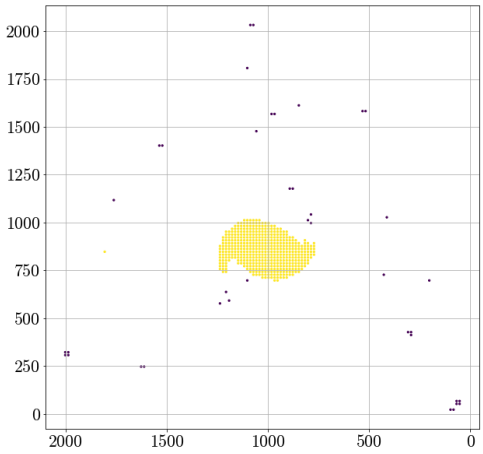
Figure 10 *Comparison n x n grid scans.*

The final step in isolation before analysis was done was getting rid of all unwanted sources. This step's goal was to produce a set of data that was a tight silhouette of the galaxy once overlayed on the master image.

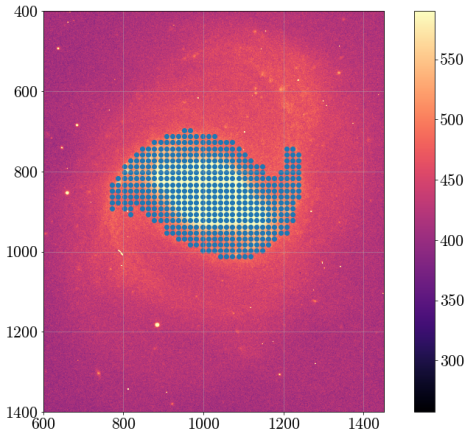
Removing outliers was completed using a Density-based spatial clustering of applications with noise (DBSCAN). This allowed for clusters to be removed based on population size and radius population clusters. The cut-off radius was taken to be the average distance between stars in an anaemic galaxy. While this cut-off number does not accurately represent M_{91} appropriately, it yielded desired results. The results of the DBSCAN can be seen below in fig. 11.
See listing 17 in appendix B.3 for an example of the code used to complete this process.

With isolated data-points, it is now possible to complete analysis and estimate the angular size. For this analysis, the boundaries of M_{91} are sources produced from the algorithm. However, this does not mean that the galaxy does not extend beyond these bounds. The limits of a barred spiral galaxy are challenging to determine, and there is no clear defined boundary.

Using the code in listing 18 in appendix B.3, the data points were rotated to the minimum, and maximum x value would create a line that would represent the semi-major axis and the same semi-minor axis in the y-direction. This produced an angular size of 4'.71 x 3'.29 arcmins before correction. Correcting the angular size for the inclination of 36.9 degrees changed each value approximately by a factor of 0.16, giving an angular size of 5'.47 x 3'.82 arcmins. See listing 18 in appendix B.3 for the code use to complete this calculation.



(a) *DBSCAN with each colour group coloured.*



(b) *DBSCAN results over layed on master image.*

Figure 11 *DBSCAN results*

While these values are within the expected range for M_{91} of 5'.4 x 4'.4 arcmins, the degree of uncertainty is relatively high. Performing this procedure 'by eye' will yield similar results; however, this method does have much more rigour. The main issue with this method is the bounds of which the galaxy is determined; this method removes sources on the galaxy's boundary, causing this method to produce underestimations of the galaxies size, in repeating the same process on a pixel-pixel analysis an angular size of 5'.7 x 4'3. However, pixel by pixel produces many outliers and grossly over-estimates the angular size without a DBSCAN which often yields inconsistent results as seen in fig. 12.

Finally we can convert the obtained apparent size into kpc with use of the following formula,

$$D = \frac{\text{angular-size}}{206625} d \tag{7}$$

Where, D is distance in parsecs and d is the distance to object. Which yields an apparent physical size of $20.4 \times 14.3\ kpc$ and $24.8 \times 19.1\ kpc$ for 15 x 15 grid and pixel by pixel analysis respectively. (3) (8)

When looking at the uncertainties for the apparent size of M_{91} , it's challenging to find one numerical value. The result obtained is dependant on many factors. Firstly the distance to M_{91} is given as 15 ± 0.8 Mpc; this already adds uncertainty to the value that would be substantially higher than any other uncertainty even before ever analyzing the image. There

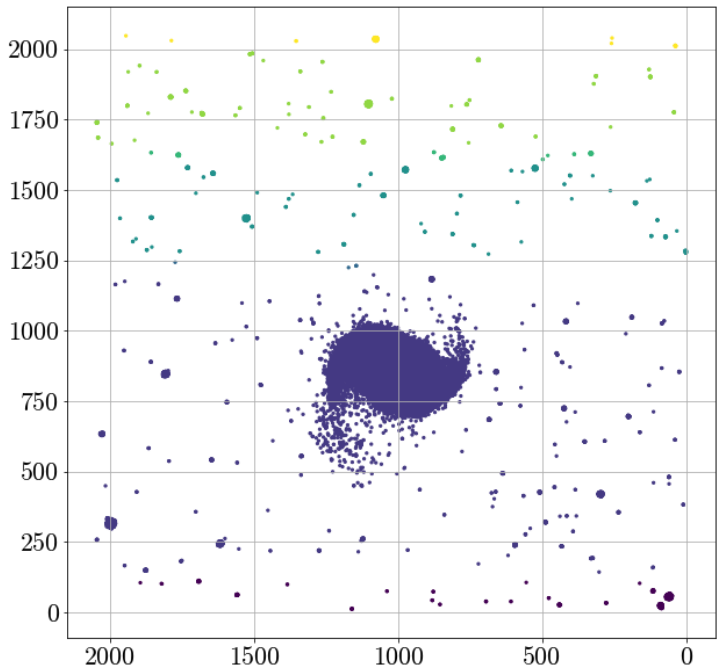


Figure 12 *DBSCAN of pixel by pixel scan.*

is also the issue of determining the galaxy's bounds begin and end as the DBSCAN doesn't account for how the source population density varies on the outer regions of the galaxy. Also, since the galaxy isn't uniformly elliptical, there are assumptions in correction and calculation of apparent size.

The uncertainty on distance is the most significant uncertainty present in the calculation of apparent size; it will give a reasonable numerical uncertainty. The apparent physical size of M_{91} was calculated with a distance of 14.2 Mpc and 15.8 Mpc with the variation used as an uncertainty. While not the most rigorous method for calculating uncertainty, it will be consistent with the calculated results.

So, in this case, the pixel by pixel method while less rigorous gives a more accurate depiction of apparent physical size. While using the grid system would work better and more consistently if this calculation was needed with more galaxies, it falls short when dealing with a spiral barred galaxy's shape and boundaries. In the case of the pixel by pixel approach, it over-estimates the semi-major axis and underestimates the semi-minor axis. Once again this is due to the defining the galaxy's bound and the variation from a conventional elliptical shape. (6) (4)

5.3 NN Serpentis

5.3.1 Combined and Reduced Images

As with M_{91} , each of the object images must be reduced before analysis can take place. This is done following the method described by eq. (5). In the case of NN Ser object images, the only filter used is a 'clear' filter. For each object image reduction was applied (see listing 6 in appendix B.2) and the resulting image saved. Each newly save image was then combined to create a 'master' image (See fig. 13 below.)

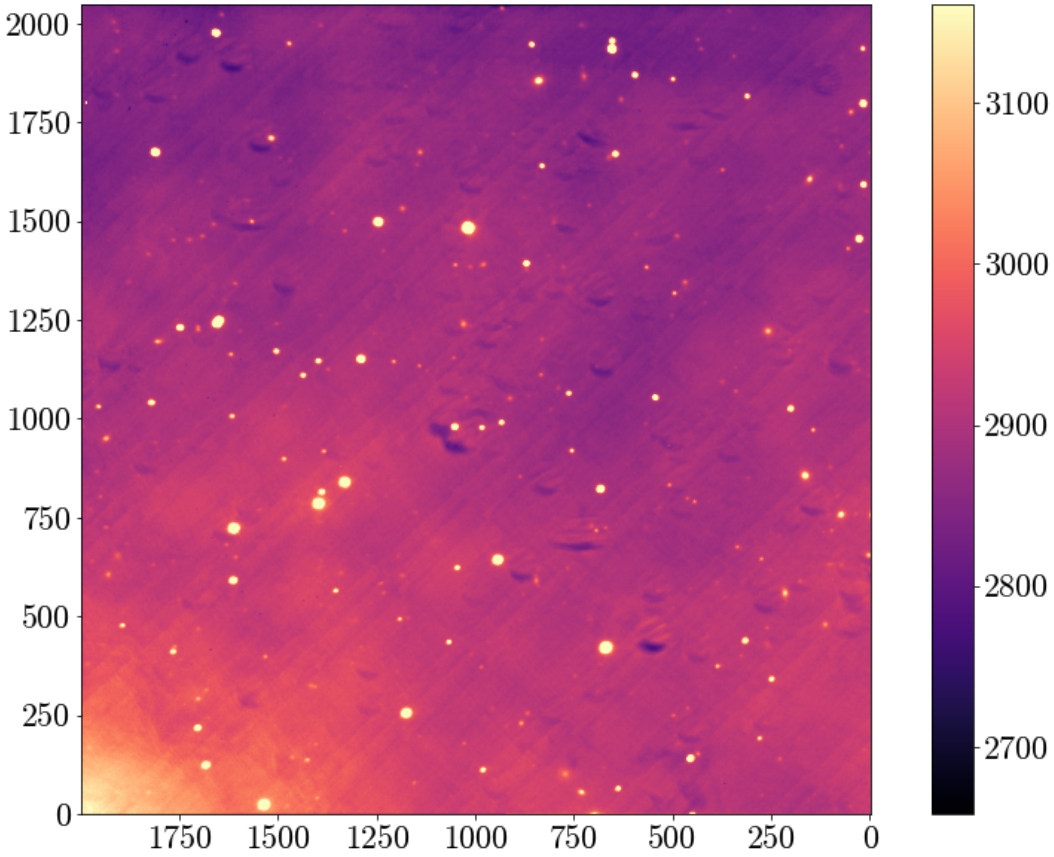


Figure 13 *Combined master image of NN Ser*

Where the calibration (bias and flat) images can be viewed in appendix A.

5.4 Light Curve produced from NN Serpentis

As discussed in section 2.3.2 and section 4.3 the light curves were plotted with calibrated r' filter magnitude against time as seen below in fig. 14 and fig. 16.

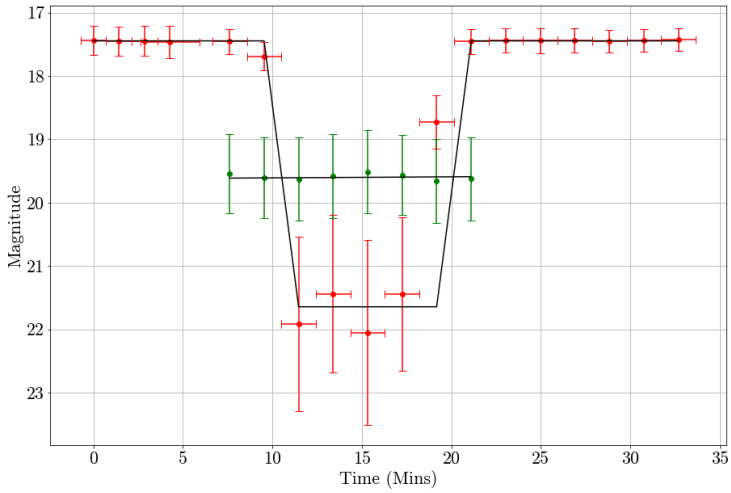


Figure 14 *Light curve of NN Ser with errors.*

Before calculating the eclipse duration, the eclipse starts and ends as the magnitude gradually decreases and increases again. To determine this a limiting magnitude as used. Once the combined magnitude of NN Ser dips below the limiting magnitude, it is no longer visible, and the eclipse has started. When the combined magnitude returns to the level of the limiting magnitude the eclipse has ended.
The faintest source was found by finding the source with the lowest flux from the photometry table (This source is circled in green in fig. 15). The magnitudes were then calibrated for the r' filter and plotted along with the light curve of NN Ser.

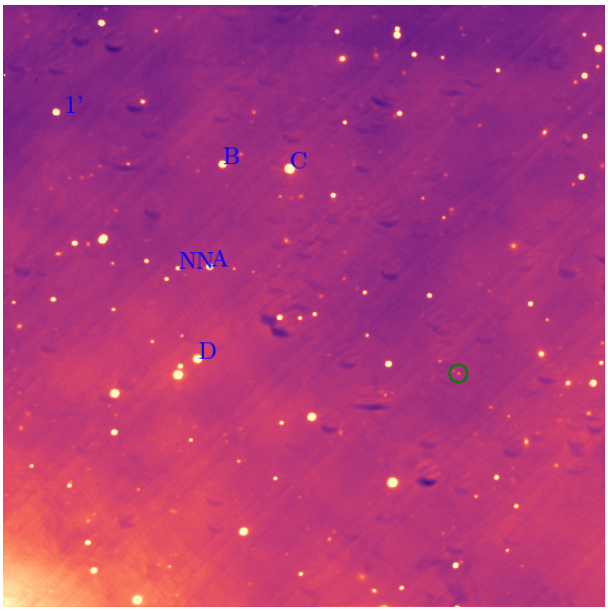


Figure 15 *Master object image marked with source used for limiting magnitude (green).*

The magnitudes of the limiting source were then plotted using a linear fit. The reason for a linear fit here is we assume that the limiting source stays at a constant brightness for the duration of the eclipse.

The magnitudes for NN Ser were fitted using a top-hat function. The base-line was determined to be the average between the points before and after the dip in combined magnitude. The depth, width and midpoint were determined by using the points where there is variation in combined magnitude.

The eclipse start and endpoints were determined to be the two points of intersection of the two functions (See. Fig 16). The eclipse's start occurred at 10.5 ± 1.5 minutes, and the end of the eclipse occurred at 22.1 ± 1.5 minutes. Thus, the duration of the eclipse was found to be 11.6 ± 2.2 minutes. The duration length is appropriate as the eclipse length for NN Ser is roughly 12 minutes.

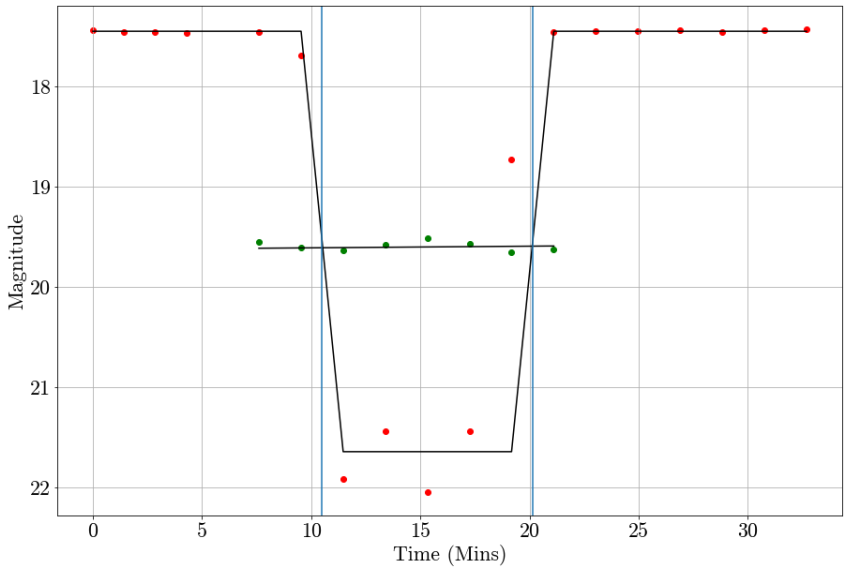


Figure 16 *Light-curve of NN Ser with intersection points of the functions marked.*

When calculating uncertainties for the light curves time and magnitudes are considered separately.
For the magnitudes, a signal noise was calculated using a generalised variation of the CCD equation,

$$S/N = \frac{F_*}{\sqrt{F_* + \left(1 + \frac{n_*}{n_{\text{sky}}}\right)\left(F_{\text{sky}} + R^2\right)}} \tag{8}$$

Where, F_* are the counts that measured from the source, F_* is the number of pixels within your background annulus F_{sky} are the number of pixels within photometric aperture and R is the readout noise for the CCD in this case is $3.37e^{-1}$. The uncertainty on the calibrated NN Ser magnitudes was then calculated using the following equation,

$$\sigma \;=\; 2.5\log_{10}\left(1+\frac{1}{S/N}\right) \tag{9}$$

For a total uncertainty on the magnitudes to be calculated an uncertainty on the zero-points also needs to be calculated and added. The error on the zero-points was calculated using a mean error formula.

$$\sigma = \frac{\sqrt{ZP_{\text{std}}}}{\sqrt{n}} \tag{10}$$

Where ZP_{std} is the standard deviation of zero point's in a particular object image and n is the number of zero points in the image. All calculations were completed and added using the code which can be in listing 26 in appendix B.4

When it comes to time, the calculation of uncertainty isn't due to issues with inaccuracies in measurement, it's an issue of the size of the increments between the time images are taken. As seen in fig. 14, there is a lack in data points between the dip in combined magnitude and the rise (Comparing this with the light curve produced by European Southern Observatory, fig. 2). This makes it harder to determine how to fit the top-hat function best and ultimately, where the eclipse starts and stops.

It's known that magnitude is changing between each data point; thus if another image is taken between each image, it would fall between the data points. So in the light-curve case, the uncertainty on time was taken to be half the difference in time between each image taken. Doing this gives a visual representation of where other data points would lie images were taken in shorter increments, and more data points were available.

During the eclipse the white dwarf in the NN Ser system is eclipsed which is illustrated[3] in fig. 17. Every 3 hours the white dwarf (smaller source) moves around the brown dwarf (Larger Source) causing an eclipse that lasts approximately 11.6 minutes when observing from Earth. From the information the light curve provides, it would be possible to determine each star's diameters and if more spectral information were available things such as relative

[3]Orbits aren't circular, they're elliptical but for this illustration, the orbits are sketched circular.

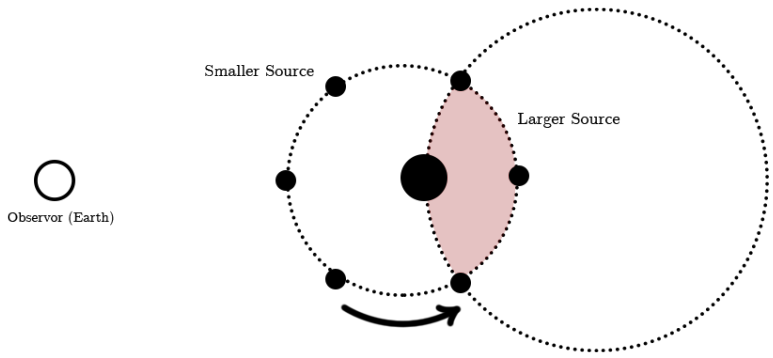


Figure 17 *Sketch of eclipse of NN Ser.*

velocity would be deduced from doppler shift.

6 Conclusion

Throughout this investigation, successful reduction and calibration were applied to scientific images for both the Messier 91 galaxy and NN Serpentis binary system taken by the IAC 80 telescope. Messier 91 apparent physical size was determined by using an algorithm that isolated the galaxies sources based on the mean count. The apparent physical size of Messier was found to be $(24.8 \times 19.1) \pm 0.5\ kpc$. This value falls within the expected range of the size outlined in the Messier catalogue.

Aperture photometry was successfully used to plot a light curve of the NN Ser system and determine the length of the eclipse to be 11.6 ± 2.2 minutes. This was within the expected range as the rough value for the eclipse duration is accepted as roughly 12 minutes. The uncertainties on the r' calibrated magnitudes were as expected with the higher count values having a smaller signal to noise ratio and the lower counts having higher.

References

[1] URL: https://www.physics.rutgers.edu/analyze/triangle_2.html.

[2] URL: http://research.iac.es/OOCC/iac-managed-telescopes/iac80/.

[3] Using angular size relationship to find the diameter. URL: http://teacherlink.ed.usu.edu/tlnasa/reference/imaginedvd/files/imagine/YBA/HTCas-size/optical.html.

[4] *Data clustering: algorithms and applications*. Chapman Hall/CRC data mining and knowledge discovery series. Chapman and Hall/CRC, 2014.

[5] Christoph Baranec, Reed Riddle, Nicholas M. Law, A.N. Ramaprakash, Shriharsh P. Tendulkar, Khanh Bui, Mahesh P. Burse, Pravin Chordia, Hillol K. Das, Jack T.C. Davis, and et al. Bringing the visible universe into focus with robo-ao. *Journal of Visualized Experiments*, (72):50021, Feb 2013. https://doi.org/10.3791/50021 doi:10.3791/50021.

[6] Derya Birant and Alp Kut. St-dbscan: An algorithm for clustering spatial–temporal data. *Data Knowledge Engineering*, 60(1):208–221, Jan 2007. https://doi.org/10.1016/j.datak.2006.01.013 doi:10.1016/j.datak.2006.01.013.

[7] W.-C. Chen. Can angular momentum loss cause the period change of nn serpentis? *Astronomy Astrophysics*, 499(1):L1–L3, May 2009. https://doi.org/10.1051/0004-6361/200911638 doi:10.1051/0004-6361/200911638.

[8] Morgan Fraser. *Astronomical Image Analysis*. UCD Department of Physics, 2 edition, 2020.

[9] Rob Garner. Messier 91, Oct 2017. URL: http://www.nasa.gov/feature/goddard/2017/messier-91.

[10] Skye Hardesty. Seds messier database (messier catalog)2005428 seds messier database (messier catalog) . cambridge, ma: Students for the exploration and development of space (seds) last visited june 2005. gratis url: www.seds.org/messier/. *Reference Reviews*, 19(8):45–45, Dec 2005. https://doi.org/10.1108/09504120510632796 doi:10.1108/09504120510632796.

[11] Steve B. Howell. *Handbook of CCD astronomy*. Cambridge observing handbooks for research astronomers. Cambridge University Press, 2nd ed edition, 2006.

[12] information@eso.org. Nn ser light curve around total eclipse. URL: https://www.eso.org/public/images/eso9936c/.

[13] Francis LeBlanc. *An introduction to stellar astrophysics*. Wiley, 2013. URL: http://rbdigital.oneclickdigital.com.

[14] Don Machholz. *The Observing Guide to the Messier Marathon: a Handbook and Atlas*. 2002. URL: https://doi.org/10.1017/CBO9780511536458.

[15] Patrick Martinez and Alain Klotz. *A practical guide to CCD astronomy*. Practical astronomy handbook series. Cambridge University Press, 1998.

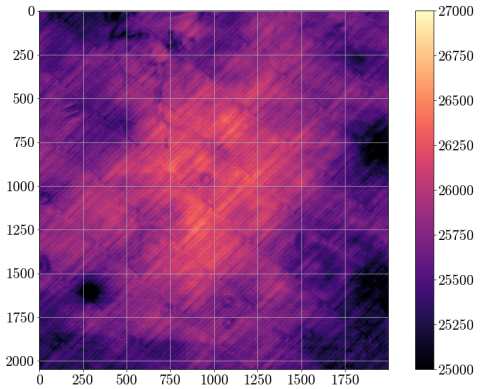
[16] Eugene F Milone. *Astronomical photometry: past. present, and future*. Springer, 2013.

[17] S.-B. Qian, Z.-B. Dai, W.-P. Liao, L.-Y. Zhu, L. Liu, and E. G. Zhao. A substellar companion to the white dwarf-red dwarf eclipsing binary nn ser. *The Astrophysical Journal*, 706(1):L96–L99, Nov 2009. https://doi.org/10.1088/0004-637X/706/1/L96 doi:10.1088/0004-637X/706/1/L96.

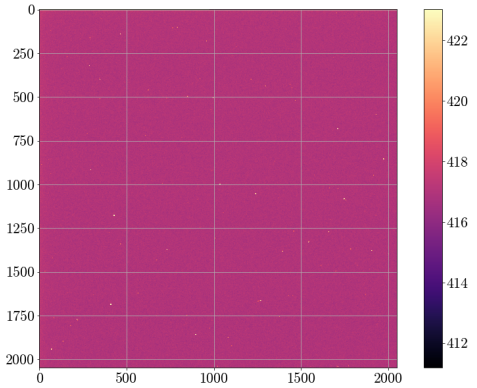
[18] Chris Woodhouse. *The Astrophotography Manual: A Practical and Scientific Approach to Deep Sky Imaging*. Focal Press, second edition, 2017.

Appendices

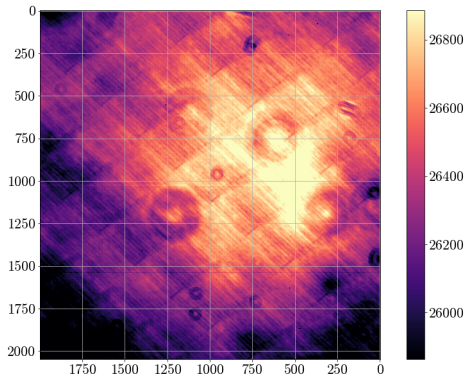
A Calibration Images



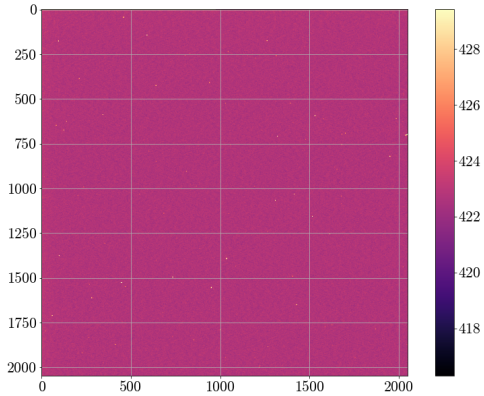
(a) *B Filter Master Flat*



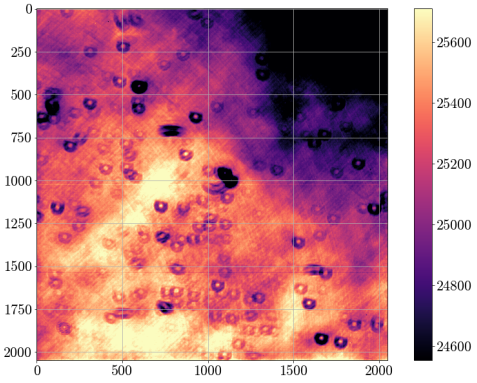
(b) *M_{91} Master Bias*



(c) *V Filter Master Flat*



(d) *NN Ser Master Bias*



(e) *NN Ser Master Flat*

Figure 18 *Calibration Images.*

B Experimental Code

B.1 Packages Used

```
import numpy as np
import pandas as pd
import os
import sys
import scipy
from scipy import stats
from astropy.io import fits #allows python to interpurt fits data files
from glob import glob # Unix style pathname pattern expansion
from matplotlib import pyplot as plt
from astropy.wcs import WCS
from photutils import DAOStarFinder
from astropy.stats import mad_std
from photutils import aperture_photometry, CircularAperture,
    CircularAnnulus
from sklearn.cluster import DBSCAN
from numpy import genfromtxt
%matplotlib inline

# ---- plot parameters ----
plt.rcParams["figure.figsize"] = (15,10)
plt.rc('font', family = 'serif', serif = 'cmr10')
plt.rcParams.update({'font.size': 22})
```

Listing 1 *M_{91} Package Load*

```
import numpy as np
import pandas as pd
# import os
import scipy
from scipy . optimize import curve_fit
from scipy . interpolate import interp1d
from astropy.io import fits #allows python to interpurt fits data files
from glob import glob # Unix style pathname pattern expansion
from matplotlib import pyplot as plt
from astropy.wcs import WCS
from astropy.stats import sigma_clip
from photutils import DAOStarFinder
from astropy.stats import mad_std
```

```
    from photutils import aperture_photometry, CircularAperture,
        CircularAnnulus
    %matplotlib inline

    # ---- plot parameters ----
    plt.rcParams["figure.figsize"] = (15,10)
    plt.rc('font', family = 'serif', serif = 'cmr10')
    plt.rcParams.update({'font.size': 22})
```

Listing 2 *NNSer Package Load*

```
    import numpy as np
    from scipy import stats
    from scipy . optimize import curve_fit
    from scipy . interpolate import interp1d
    from scipy import optimize
    from astropy.io import fits #allows python to interpurt fits data files
    from glob import glob # Unix style pathname pattern expansion
    import uncertainties
    from astropy.stats import sigma_clip
    from uncertainties import ufloat_fromstr
    from uncertainties.umath import
    from uncertainties import unumpy as unp
    from exoctk.lightcurve_fitting.lightcurve import LightCurve
    from exoctk.lightcurve_fitting.parameters import Parameters
    from exoctk.lightcurve_fitting.models import PolynomialModel, TransitModel
    import matplotlib.mlab as mlab
    import matplotlib.pyplot as plt

    # ---- plot parameters ----
    plt.rcParams["figure.figsize"] = (15,10)
    plt.rc('font', family = 'serif', serif = 'cmr10')
    plt.rcParams.update({'font.size': 22})
```

Listing 3 *Error Package Load*

B.2 Image Calibration and Reduction

```
# ---- Data Import of Lab Data ----

NNSer_bias_data = [fits.getdata(image) for image in glob('lab_data/Ser/bias/
    B120308 ')]
```

```
4  print(np.shape(NNSer_bias_data))
5
6  NNSer_flat_clear_data = [fits.getdata(image) for image in glob('lab_data/Ser/
       flats/clear/F120308 ')]
7  print(np.shape(NNSer_flat_clear_data))
8
9  NNSer_object_clear_data = [fits.getdata(image) for image in glob('lab_data/Ser
       /object/clear/O120308 ')]
10 print(np.shape(NNSer_object_clear_data))
11
12 # ---- Clipping Overscan ----
13
14
15 master_bias_NNSer = np.average(NNSer_bias_data, axis = 0)
16 master_bias_NNSer = np.delete(master_bias_NNSer, np.s_[0:50], axis = 1)
17 print(np.shape(master_bias_NNSer))
18 print(master_bias_NNSer.mean())
19
20 master_flat_NNSer = np.average(NNSer_flat_clear_data, axis = 0)
21 master_flat_NNSer = np.delete(master_flat_NNSer, np.s_[0:50], axis = 1)
22 print(np.shape(master_flat_NNSer))
23
24 master_flat_NNSer_Norm =  master_flat_NNSer/master_flat_NNSer[a:b, c:d].mean()
25 print(np.shape(master_flat_NNSer_Norm))
26 print(np.mean(master_flat_NNSer_Norm))
27
28 # ---- Saving Master Files ----
29
30 fits.PrimaryHDU(master_bias_NNSer).writeto('Ser_Redlab_data\Ser_Master_Bias',
       overwrite = True)
31 fits.PrimaryHDU(master_flat_NNSer).writeto('Ser_Redlab_data\Ser_Master_Flat',
       overwrite = True)
32 fits.PrimaryHDU(master_flat_NNSer_Norm).writeto('Ser_Redlab_data\
       Ser_Master_Flat_Normalised', overwrite = True)
```

Listing 4 *NNSer: Intial data load and saving of calibrated files*

```
1  # ---- Plots of Master Flat and Bias ----
2
3  # ---- Plotting Master Bias ----
4  bmax = master_bias_NNSer.mean() + master_bias_NNSer.std()
5  bmin = master_bias_NNSer.mean() - master_bias_NNSer.std()
6
```

```
7  plt.imshow(master_bias_NNSer, cmap = 'magma', vmin = bmin , vmax = bmax)
8  # plt.title("Master Bias for NNSer")
9  plt.grid()
10 plt.colorbar()
11 plt.show()
12 plt.savefig("images/Ser/NNSer_Master_Bias.svg")
13
14 # ---- Plotting Master Flat ----
15 bmax = master_flat_NNSer.mean() + master_flat_NNSer.std()
16 bmin = master_flat_NNSer.mean() - master_flat_NNSer.std()
17
18 plt.imshow(master_flat_NNSer, cmap = 'magma', vmin = bmin , vmax = bmax)
19 # plt.title("Master Flat for NNSer")
20 plt.grid()
21 plt.colorbar()
22 plt.show()
23 plt.savefig("images/Ser/NNSer_Master_Flat.svg")
24
25 # ---- Plotting Master Flat Normalised ----
26 bmax = master_flat_NNSer_Norm.mean() + master_flat_NNSer_Norm.std()
27 bmin = master_flat_NNSer_Norm.mean() - master_flat_NNSer_Norm.std()
28
29 plt.imshow(master_flat_NNSer_Norm, cmap = 'magma', vmin = vmin , vmax = vmax)
30 # plt.title("Normalised Master Flat for NNSer")
31 plt.grid()
32 plt.colorbar()
33 plt.show()
34 plt.savefig("images/Ser/NNSer_Master_Flat_Norm.svg")
```

Listing 5 *NNSer: Plotting Calibration images for observation and saving.*

```
1      # ---- Reducing Object Images and Producing a master file ----
2
3      object_B = glob('lab_data\Ser\object\clear\O120308 ')
4      # print(object_B)
5      object_B_pixels = NNSer_object_clear_data
6
7      mb_clipped = fits.getdata('Ser_Redlab_data\Ser_Master_Bias')
8
9      for file in object_B:
10         object_BB = fits.open(file) # stores fit file
11         object_B_pixels = object_BB[0].data # access of pixel data
12         object_B_subtracted = np.delete(object_B_pixels, np.s_[0:50], axis=1)
```

```
            - mb_clipped #
        object_B_reduced =  object_B_subtracted/master_flat_NNSer_Norm #
        object_B_pixels = object_B_reduced
        HDU = fits.PrimaryHDU(object_B_reduced)
        name = 'Ser_Red' + file.split('/')[-1] #"Ser\object_reduced\
            object_reduced_" +
        # print(newfilename)
        HDU.writeto(name, overwrite = True)
```

Listing 6 *NNSer: Reducing and Creating files for each object frame.*

```
reduced_object = glob('Ser_Redlab_data/Ser/object/clear/O120308 ')
reduced_object_data = [fits.getdata(image) for image in glob('Ser_Redlab_data/
    Ser/object/clear/O120308 ')]
print(np.shape(reduced_object_data))

# ---- Combine ----
combined_reduced_object_data = np.average(reduced_object_data, axis = 0)

# ---- Saving the Combined Image ----
fits.PrimaryHDU(combined_reduced_object_data).writeto('master_images\
    master_image_NNSer', overwrite = True)
```

Listing 7 *Loading reduced image files*

```
    # ---- Looking at the reduced image output before proceeding further

    test_print = glob('Ser_Redlab_data/Ser/object/clear/O120308 ')
    sources_array = []

    for i in range(0, len(test_print) - 1):

        # ---- Data Import ----
        data = reduced_object_data[i]
        print(np.shape(data))
        print(data.mean())
        std = (np.std(data))
        vmin = data.mean() - data.std()
        vmax = data.mean() + data.std()

        # ---- Sources ----
        std = mad_std(data)
        daofind = DAOStarFinder(fwhm = 6., threshold = 5. std)
```

```
        mst_sources = daofind(data)
        print(len(mst_sources['id']))
        sources_array = np.append(sources_array, len(mst_sources['id']))

        #--- Plot ---
        plt.imshow(np.delete(data, np.s_[0:50], axis=1), cmap = 'magma', vmin
            = vmin, vmax = vmax)
        plt.colorbar()
        plt.gca().invert_yaxis()
        plt.show()
```

Listing 8 *NNSer: Printing each image for observation to ensure correct reduction.*

```
filterB = [fits.getdata(image) for image in glob('M91/object_B_reduced_B/
    O130311 ')]
filterHA = [fits.getdata(image) for image in glob('M91/
    object_HA_reduced_Halpha_NII/O130311 ')]
filterV = [fits.getdata(image) for image in glob('M91/object_V_reduced_V/
    O130311 ')]

print(np.shape(filterB), np.shape(filterHA), np.shape(filterV))

combined_array = np.concatenate((filterB, filterHA, filterV), axis = 0)
print(np.shape(combined_array))

master_image = np.average(combined_array, axis = 0)


newfilename = "master_images/master_image_M91"
print(newfilename)
newheader.writeto(newfilename, overwrite = True)
```

Listing 9 *M_{91} : Combining for master image.*

B.3 M_{91} Angular Size Algorithm Code

```
M91_master_image = fits.open('master_images/master_image_M91')
M91_master_data = fits.getdata('master_images/master_image_M91')
M91_master_data = np.delete(M91_master_data, np.s_[0:50], axis = 1)
M91_wcs = WCS(M91_master_image[0].header)

# print(M91_wcs)
```

```
7     print(M91_master_data.mean())
8     print(M91_master_data.std())
9
10    # ---- Projection Plot ----
11
12    vmin = M91_master_data.mean() - M91_master_data.std()
13    vmax = M91_master_data.mean() + M91_master_data.std()
14
15    # plt.subplot(projection = M91_wcs)
16    plt.imshow(M91_master_image[0].data, cmap = 'magma', vmin = vmin, vmax =
          vmax)
17    plt.colorbar()
18    plt.grid(True, which = 'major', alpha = .5)
19    # plt.minorticks_on()
20    # plt.grid(True, which = 'minor', alpha = .2)
21    plt.xlim(600, 1450)
22    plt.ylim(400, 1400)
23    plt.gca().invert_xaxis()
24    plt.savefig('images/M91/M91_plain-master-zoom.png')
25    plt.show()
26
27    plt.imshow(M91_master_image[0].data, cmap = 'magma', vmin = vmin, vmax =
          vmax)
28    plt.colorbar()
29    plt.grid(True, which = 'major', alpha = .5)
30    # plt.minorticks_on()
31    # plt.grid(True, which = 'minor', alpha = .2)
32    # plt.gca().invert_xaxis()
33    # plt.show()
34    plt.savefig('images/M91/M91_plain-master.png')
```

Listing 10 *Plotting master image and area of intrest.*

```
1     baseline = np.mean(M91_master_image[0].data[1800:2000, 1800:2000] + 5(
          M91_master_image[0].data[1800:2000, 1800:2000].std())) #creating a
          standard to compare images against.
2
3     M91_master_data = fits.getdata('master_images/master_image_M91')
4
5     bright_sources = np.where(M91_master_data > baseline)
6     x_pos = bright_sources[0]
7     y_pos = bright_sources[1]
8
```

```
9     combined_positions = np.vstack((x_pos, y_pos)).T # transposing so axis
          match images.
10    print(np.shape(combined_positions))
11
12    empty_image = np.zeros((np.shape(M91_master_image[0].data)[1], np.shape(
          M91_master_image[0].data)[1])) # empty array the size of the image so
          position data can be overlayed.
13    print(np.shape(empty_image))
14
15    for i in combined_positions:
16        empty_image[i[0], i[1]] = 1 # loop to add positions to 'blank' image.
17
18    bright_isolated_image = empty_image
19
20    # ---- Plotting ----
21
22    plt.grid(True, which='major', alpha=0.5)
23    plt.minorticks_on()
24
25    plt.grid(True, which='minor', alpha=0.2)
26
27    plt.imshow(bright_isolated_image, cmap = 'magma', vmin=0, vmax=1)
28    plt.xlim(600, 1500)
29    plt.ylim(400, 1400)
30    plt.colorbar()
31    plt.gca().invert_xaxis()
32    plt.savefig('images/M91/M91_1pixel-heatmap.png')
33
34
35    # ---- Plotting onto White Backghround ----
36
37    plt.figure(figsize = (10, 10))
38    plt.grid(True, which='major', alpha = 0.5)
39    plt.minorticks_on()
40    plt.gca().invert_xaxis()
41    # plt.xlim(600, 1500)
42    # plt.ylim(350, 1400)
43    plt.grid(True,  which = 'minor', alpha=0.5)
44    plt.scatter(y_pos, x_pos, s = .5)
45    plt.savefig('images/M91/M91_1pixel-master.png')
```

Listing 11 *Creating a set of data points above certain baseline count (480)*

```
    def brightness_function(pixel_increment, image_data, baseline):
    # ---- constant definitions and variable set up: ----
    y_start = 0
    y_limit = 0
    counter = 0

    x_array = []
    y_array = []

    while y_limit < np.shape(image_data)[1]:
        for i in range(0, np.shape(image_data)[1], 1):
            counts = mst_image_data[i pixel_increment:i pixel_increment +
                pixel_increment, y_start:y_start + pixel_increment]
            counter += 1
            if counts.mean() > baseline:
                x_pos_c = (i pixel_increment + (i pixel_increment +
                    pixel_increment))/2 #center of box for easy viewing +
                    calculations.
                x_array = np.append(x_array, x_pos_c)

                y_pos_c = (y_start + (y_start + pixel_increment))/2 # doing
                    the same for the y-pos
                # print(y_pos_c)
                y_array = np.append(y_array, y_pos_c)

            # y_limit += pixel_increment

            if counter > np.shape(image_data)[1] + 1:
                y_start += pixel_increment
                y_limit = 0
                # print(y_start)
                counter = 0
                if y_start > 2000:
                    y_limit = np.shape(mst_image_data)[1]
                    return(x_array, y_array)
    print("The number of", pixel_increment, 'x', pixel_increment, "that
        exceeded the baseline value of", baseline, "counts, was found to be",
        len(x_array))
```

Listing 12 *Creating function that will scan specfied n x n grids.*

```
    plt.figure(figsize = (10, 10))
```

```
    plt.scatter(y_pos, x_pos, color = 'red', s = .2, label = '1 x 1')
    plt.scatter(t_10[1], t_10[0], color = 'blue', s = 1.5,  label = '10 x 10')
    plt.scatter(t_20[1], t_20[0], color = 'green', s = 2, label = '15 x 15')
    plt.grid()
    plt.legend()
    plt.gca().invert_xaxis()
    plt.savefig('images/M91/M91_grid_compare.png')
```

Listing 13 *Plotting comparison overlay.*

```
    vmin = M91_master_data.mean() - M91_master_data.std()
    vmax = M91_master_data.mean() + M91_master_data.std()

    # plt.plot(y_pos, x_pos, '.')

    plt.scatter(t_10[1], t_10[0], s = 4)
    plt.scatter(t_20[1], t_20[0], s = 4, color = 'purple')
    plt.imshow(M91_master_image[0].data, cmap = 'magma', vmin = vmin, vmax =
        vmax)


    plt.xlim(600, 1450)
    plt.ylim(400, 1400)

    # plt.imshow(bright_isolated_image, cmap = 'magma', vmin=0, vmax=1)
    plt.colorbar()
    plt.grid(True, which = 'major', alpha = .5)
    plt.gca().invert_yaxis()

    plt.savefig('images/M91/M91_overlay_compare.png')
```

Listing 14 *Plotting comparison overlay over master image.*

```
    feature_space = pd.read_csv("saved_data/15_pixels.csv", header = None, sep
        =",")
    # print(feature_space.tail)
    data = feature_space.iloc[:, 0:2].values
    # print(data)

    model = DBSCAN(eps = w, min_samples  =  e).fit(data)

    csv = genfromtxt("saved_data/15_pixels.csv", delimiter = ",")

```

```
    # Slicing array
    # X = csv[:, 0]
    # Y = csv[:, 1]

    X = x_array
    Y = y_array

    # Y = y_pos
    # X = x_pos

    print(X.max())

    var = model.labels_
    # print(var)

    plt.figure(figsize=(10, 10))
    plt.scatter(Y, X, c = var, marker='o', s = 5)
    plt.grid()
    plt.gca().invert_xaxis()

    plt.savefig('images/M91/M91_DBSCAN_15.png')
```

Listing 15 *DBSCAN Plot*

```

    obo = np.array([x_pos, y_pos])
obo = np.column_stack((obo))
df = pd.DataFrame(obo)

df.to_csv('saved_data/obo.csv')
print(df)
```

Listing 16 *writing to .csv*

```
    Xn = []
    Yn = []

    for i in range(0, len(var)):
        if var[i] == 0:
            Xn = np.append(Xn, X[i])
            Yn = np.append(Yn, Y[i])

    pos = [Xn, Yn]
```

```
10
11      # print(np.count_nonzero(var == 1))
12      # print(len(Xn), len(Yn))
13
14      print(Yn.max(), Xn.max())
15      plt.scatter(Yn, Xn)
16
17      vmin = M91_master_data.mean() - M91_master_data.std()
18      vmax = M91_master_data.mean() + M91_master_data.std()
19
20      plt.imshow(M91_master_image[0].data, cmap = 'magma', vmin = vmin, vmax =
            vmax)
21
22      plt.xlim(600, 1450)
23      plt.ylim(400, 1400)
24
25      # plt.imshow(bright_isolated_image, cmap = 'magma', vmin=0, vmax=1)
26      plt.colorbar()
27      plt.grid(True, which = 'major', alpha = .5)
28      plt.gca().invert_yaxis()
29
30      plt.savefig('images/M91/M91_overlay_DBSCAN.png')
```

Listing 17 *Isolating cluster of intrest from DBSCAN*

```
1       # ---- Rotating Data Points ----
2
3       print(np.shape(pos))
4
5       radians = 2.59
6       new_pos = []
7
8       tt = []
9
10      for i in range(0, len(pos[1])):
11          x = pos[0][i]
12          y = pos[1][i]
13          c, s = np.cos(radians), np.sin(radians)
14          j = np.matrix([[c, s], [-s, c]])
15          m = np.dot(j, [x, y])
16
17          NP = float(m.T[0]), float(m.T[1])
18          NP = np.asarray(NP)
```

```
19          new_pos.append(NP)
20
21      new_x = []
22      new_y = []
23
24      for i in range(0, 472):
25          # print(new_pos[i][0])
26          new_x.append(new_pos[i][0])
27          new_y.append(new_pos[i][1])
28
29      new_x = np.array(new_x)
30      new_y = np.array(new_y)
31
32      # ---- semi-major-axis ----
33
34      ma = (new_y.max() - new_y.min())
35      mm = (new_x.max() - new_x.min())
36
37      ma_arcsec = ma.304
38      print('Semi-major-axis, pixels:', ma)
39      print('Semi-major-axis, areseconds:', ma_arcsec)
40      print('Semi-major-axis, arcmins:', ma_arcsec/60)
41
42      mm_arcsec = mm.304
43      print('Semi-minor-axis, pixels:', mm)
44      print('Semi-minor-axis, areseconds:', mm_arcsec)
45      print('Semi-minor-axis, arcmins:', mm_arcsec/60)
46
47      # ---- Correcting ----
48
49      def parsecs(arcsecs, distance):
50          return (arcsecs/206625) distance
51
52      distance = 15e3 # \pm .8
53
54      print('Major-axis, KPC:', parsecs(ma_arcsec, distance))
55      print('minor-axis, KPC:', parsecs(mm_arcsec, distance))
56
57      ma_degrees = ma_arcsec/3600
58      mm_degrees = mm_arcsec/3600
59
60      def view_correction(distance, angle, i):
```

```
61          return (distance np.tan(angle))/np.cos(i)
62
63      i = 36.9
64
65      print('Correction of MA:', view_correction(parsecs(ma_degrees, dd),
            ma_degrees, i))
```

Listing 18 *Calculating angular size and applying corrections for viewing angle.*

```
1       vmin = M91_master_data.mean() - M91_master_data.std()
2       vmax = M91_master_data.mean() + M91_master_data.std()
3
4       # plt.plot(y_pos, x_pos, '.')
5       plt.scatter(Yn, Xn, s = 3)
6       plt.imshow(M91_master_image[0].data, cmap = 'magma', vmin = vmin, vmax =
            vmax)
7
8       # plt.imshow(bright_isolated_image, cmap = 'magma', vmin=0, vmax=1)
9       plt.colorbar()
10      plt.grid(True, which = 'major', alpha = .7)
11
12      plt.savefig('images/M91/M91_DBSCAN_overlay.png')
```

Listing 19 *Plotting overlay of DBSCAN on master-image.*

B.4 NN Serpentis Photometry Code

```
1       NNSer_master_image = fits.open('master_images/master_image_NNSer')
2       image_data = NNSer_master_image[0].data
3       image_data = np.delete(image_data, np.s_[0:50], axis=1)
4
5       # print(image_data)
6       # print(type(image_data))
7
8       mst_std = mad_std(image_data)
9       daofind = DAOStarFinder(fwhm = 6., threshold = 5. mst_std)
10      mst_sources = daofind(image_data)
11
12      print(type(mst_sources))
13
14      len(mst_sources['id'])
```

Listing 20 *Creating array of master sources.*

```
    for file in glob('Ser_Redlab_data/Ser/object/clear/O120308 '):
    image_data = fits.open(file)
    image_data = image_data[0].data
    print(image_data.mean())

for image in glob('Ser_Redlab_data/Ser/object/clear/O120308 '):
#     print(image)
    image = fits.getdata(image)
    print(image.mean())

    bkg_sigma = mad_std(image)
    daofind = DAOStarFinder(fwhm=6., threshold= 5 bkg_sigma)
    sources = daofind(image)
    # Get the positions of sources in the field from the table above.
    positions = np.transpose((sources['xcentroid'], sources['ycentroid']))

    # Set up aperture and annulus
    aperture = CircularAperture(positions, r=5.)
    annulus_aperture = CircularAnnulus(positions, r_in=10., r_out=15.)

    # Make a list of apertures
    apers = [aperture, annulus_aperture]

    # And run aperture photometry
    phot_table = aperture_photometry(image, apers)


    # We calculate the mean counts in each pixel in the background annulus,
        and then multiply by the area
    # in the aperture to get the total background counts within each aperture

    bkg_mean = (phot_table['aperture_sum_1'])/(annulus_aperture.area)
    bkg_sum = bkg_mean   aperture.area

    # Now we get the final table of background subtracted counts within each
        aperture
    final_sum = phot_table['aperture_sum_0'] - bkg_sum


    plt.imshow(image, vmin=np.mean(image)-np.std(image), vmax=np.mean(image)+
        np.std(image), cmap = 'magma')
#     print(np.mean(image))
```

```
40      plt.annotate('D', (mst_sources['xcentroid'][34], mst_sources['ycentroid'
            ][34]), color = 'blue')
41      plt.annotate('NN', (mst_sources['xcentroid'][51], mst_sources['ycentroid'
            ][51]), color = 'blue')
42      plt.annotate(''' 1' ''', (mst_sources['xcentroid'][73], mst_sources['
            ycentroid'][73]), color = 'blue')
43      plt.annotate('A', (mst_sources['xcentroid'][52], mst_sources['ycentroid'
            ][52]), color = 'blue')
44      plt.annotate('C', (mst_sources['xcentroid'][65], mst_sources['ycentroid'
            ][65]), color = 'blue')
45      plt.annotate('B', (mst_sources['xcentroid'][66], mst_sources['ycentroid'
            ][66]), color = 'blue')
46  #     plt.xlim(200,1000)
47  #     plt.ylim(1100,0)
48
49      plt.gca().invert_xaxis()
50      plt.gca().invert_yaxis()
51
52      plt.colorbar()
53      plt.show()
```

Listing 21 *Marking standard stars in each image*

```
1
2
3       # ---- Plotting Objects ----
4
5       vmin = image_data.mean() - image_data.std()
6       vmax = image_data.mean() + image_data.std()
7
8       plt.imshow(image_data, vmin = vmin, vmax = vmax, cmap = 'magma')
9       # plt.scatter(M91mst_sources['xcentroid'], M91mst_sources['ycentroid'],
            alpha = 0.3, color = 'green')
10      plt.gca().invert_yaxis()
11      # plt.xlim(2000, 750)
12      # plt.ylim(500, 1750)
13      # plt.gca().invert_xaxis()
14      plt.colorbar()
15      plt.savefig("images/Ser/unmarked_photometry_NNSer.svg")
16
17      for i in range(0, len(mst_sources)):
18          plt.annotate(i, (mst_sources['xcentroid'][i], mst_sources['ycentroid'
                ][i]), color = 'blue')
```

Listing 22 *Plotting master image and marked sources (zoomed).*

```
    # ---- Plotting Objects ----

    vmin = image_data.mean() - image_data.std()
    vmax = image_data.mean() + image_data.std()

    plt.imshow(image_data, vmin = vmin, vmax = vmax, cmap = 'magma')
    # plt.scatter(M91mst_sources['xcentroid'], M91mst_sources['ycentroid'],
        alpha = 0.3, color = 'green')
    plt.gca().invert_yaxis()
    plt.gca().invert_xaxis()
    plt.colorbar()
    plt.annotate('D', (mst_sources['xcentroid'][34], mst_sources['ycentroid'
        ][34]), color = 'blue')
    plt.annotate('NN', (mst_sources['xcentroid'][51], mst_sources['ycentroid'
        ][51]), color = 'blue')
    plt.annotate(''' 1' ''', (mst_sources['xcentroid'][73], mst_sources['
        ycentroid'][73]), color = 'blue')
    plt.annotate('A', (mst_sources['xcentroid'][52], mst_sources['ycentroid'
        ][52]), color = 'blue')
    plt.annotate('C', (mst_sources['xcentroid'][65], mst_sources['ycentroid'
        ][65]), color = 'blue')
    plt.annotate('B', (mst_sources['xcentroid'][66], mst_sources['ycentroid'
        ][66]), color = 'blue')
    # plt.title('Comparison Stars Labelled with NNSer')
    plt.savefig("images/Ser/marked_photometry_NNSer.svg")
```

Listing 23 *Plotting master image and marked sources.*

```
    # ---- Photo-Metry ----

    image_data = r_object_avg

    mst_std = mad_std(image_data)
    daofind = DAOStarFinder(fwhm = 6., threshold = 5. mst_std)
    mst_sources = daofind(image_data)

    coords = np.transpose((mst_sources['xcentroid'], mst_sources['ycentroid'])
        ) # transposes so the array is line correctly with projected image.
    # print(coords, np.shape(coords), type(coords))
```

Listing 24 *Photometry*

```
def SN(source_counts, sky_counts, pixels_source, pixels_sky, R):
    return source_counts/np.sqrt(source_counts + pixels_source (1 +
        pixels_source / pixels_sky ) (sky_counts + R 2) )

def SNsig(SN):
    return 2.5 np.log(1 + 1/SN)

def mean_err(std, n):
    return np.sqrt(std)/np.sqrt(n)
```

Listing 25 *Defining functions for errors.*

```
    aperture = CircularAperture(coords, r = 6)
    annulus_aperture = CircularAnnulus(coords, r_in = 10, r_out = 15)

    apers = [aperture, annulus_aperture]

    A_array = []
    B_array = []
    C_array = []
    D_array = []
    NN_Array  = []
    SN_data = []
    SN_sig_data = []
    zp_err = []

    for file in glob('Ser_Redlab_data/Ser/object/clear/O120308 '):
        image = fits.open(file)
        image = image[0].data
        # print(image.mean())

        phot_table = aperture_photometry(image, apers)

        bkg_mean = phot_table['aperture_sum_1']/annulus_aperture.area
        bkg_sum = bkg_mean aperture.area

        final_sum = phot_table['aperture_sum_0'] - bkg_sum

        magnitude = -2.5 np.log10(final_sum)

```

```
29          # ---- definie array of zero points for r filter ----
30
31          A_ZP =  -magnitude[34] + 15.8
32          B_ZP =  -magnitude[52] + 15.1
33          C_ZP =  -magnitude[65] + 13.7
34          D_ZP = - magnitude[34] + 13.7
35
36          zeropt = np.array([A_ZP, B_ZP, C_ZP, D_ZP]) # array of zero-points
37          zp_avg = zeropt.mean()
38
39          A_array = np.append(A_array, A_ZP) # arrays of zeropoints from each
                object image. Just used for observation.
40          B_array = np.append(B_array, B_ZP)
41          C_array = np.append(C_array, C_ZP)
42          D_array = np.append(D_array, D_ZP)
43
44          NN_Flux = final_sum[51]
45          NN_ZP = magnitude[51] + zp_avg
46          NN_Array = np.append(NN_Array, NN_ZP)
47
48          # ---- Error ----
49
50          a = final_sum[51]
51          b =  phot_table['aperture_sum_1'][51]
52          c = annulus_aperture.area
53          d = aperture.area
54          e = 3.37
55
56          SN_data.append(SN(a, b, c, d, e))
57          SN_sig_data.append(SNsig(SN(a, b, c, d, e)))
58
59          zp_err.append(mean_err(zeropt.std(), len(zeropt)))
60
61
62
63      # ---- writing to .csv files ----
64
65      aper_sum = []
66      f_sum = []
67      # print(f_sum)
68
69      for i in range(0, len(phot_table['aperture_sum_1'])):
```

```
70          aper_sum = np.append(aper_sum, phot_table['aperture_sum_1'][i])
71
72      for i in range(0, len(final_sum)):
73          f_sum = np.append(f_sum, final_sum[i])
74
75      # print(aperture.area)
76      # print(annulus_aperture.area)
77
78      Output = [f_sum, zeropt, aper_sum, aperture.area, annulus_aperture.area]
```

Listing 26 *Performing r' calbration and calculation of magnitude.*

```
1       # ---- Obtaining Universal Time from Orginal Fit files ----
2
3       t = []
4
5       for file in  glob('lab_data/Ser/object/clear/O120308 '):
6           image = fits.open(file)
7           time = (image[0].header['UT'])
8           t.append(time)
9
10
11      t_sp = []
12
13      for i in range(0, len(t)):
14          t_s = t[i].split(':')
15          t_sp = np.append(t_sp, t_s)
16
17      # print(len(t))
18      # print(len(t_sp))
19
20      hour = []
21      mins = []
22      seconds = []
23
24      for i in range(0, len(t)):
25          hour_v = t_sp[0 + i 3]
26          hour = np.append(hour, hour_v)
27
28          mins_v = t_sp[1 + i 3]
29          mins = np.append(mins, mins_v)
30
31          seconds_v = t_sp[2 + i 3]
```

```
32          seconds = np.append(seconds, seconds_v)
33
34      # print(int(hour[1]) 3600 + int(mins[1]) 60 + float(seconds[1]))
35      # print(mins)
36      # print(seconds)
37
38      total_seconds = []
39
40      # ---- converting universal time to seconds ----
41      for i in range(0, len(t)):
42          total_seconds_v = int(hour[i]) 3600 + int(mins[i]) 60 + float(seconds[
                i])
43          total_seconds = np.append(total_seconds, total_seconds_v)
44
45      # ---- writing to dat files ----
46
47      dat = np.array([NN_Array, total_seconds])
48
49      a= np.column_stack((dat))
50      # print(a)
51      hdrtxt='Magnitude, Seconds'
52      np.savetxt('saved_data/light_curve.dat', a, fmt='%f')
53
54      # np.savetxt('saved_data/jb_vfreefall.dat', a, delimiter=',', header =
            hdrtxt)
```

Listing 27 *Obtaining Universal Time from Orginal Fit files.*

```
1       # ---- Fitting the Data ----
2
3       func = interp1d(total_seconds, NN_Array, kind = 'linear')
4       fitted_magnitude = func(np.linspace(total_seconds.min(), total_seconds.max
            (), 1000))
5       fitted_time = np.linspace(total_seconds.min(), total_seconds.max(), 1000)
6
7       len(fitted_magnitude)
8
9       plt.scatter(total_seconds, NN_Array)
10      plt.plot(fitted_time, fitted_magnitude)
11      plt.gca().invert_yaxis()
12      plt.grid()
13      plt.xlabel('Time (seconds)')
14      plt.ylabel('Magnitude')
```

```
15      # plt.title('Light Curve NNSer')
16      plt.savefig("images/Ser/light_curve_fitted.svg")
```

Listing 28 *Intial light-curve plot.*

B.5 Error Analysis code

```
1       # ---- Functions ----
2
3       def master_sigma(name, a, b, x, y, std):
4
5           files = glob(name)
6
7           print(len(files))
8
9           #defining arrays and variables
10          all_data = []
11          i = 0
12
13          while i < len(files):
14              # using a loop to obtain data for all pixels
15              pixel_data = np.delete((fits.open(files[i])[0].data[a:b, x:y]),
                    np.s_[0:50], axis = 1)
16              all_data.append(pixel_data)
17              print(np.shape(all_data))
18              i += 1
19
20          all_data_array = np.asarray(all_data)
21          print(np.shape(all_data_array))
22
23          all_data_transposed = np.concatenate(np.swapaxes((all_data_array), 0,
                2)) #transposing data to sigma clip
24
25          i = 0 #resetting counter
26          clipped_data = []
27
28          while i < len(all_data_transposed):
29              clipped_data.append(sigma_clip(all_data_transposed[i], sigma = std
                    ))
30              i +=1
31
32          clipped_data_transposed = np.asarray(clipped_data)
```

```
33
34          clipped_data_3D = np.swapaxes(np.array_split(clipped_data_transposed,
                len(all_data_array[0][1])), 2, 0)
35          print(np.shape(clipped_data_3D))
36
37          nonzero = np.nonzero(clipped_data_3D == 0)
38          count = np.count_nonzero(nonzero)
39          print(count) #counts number of clipped pixel in sample set
40
41          return np.average(clipped_data_3D, axis = 0)
```

Listing 29 *Sigma Clipping counting function*

```
1       magnitude, LM, seconds  = np.loadtxt("saved_data/light_curve.dat", unpack
            = True)
2       SN, SN_sig, ZP_sig, SNLM, LM_sig = np.loadtxt("saved_data/light_curve_err.
            dat", unpack = True)
3
4       seconds = seconds + exp
5       seconds = seconds - seconds[0]
6       mins = seconds/60
7
8       # ---- error prop ----
9
10      time_err = np.diff(mins)
11      print(np.shape(time_err))
12      time_err = np.append(time_err, time_err.mean())
13      print(np.shape(time_err))
14      time_err = time_err/2
15
16      # -- func more than one var --
17
18      mag_err = np.sqrt(SN_sig  2 + ZP_sig  2)
19      LM_err =  np.sqrt(LM_sig  2 + ZP_sig  2)
20
21      def sine_func(x, y, z):
22      return y np.sin(x z)
23
24  def lin_func(x, m, c):
25      return m x + c #equation of a line y = mx + c
26
27  def log(a, b, x):
28      return np.exp(a) np.exp(b x)
```

```
29
30 def quad_func(a, b, c, x):
31     return a (x 2) + b (x) + c
32
33 def tophat(x, base_level, hat_level, hat_mid, hat_width):
34     return np.where((hat_mid - hat_width/2. < x) & (x < hat_mid + hat_width
           /2.), hat_level, base_level)
35
36
37 # --- Calculating Optimum Parameters and Covariance Matrix ---
38
39 popt1, pcov1 = curve_fit(lin_func, mins, LM)
40 LM_F = lin_func(mins, popt1[0], popt1[1])
41
42 mins_s = np.linspace(mins.min(), mins.max(), 1000)
43
44 # ---
45 a = magnitude[0:5]
46 b = magnitude[12:18]
47
48 baseline = (np.concatenate((a, b), axis=0)).mean()
49
50 midpoint = mins[6:11].min() + (mins[6:11].max() - mins[6:11].min())/2
51
52 tphat = tophat(mins_s, baseline, 1, midpoint, 1)
53
54 tpaht = np.abs(tphat)
55 plt.plot(mins_s, tpaht)
56 plt.gca().invert_yaxis()
57
58
59     from astropy.modeling.models import Box1D
60
61     # --- Box1D (amp, baseline, width 1)
62
63     a = magnitude[0:5]
64     b = magnitude[12:18]
65
66     baseline = (np.concatenate((a, b), axis=0)).mean()
67
68     amp = magnitude[7:10].mean() - baseline
69
```

```
70      midpoint = mins.max()/2
71
72      hat = Box1D(amp, midpoint - 1, 10)
73
74      x = mins
75
76      init = Box1D()
77
78      y = hat(x) + baseline
79
80      plt.gca().invert_yaxis()
81      plt.scatter(mins, magnitude)
82      plt.plot(x, y)
83
84
85
86      LM_fit = func(mins)
87
88
89      # ---- Plot Data points ----
90
91      plt.scatter(mins, magnitude, color = 'red')
92      plt.scatter(mins[4:12], LM[4:12], color  = 'green')
93
94      # ---- Plotting the Data using straight fitted functions ----
95
96      plt.plot(mins[4:12], LM_F[4:12], color = 'black') #using calculated
            parameters to plot best fit
97
98
99      # plt.plot(mins_s, tpaht)
100
101     plt.plot(x, y, color = 'black')
102
103     plt.grid()
104     plt.gca().invert_yaxis()
105
106     plt.errorbar(mins, magnitude, xerr = time_err, yerr = .15 mag_err, capsize
            = 5, ls='none', color = 'red')
107     plt.errorbar(mins[4:12], LM[4:12], xerr = 0, yerr = .15 LM_err[4:12],
            capsize= 5, ls='none', color = 'green')
108     plt.xlabel('Time (Mins)')
```

```
109     plt.ylabel('Magnitude')
```

Listing 30 *Light Curve error calculations*

```
1     # ---- Calculation and Uncertainty on Eclipse time ----
2     mins_err =  unp.uarray(mins, time_err)
3     # print(mins_err.mean())
4
5     c1 = mins_err[4:6]
6     c2 = mins_err[10:12]
7
8     # print(c1, c2)
9
10    p1 = c1[0]
11    p2 = c1[1]
12
13    p3 = c2[0]
14    p4 = c2[1]
15
16
17    ES = p2 + (p2 - p1)/2
18    EE = p4 + (p4 - p3)/2
19
20    print(ES, EE)
21
22    print('Eclipse duration in mins:', EE - ES)
```

Listing 31 *Calculation and Uncertainty on Eclipse time*

```
1
2     LM_fit = func(mins)
3
4
5     # ---- Plot Data points ----
6
7     plt.scatter(mins, magnitude, color = 'red')
8     plt.scatter(mins[4:12], LM[4:12], color  = 'green')
9
10    # ---- Plotting the Data using straight fitted functions ----
11
12    plt.plot(mins[4:12], LM_F[4:12], color = 'black') #using calculated
          parameters to plot best fit
13
```

```
14
15      # plt.plot(mins_s, tpaht)
16
17      plt.plot(x, y, color = 'black')
18
19      plt.grid()
20      plt.gca().invert_yaxis()
21
22      plt.errorbar(mins, magnitude, xerr = time_err, yerr = .mag_err, capsize=
            5, ls='none', color = 'red')
23      plt.errorbar(mins[4:12], LM[4:12], xerr = 0, yerr = LM_err[4:12], capsize=
             5, ls='none', color = 'green')
24      plt.xlabel('Time (Mins)')
25      plt.ylabel('Magnitude')
26
27      # ---- Plot Data points ----
28
29  plt.scatter(mins, magnitude, color = 'red')
30  plt.scatter(mins[4:12], LM[4:12], color  = 'green')
31
32  # ---- Plotting the Data using straight fitted functions ----
33
34  plt.plot(mins[4:12], LM_F[4:12], color = 'black') #using calculated parameters
        to plot best fit
35
36
37  # plt.plot(mins_s, tpaht)
38
39  plt.plot(x, y, color = 'black')
40
41  plt.grid()
42  plt.gca().invert_yaxis()
43
44  plt.axvline(x = a)
45  plt.axvline(x = b)
46  # plt.fill_betweenx(mins, a, x2 = b, alpha = '.5')
47
48  # plt.errorbar(mins, magnitude, xerr = time_err, yerr = mag_err, capsize= 5,
        ls='none', color = 'red')
49  # plt.errorbar(mins[4:12], LM[4:12], xerr = 0, yerr = LM_err[4:12], capsize=
        5, ls='none', color = 'green')
50  plt.xlabel('Time (Mins)')
```

```
51 plt.ylabel('Magnitude')
```

Listing 32 *Light Curve Plots with errors.*