CONTINUED OBSERVATIONS OF THE VARIABLE STAR V577 OPHIUCHI, A BINARY SYSTEM WITH A δ SCUTI COMPONENT

by

Natalie M. Porter

A senior thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Bachelor of Science

Department of Physics and Astronomy

Brigham Young University

August 2008

BRIGHAM YOUNG UNIVERSITY

DEPARTMENT APPROVAL

of a senior thesis submitted by

Natalie M. Porter

This thesis has been reviewed by the research advisor, research coordinator, and department chair and has been found to be satisfactory.

Date	Eric Hintz, Advisor
Date	Eric Hintz, Research Coordinator
Date	Ross Spencer, Department Chair

ABSTRACT

CONTINUED OBSERVATIONS OF THE VARIABLE STAR V577 OPHIUCHI, A BINARY SYSTEM WITH A δ SCUTI COMPONENT

Natalie M. Porter

Department of Physics and Astronomy

Senior Thesis

The brightness of the system V577 Ophiuchi varies for two reasons. It is an eclipsing binary system so the brightness changes as the stars pass in front of each other. Also, one of the components is a δ Scuti star whose luminosity changes because of the star's pulsation. We are studying this system to better understand the δ Scuti component. We used differential photometry, which compares the magnitude of the variable star with the magnitude of a constant luminosity star. Initial measurements of the period and amplitude of the δ Scuti were made by isolating the variation due to pulsation from the variation due to eclipses. The semi-amplitude of the pulsation is 0.022 mag in the V filter and the period is 0.06949 days. Both of these measurements are consistent with previously published results. A more thorough analysis of the data consisted of Fourier transforming the data to find the main frequency and any other frequencies. We found a total of 7 frequencies greater than one, one of which

is the double of the main frequency. We also calculated each time of maximum light and compared them to the observed time. The difference between the two appeared to be slightly sinusoidal. Based on these findings, we conclude that the pulsation of the δ Scuti does not have significant long term variation, though there are more than just the main frequency present.

ACKNOWLEDGMENTS

I would like to thank Dr. Eric Hintz for all of his help on reducing and analyzing the data, as well as on this paper. Many thanks go to Paul Iverson and Craig Swenson for the use of their scripts, their assistance in changing the scripts to fit my data and their help in resolving errors. I would also like to thank them for their encouragement as I attempted to write my own scripts. I would like to thank all of those that observed with me: Tiffany Brown, Amanda Henderson, McKay Bonham, Christie Pennington, Jeremy Schoonmaker and Kent Slack. I would also like to thank my husband, Ben, who let me observe and helped me with EXCEL. Finally, I would like to thank Maureen Hintz, Dr. Rees and Dr. Moody for their help on this paper.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Background

1.1 Background on Variable Stars

A variable star is one whose brightness changes. There are two main classifications of variable stars: intrinsic and line-of-sight. The brightness of intrinsic variables changes due to physical changes within the star. The main categories of intrinsic variables are listed in Table 1.1. These stars are classified based on the cause of variation. Sub-classifications are determined by the duration of the period, the amplitude of the variation, the spectral type, and the chemical composition of the star. The sub-classification groups are usually named after the first star discovered of that type. For example, a δ Scuti variable is a sub-classification of a pulsating variable star. They are characterized by low amplitude luminosity changes and short periods. The light curves of these stars are sinusoidal with typical amplitudes ranging from 0.003 to 0.9 magnitudes and typical periods ranging from 0.01 to 0.2 days. The spectra range from types A0-F5 III-V. In some δ Scuti stars, the variation in luminosity is irregular and occasionally stops completely (General Catalogue of Variable Stars (2004)).

Line-of-sight variables change because of the way we see them. They are often binary systems in which two (or more) stars pass in front of one another. As one star passes in front of the other, it blocks the light from its companion, making the system look dimmer. A typical binary light curve is characterized by a large drop in luminosity as the primary component passes in front of its companion, followed by a period of constant brightness when both stars are visible. There is a smaller drop in luminosity as the secondary component passes in front of the primary. If the system undergoes a total eclipse, the light curve will have flat-bottomed primary drop. The

Table 1.1. Types of Intrinsic Variable Stars and Their Classifications

Physical Classification	Description
Eruptive	Stars varying in brightness because of violent processes and flares occurring in their chromospheres and coronae due to shell events or mass outflow in the form of stellar winds of variable intensity and/or interaction with the surrounding interstellar medium.
Pulsating	Stars showing periodic expansion and contraction of their surface layers. The pulsations may be radial or nonradial. A radially pulsating star remains spherical in shape, while in the case of nonradial pulsations the star's shape periodically deviates from a sphere, and even neighboring zones of its surface may have opposite pulsation phases. Subclassifications depend on the period value, the mass and evolutionary status of the star, and on the scale of pulsational phenomena.
Rotating	Stars with nonuniform surface brightness and/or ellipsodial shapes. The nonuniformity of surface brightness distributions may be caused by the presence of large spots or due to thermal or chemical inhomogeneity caused by a tilted magnetic field.
Cataclysmic	Stars showing outbursts caused by thermonuclear burst processes in their surface layers (novae) or deep in their interiors (supernovae). Used for variables that show novalike outbursts caused by rapid energy release in the surrounding space and also for objects not displaying outbursts but resembling explosive variables at minimum light by their spectral (or other) characteristics. The majority of explosive and novalike variables are close binary systems, and that the hot dwarf component of the system is surrounded by an accretion disk formed by matter lost by the other.
Other	These include intense variable X-ray sources and others whose natures are not fully known.

classification of line-of-sight variables is based on the shape of the light curve, and the physical characteristics of the system (General Catalogue of Variable Stars (2004)). If either of the stars are also intrinsic variables, it leads to even more variation in the brightness of the system. Occasionally, a star is variable for a combination of (multiple) reasons. V577 Ophiuchi is one such star.

1.2 Previous Research

V577 Oph is both an eclipsing and a pulsating variable. It is a binary system in which one of the components is a δ Scuti variable. V577 Oph does not experience total eclipses so its light curves are not flat-bottomed. V577 has been studied over the course of at least 20 years and appears to have a stable δ Scuti component. Previous research done by Shugarov and cited by Volkov et al. (1990) estimates the eccentricity of the orbit to be around 0.22, which is supported by Deithelm et al. (1993) who give the uncertainty of that value to be ±0.08. Volkov et al. (1990) states that the binary system has orbital period of 6.079084 days that is stable over the 2.5

years they observed it. The ephemeris equation they give for the primary eclipse is $HJD = 2447406.1955 + 6.079084x$ and the ephemeris equation they give for the δ Scuti pulsation is $HJD = 2447620.379 + 0.0694909x$ where x is the cycle number in both equations. Volkov et al. (1990) also mentions an increase of reddening during the minimum and hypothesize that there could be a third body that is cooler than the other two components. Deithelm et al. (1993) give the period of pulsation for the δ Scuti component as 0.07 days, which matches closely with the 0.069491 days period given by Zhou et al. (2001). Volkov et al. (1990) give the amplitude of the δ Scuti in the B, V, and R filters as 0.052 mag, 0.070 mag, and 0.040 mag respectively. Zhou et al. (2001) give only the semi-amplitude for the V filter at 0.0289 mag which is a little lower than the value given by Volkov et al. (1990), but still close.

1.3 Motivation

We have examined V577 Oph to further clarify the nature of the pulsating component. In doing so, we hope to gain a better understanding of the mechanisms driving the pulsation for this and similar stars. Currently, little is known about δ Scuti variables and studying these stars is complicated by the number of unknown quantities required in modeling them. In a non-eclipsing binary system, it is hard to accurately determine the masses of the individual stars. However, in an eclipsing system, the determination of the masses is relatively simple. Since V577 Oph is an eclipsing binary, we are able to determine the masses of the components. Since one of the components is a δ Scuti, knowing the mass allows us to create more accurate models for these variable stars. However, if the stars in the binary are too close, the pulsation of the δ Scuti may be driven gravitationally instead of internally. V577 Oph is one of a few δ Scuti eclipsing binary systems in which the stars are well separated, making it an excellent system to study. We have a total of 28 nights of data: eight nights in 2001, six in 2003, one in 2005, seven in 2006, and five in 2007. Once we reduced the images using the standard IRAF reduction package, we used differential photometry to find the differential magnitude of V577 Oph. From the differential magnitude, we focused on the variation due to the δ Scuti component. We measured the amplitude of the variation and calculated an equation to predict the pulsation of the δ Scuti component.

1.4 Reduction

Data is recorded on a charge-coupled device (CCD). A CCD is comprised of pixels that collect photons as they strike the chip. The photons are converted into electrons that are stored in each pixel well. The number of electrons in each pixel is related to the amount of light absorbed, which is used to create the image. However, the raw images coming from the CCD do not accurately measure the amount of light hitting the frame.

There are three different types of noise that must be removed before useful information can be extracted. The first level of noise is due to electrons registered when the electronics are turned on. Each pixel will have a different residual level that must be subtracted from the other calibration frames and the object frames to provide a common zero point. To measure the residual levels, we take images using a zero second exposure time. These images are called bias or zero frames. The second level of noise is due to electrical currents in the equipment, which cause the CCD to register electrons that are not due to photons. To correct for this, we record images with the shutter closed so no light hits the CCD and the only electrons registered are noise. We use the same exposure time as for our object frames because the number of extra electrons the CCD registers depends on the length of the exposure. These calibration frames are called dark frames and are subtracted from the last type of calibration frames and the object frames. The third type of noise is due to the unequal sensitivity of each pixel, which causes an unequal absorbtion of photons. To account for this, we take short exposures of a uniformly bright and colored area of sky. These images, called flat frames, are scaled to an average of one and divided out of the object frame.

1.5 Photometry

1.5.1 Aperture Photometry

In order to find the magnitude of several stars on a frame, we use a method called aperture photometry. Two circles are drawn centered on each star. The inner circle includes the star and totals the number of counts for the star and the sky. Partial pixel counts are calculated by what fraction of the pixel lies within the circle. The outer circle totals the number of counts between the two circles, which is called the sky count. The sky count is then subtracted from the counts within the inner circle to obtain the total counts for just the star. The counts are then converted into an instrumental magnitude for each star.

1.5.2 Differential Photometry

There are many things that affect the light measured from a given star, atmospheric conditions being one of the most important factors. The atmosphere scatters light and the more atmosphere the light has to travel through, the more it scatters. Astronomers use a unit called airmass to measure how thick the atmosphere is. One airmass corresponds to directly overhead; it is the smallest airmass possible. Data becomes unreliable when the airmass exceeds two, which occurs when looking 30 degrees above the horizon. Fluctuations in the atmosphere cause light to bend which prevents some of the light from reaching the telescope and causes the star to look dimmer than it really is. This also spreads the light over a greater area on the CCD. Because of this, the magnitudes we get from aperture photometry are not true apparent magnitudes and vary from frame to frame during the night, and differ from night to night. To compensate for this, we use a method called differential photometry. Within each of our frames, there are at least two reported constant stars; stars whose brightness do not vary. We average the instrumental magnitude of the the two constant stars and then subtract that magnitude from that of the variable star.

This difference is called the differential magnitude of the variable star and the variations in magnitude are only because of the star itself. This is done for each frame. If the apparent magnitude of the constant stars is well known, we can then convert the differential magnitude of the variable star into an apparent magnitude, which is consistent over all the nights of data. If we do not know the apparent magnitude of the standard stars, we calculate the average instrumental magnitude for the variable star for each night, and then find the difference between the average magnitude of the first night and the each of rest of the nights. The resulting difference is called the zero-point for each night. The zero-point is added to each night, which results in a differential magnitude that is consistent for all the data.

Chapter 2

Data Acquisition, Reduction, and Photometry

2.1 Observations

Data were acquired using the David Derrick Telescope (DDT) at the Orson Pratt Observatory on the campus of Brigham Young University. The DDT is a 0.4 m telescope with both a Cassagrain and a Newtonian focus. Most of the data were taken using the Cousins V filter with the exception of four nights that have some data taken using the Cousins B filter and two with some data taken using the Cousins R filter. Only the data taken in the Cousins V filter were used in analysis. The date of each night, with the number of frames taken in each filter are detailed in Table 2.1. The data cover a period of seven years, from 2001 to 2008. During that time, three different CCD's were used. The physical size of the CCD determines the area of sky that is imaged and the number of pixels in the chip, or plate scale, determine the resolution. The focus also has an affect on the resolution. Table 2.2 lists the different CCD's along with the focus they were used with, the plate scale and resulting field of view. The right ascension (RA) of V577 Oph is 18 hrs 16 min 45.8529 sec and the declination (δ) is $+06^\circ\ 54'\ 18.241''$.

Fig. 2.1 shows the star field of V577 Oph. V577 Oph is the star in the center, labeled star 1. The other numbered stars are comparison stars, with the exception of star 4, which was not used because it was saturated in most frames.

2.2 Reduction

The files coming from the CCD contain both the image and a header. The header contains details about each frame, such as when it was recorded. We used a

Table 2.1. Summary of Data

Year	Night	V Filter(frames)	B Filter(frames)	R Filter(frames)	Notes
2001	4-Jul	5	0	0	
	12-Jul	11	0	0	
	19-Jul	25	0	0	
	25-Jul	22	0	0	
	1-Aug	33	0	0	
	2-Aug	28	0	0	
	9-Aug	18	0	0	
	15-Aug	33	0	0	
2003	9-Jul	126	0	0	
	16-Jul	24	0	0	
	24-Jul	58	0	0	
	30-Jul	3	0	0	
	5-Aug	15	0	0	Not used because the recorded HJD is incorrect
	10-Aug	19	17	0	
2005	1-Oct	9	10	7	
2006	18-Jun	4	3	4	
	21-Jul	53	0	0	
	23-Jul	74	0	0	
	27-Jul	43	0	0	
	28-Jul	58	0	0	
	3-Aug	55	0	0	
	9-Aug	83	0	0	
2007	1-May	54	0	0	Not used in analysis because the differential magnitude is too high
	26-Jun	37	0	0	Not used in analysis because the differential magnitude is too low
	30-Jun	0	0	0	
	4-Aug	45	9	0	
	11-Aug	93	0	0	
2008	3-May	31	0	0	
	7-May	24	0	0	
	17-May	35	0	0	

Table 2.2. CCD Plate scale and Field of View

CCD	Focus	Plate Scale ($''$/mm)	Field of View ($'$)	Corresponding Data
Apogee 8	Cassegrain	40.64	16.64x16.64	4-Jul-01 to 10-Aug-03
ST-1001	Cassegrain	40.64	16.64x16.64	1-Oct-05 to 9-Aug-06
ST-10	Newtonian	127.01	31.30x21.00	1-May-07 to 17-May-08

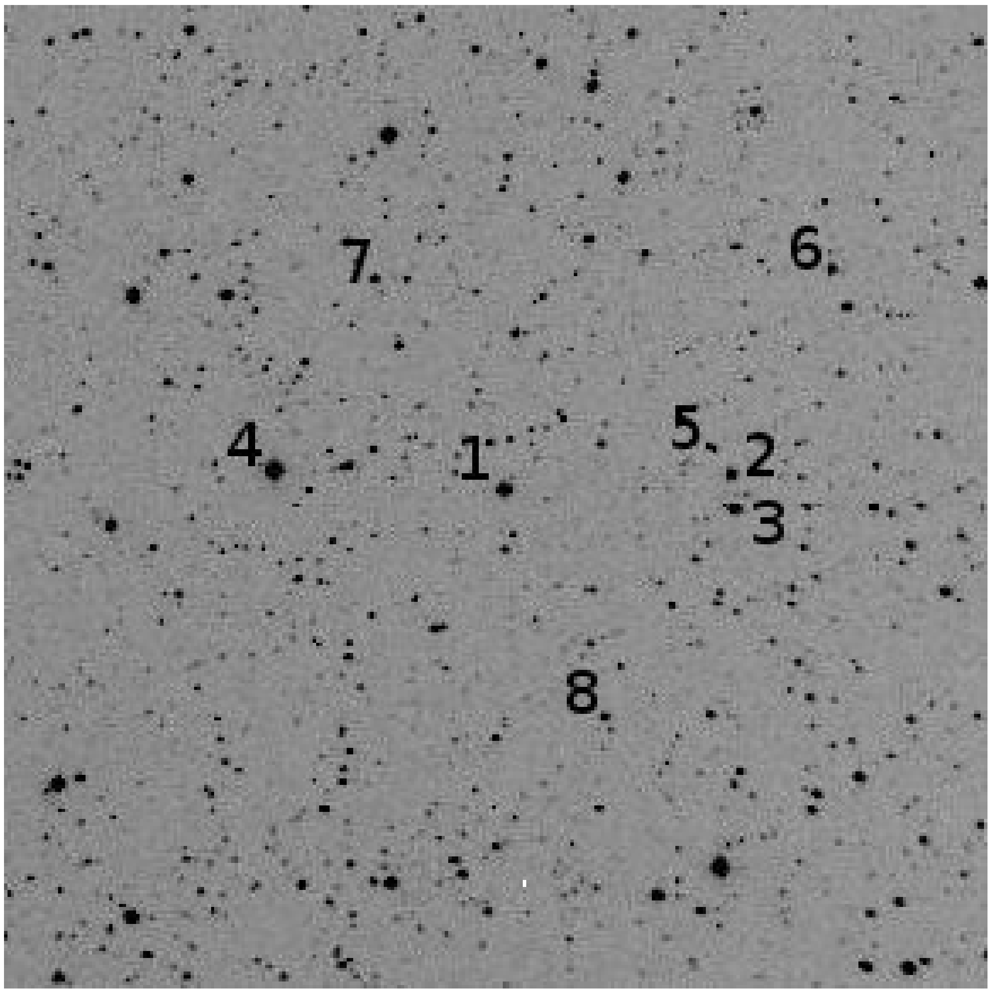


Figure 2.1: V577 Oph is the star in the center, labeled star 1. Star 4 was not used as a comparison star because it was saturated in most frames.

program called Image Reduction and Analysis Facility (IRAF) to reduce our images. Before applying the calibration frames to the images, the headers were edited to include the object, the observer, the airmass, which filter the frame was taken in, and converted the date to the Julian Date. The calibration frame headers were also edited to include the type of frame and the filter. To apply the calibration frames to the images, all of the zero frames were combined into a master zero frame. This was then used to correct the dark frames. The corrected dark frames where then combined into a master dark frame. Both the master dark frame and the master zero frame were used to correct the flat frames. The corrected flat frames were combined into a master flat frame. All three of the master frames were used to correct the object frames. Scripts written by a fellow research assistant, Paul Iverson, were used to facilitate this process. These scripts were designed to be used with IRAF and were altered to work with the data on V577 Oph.

2.3 Photometry

Once the images have been reduced, any bad frames must be removed. Occasionally, the pier that the telescope sits on shakes while data is being taken. This results in frames where the stars are smeared from circles into ellipses, or yields two sets of stars. Other bad frames are the result of a too bright background or the presence of clouds. To remove the bad frames, each frame has to be looked at individually and then the bad ones are deleted. A script that displays frames in succession and then deletes specified frames was used to facilitate this process.

Photometry was done using scripts designed to work with IRAF. The first series of scripts calculate the average gain and read noise for each night and put the values into the header. The read noise is electronic noise that is introduced as the image is downloaded from the CCD. Gain is the relationship between the number of photon hits and the number of electrons registered. With the gain, the number of electrons can be converted into a flux. Gain and read noise are calculated using two zeros and two flats. To get a more accurate average, each frame was divided into five sections, one in each corner and a section in the center which overlaps each of the others. The first script in the series calculates the gain and read noise for each of the five sections, using all of the flats and zeros for that night, for each filter. The second script takes the gain and read noise from each section, for each combination of flats and zeros, and averages it. The third script puts the averages into the header. Differential photometry is not affected too much by either gain or read noise because they are typically uniform throughout the night. The main purpose of calculating the gain and read noise was to align the frames, since the *align* command in IRAF requires values for both.

The *align* command also requires the value of the average Full Width Half Maximum (FWHM) and the standard sky deviation. The FWHM is how broad the star appears on the CCD. Even though stars are points of light,they are smeared into circles by the atmosphere so the total amount of light gets spread out. The FWHM is twice the distance from the peak brightness to the point where it has decreased to half the peak value. To standardize data, only the light within in the FWHM is counted for the star. Since the sky is not completely black, all the pixels in the CCD register some light. The sky value is the average instrumental magnitude of the sky, not including the magnitudes of the stars in the frame. Each pixel has its own deviation from the average sky value, found by subtracting the sky value of each pixel from the average. The standard sky deviation is the average of the individual deviations. The first script in the second series trims the edges off each frame to remove bad pixels and then calculates the average FWHM and the sky deviation. Both values are put into the header of each image. The second script in the series aligns the trimmed files generated by the FWHM script. To align the frames, the script selects several stars in the first frame and shifts the rest of the frames so that those stars are in the same position in each image. The purpose for aligning the frames is to facilitate the use of the *phot* command in IRAF. The third script in the second series is modified from one written by Dr. Eric Hintz, a professor at Brigham Young University. This script takes the FWHM from the header, which is also used in the *phot* command. The *phot* command generates a magnitude file for each star that is designated by a coordinate file. The coordinate file used specified a total of eight stars; V577 Oph and 7 potential constant stars. If the frames are not aligned, that coordinate file is different for each frame. With aligned frames, one coordinate file works for the entire night.

For more details on the scripts used, please see Appendix A.

Chapter 3

Analysis

3.1 Statistical analysis

As mentioned in Sec. 1.5.2, the magnitudes generated by the *phot* command in IRAF are not true apparent magnitudes so we use differential photometry to analyze the light curves. To calculate differential magnitudes, I used a program called VARSTAR, which was also written by Dr. Hintz (see Hintz et al. (1997) for details). The input file for VARSTAR is compiled using the magnitude files generated by the *phot* command. The input file contains the ID of each star, numbered as they were entered into the coordinate file, as well as the magnitude, the HJD, the filter and the airmass for each star. VARSTAR averages all the instrumental magnitudes for each frame. An individual star's magnitude is then subtracted from the average to give the differential magnitude for that star. VARSTAR also calculates the standard deviation for each star in units of magnitude, which is the error. Stars with the largest errors are removed and the process is repeated with those stars excluded, until the errors of the remaining stars drop to 0.01 mag or less. The remaining stars are usually constant stars, so differential magnitude changes for the removed stars are due solely to changes in luminosity or just large noise. VARSTAR creates output files for each star containing the HJD and the differential magnitude, as well as a log file, which records which stars were left in the average at each iteration, and the average differential magnitude for each star. Only two constant stars were used for night of data, though which stars remained varied from night to night. The output files from VARSTAR were imported to EXCEL for additional analysis. Fig. 3.1 shows the differential light curve of two of the constant stars. The light curves of these two stars have a little variation from a straight line, but are predominately constant.

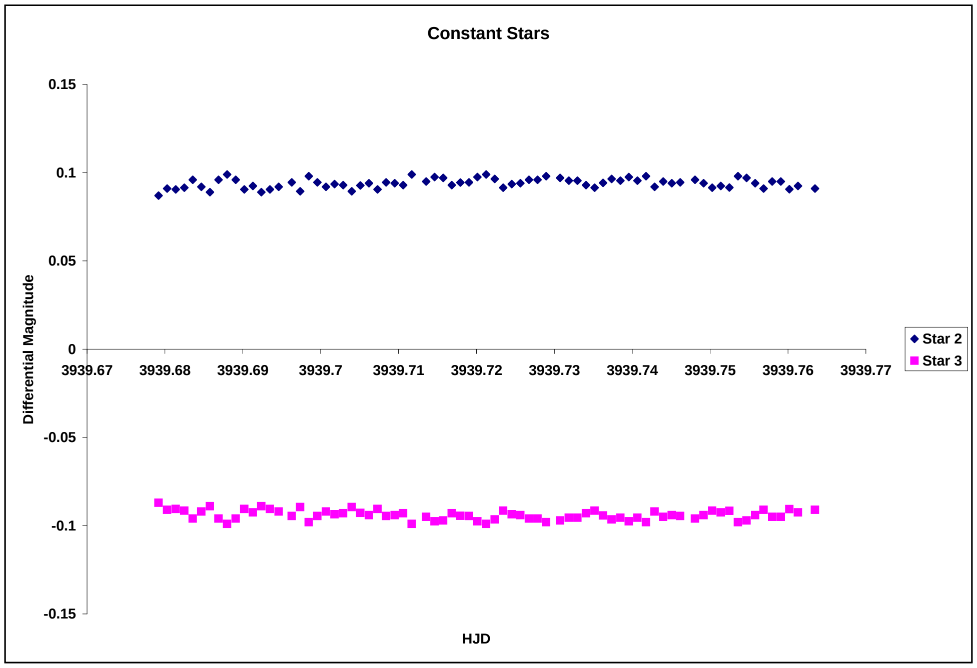


Figure 3.1: This is a light curve of the two constant stars for one night.

3.2 Initial Analysis

3.2.1 Adjusting the Differential Magnitudes

As mentioned in Sec. 1.5.2, the zero-point of each night of data is different. The differential magnitudes for each are taken from the VARSTAR log file and averaged. These average values are subtracted from that of the first night to give the zero-point for each subsequent night. This is done by filter, since each filter gives a slightly different magnitude. The first night corresponds to the first calendar night each filter was taken: 4 July 2001 for the V filter, 10 August 2003 for the B filter, and 1 October 2005 for the R filter. The zero-points are then added to each corresponding night to get consistent differential magnitudes. This was done using three different sets of zero-points. The first set contained the zero-points of only one star. The second set was composed of the average zero-points of all the stars, excluding V577 Oph and a star that was saturated in most frames. The third set was the average zero-points with any outliers excluded from the averages. The second and third sets differ only in the zero-points of several nights. The third set of zero-points yielded the most consistent differential magnitudes, so that one was used. Fig. 3.2 shows three separate nights, corrected with each of the different zero-point sets. The bottom graph illustrates the set that was used.

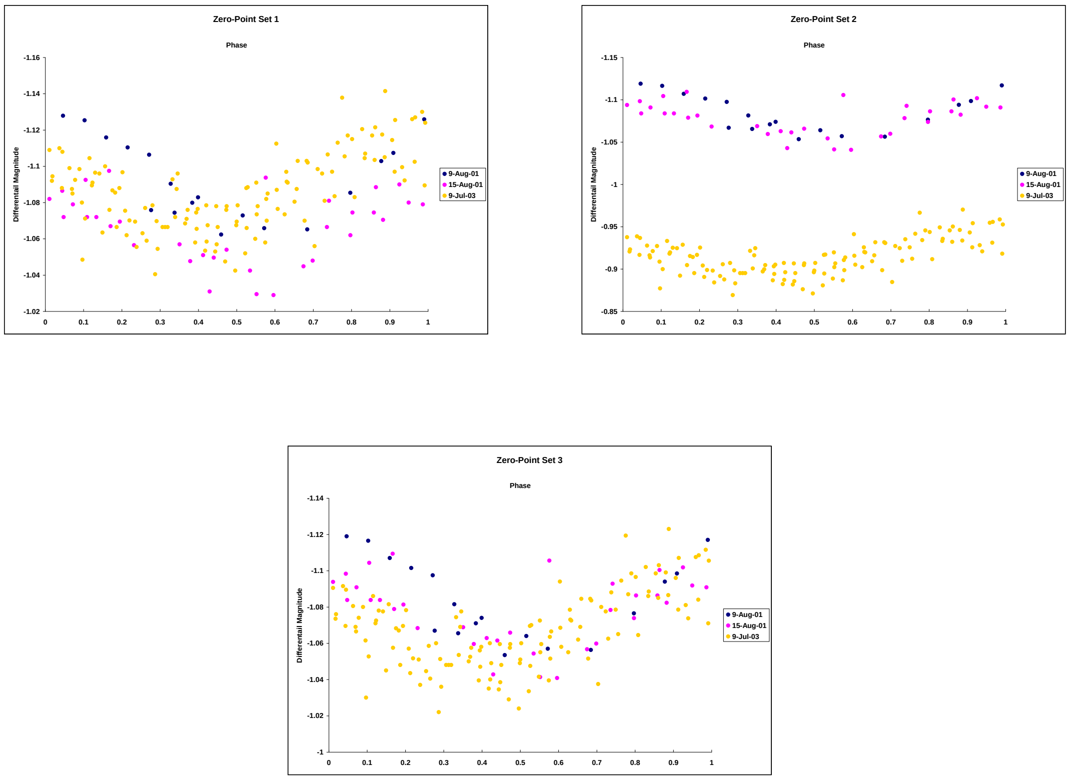


Figure 3.2: These three graphs illustrate the three different sets of zero-points. Each graph shows three separate nights, and each graph is corrected using a different set of zero-points. The zero-points used in the analysis are those used in the bottom graph.

3.2.2 Phasing the Data

Once all the differential magnitudes have been adjusted, the time scale needs to be adjusted as well. Since the data was taken over several years, it is difficult to see patterns in the differential magnitude of V577 Oph. Sec. 1.1 describes the typical light curves of both a binary system and a δ Scuti variable. Out of 28 nights of data, there are only four that show parts of the primary eclipse and only one that shows the secondary eclipse. To get a better idea of the shape of the light curve, we use a process called "phasing the data". Since previous research has been done on V577 Oph, the binary ephemeris equation published by Volkov et al. (1990) (see Sec. 1.2) was used. Phasing the data involves entering the HJD values into the ephemeris and solving for the cycle number. To get the fraction of the cycle, the whole number value is subtracted off. Phased light curves plot the adjusted data vs the fractional cycle number.

Fig. 3.3 shows the data after it has been phased using the binary ephemeris equation. The first dip in the light curve is the primary eclipse. Slight variations in the downward curve are due to the δ Scuti component. Only the latter part of the secondary eclipse is evident, since the data does not cover the whole binary period. Nevertheless, the upward curve is smooth, since during this eclipse, the δ Scuti component is behind its companion. During an eclipse, the only light we see is from the star in front. There are two nights which do not appear to fit with the rest of the data; one is too low and one is too high. We are not sure why but they were removed from consideration.

Since the curve of the secondary eclipse is smooth, the companion star of V577 Oph does not appear to be a variable star. Therefore, changes in the light curve, excluding the primary and secondary eclipses, are due only to the δ Scuti component. The data between the eclipses were phased using the δ Scuti ephemeris equation given by Volkov et al. (1990) (see Sec. 1.2). The process was the same as that detailed above using the binary ephemeris equation. Fig. 3.4 shows the phased

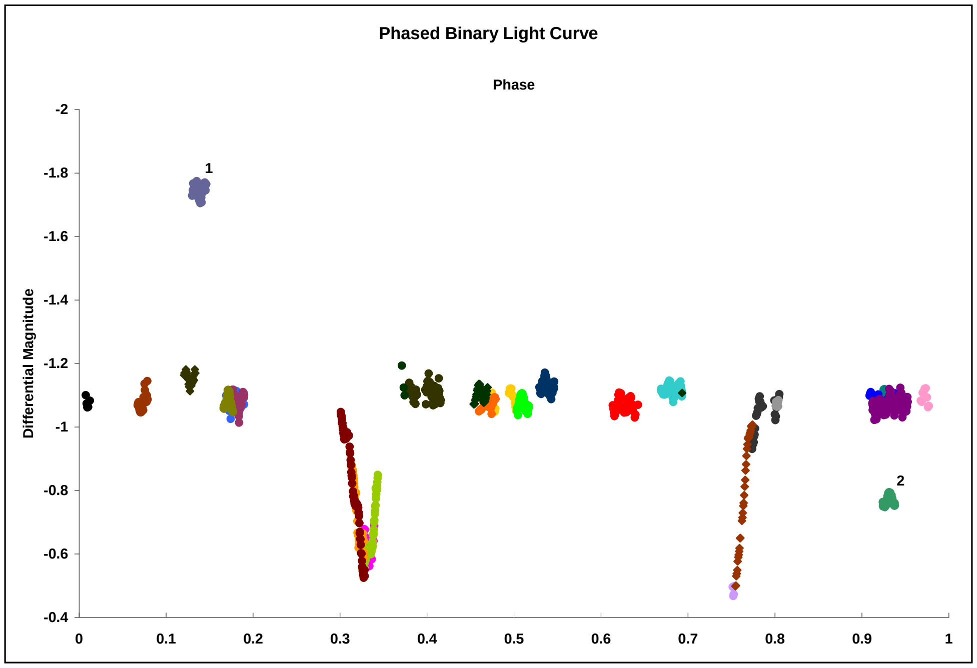


Figure 3.3: This figure shows the data after it has been phased using the binary ephemeris equation. The first dip is the primary eclipse. Only the latter half of the secondary eclipse was recorded. The nights labeled 1 and 2 were not used in analysis. Each different color corresponds to a different night.

light curve. The spread in magnitudes is roughly 0.09 mag which may be due errors in calculating the different zero-points. However, a closer examination of only two nights (see Fig. 3.5) shows that the peaks of individual nights occur at different points in the cycle. Each night is indicated by a different color. This may be caused by the presence of pulsation frequencies other than the published value. This also may be due to the fact that the δ Scuti star is alternatively moving toward and away from us, which changes the times between maxima that we measure. When the star is farther away from us, it takes longer for the light to reach us, since it has a larger distance to cover. Therefore, the time that we observe the peak in brightness will be later than we calculated. When the star is closer to us, it takes less time for the light to reach us, since there is a smaller distance to travel. Therefore, the time we observe the peak will be earlier than we expected. Since the star is alternatively moving farther and closer, the observed peak will switch between being earlier and later than we expect it, adding another frequency to the pulsation of the star.

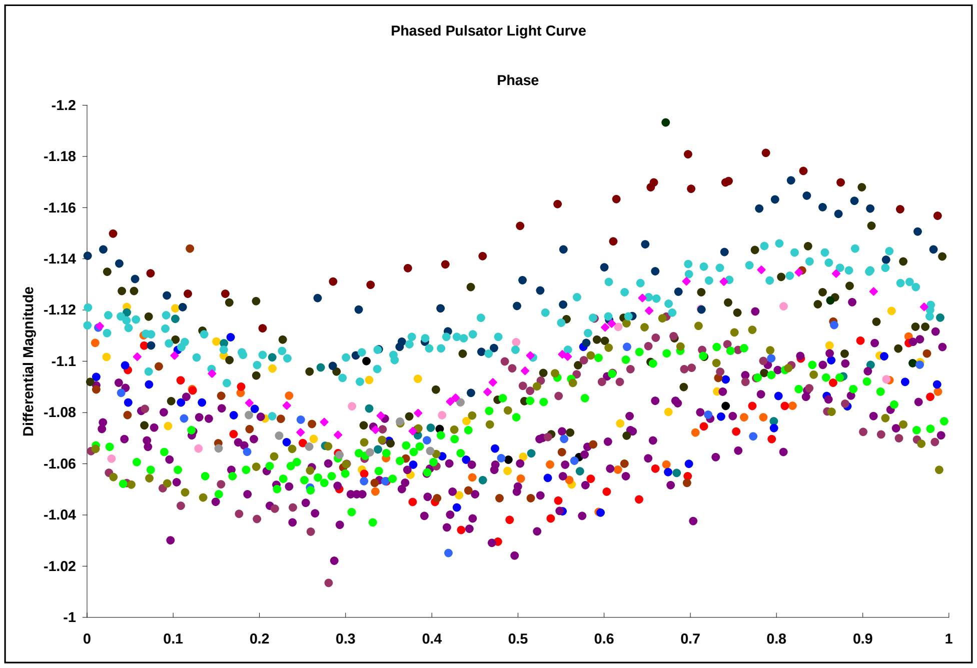


Figure 3.4: This figure shows the data after it has been phased using the δ Scuti ephemeris equation.

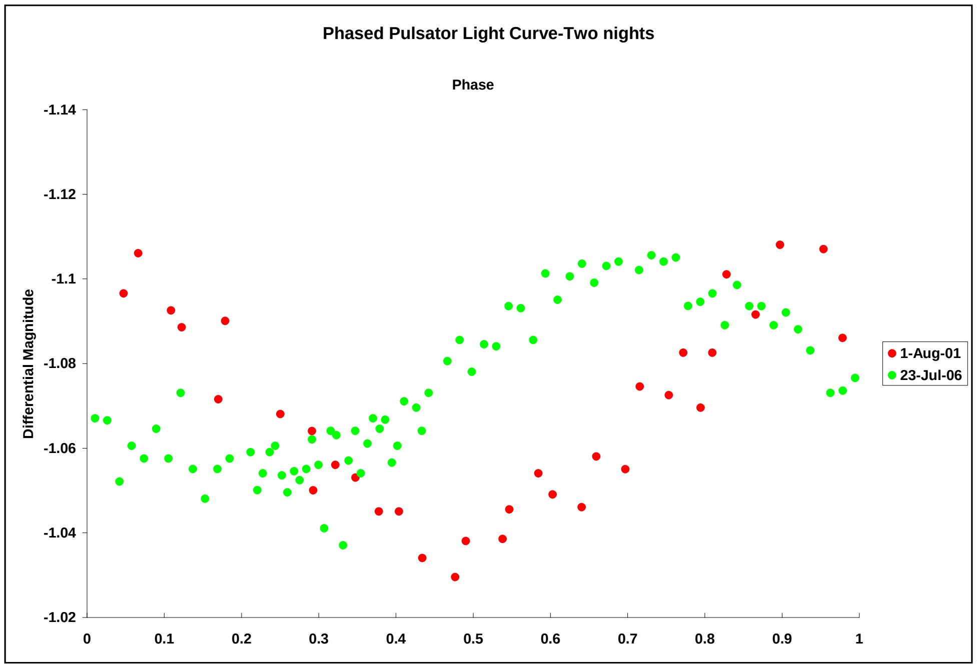


Figure 3.5: This figure shows two different nights, with the peak at slightly different cycle numbers.

3.3 Secondary Analysis

3.3.1 Period04

A more thorough analysis of the δ Scuti data was made using a program called Period04. Period04 does an analysis similar to a Fourier analysis(See Lenz and Breger (2005) for details), which fits a sinusoidal curve to the data. From that curve, the frequency of the data variation is calculated. To get an accurate fit, it is often necessary to include multiple frequencies in the equation. To obtain multiple frequencies, Period04 first calculates the most prominent frequency and subtracts it

Table 3.1. Frequencies and Amplitudes From Period04

Frequency	Corresponding Period	Amplitude	Comments
14.3904868	0.069490352	0.022275193	Primary frequency
2.80852102	0.35605929	0.007453959	
16.3862702	0.0610267	0.005659307	
7.92353629	0.126206275	0.005573664	
10.7465215	0.093053366	0.003848096	
12.845215	0.077850001	0.003715614	
28.7713225	0.034756831	0.002822814	Double the primary frequency

from the data using a method similar to that detailed in Sec. 3.3.2. Period04 can then go through and find the next most prominent frequency. We ran Period04 a total of ten times and obtained seven frequencies greater than one, which are detailed in Table 3.1. The first frequency we found was 14.3904868, which corresponds to a period of 0.069490352 days. This is very close to the published period of 0.069491 days. The semi-amplitude given by Period04 for that frequency is 0.0223, which is 0.0066 mag less than the published value of 0.0289 mag. The final frequency we found was 28.7713225, which is almost exactly double the initial frequency. This strongly supports the claim that 0.069490352 days is the primary period. However, we suspect that the other frequencies found by Period04 are important. Fig. 3.6 shows two separate nights of data and two different fit curves generated by Perio04. The two graphs on the right show each separate night of data and the fit curve using only the primary frequency. The two graphs on the left show the same two nights, but the fit curve uses all the frequencies from Period04. The second set of fit curves matches much more closely to the data, indicating that the other frequencies are actually present in the data.

3.3.2 O-C Diagram

An Observed-Calculated diagram plots the difference between when we think maxima will occur based on our calculated period and when the maxima actually

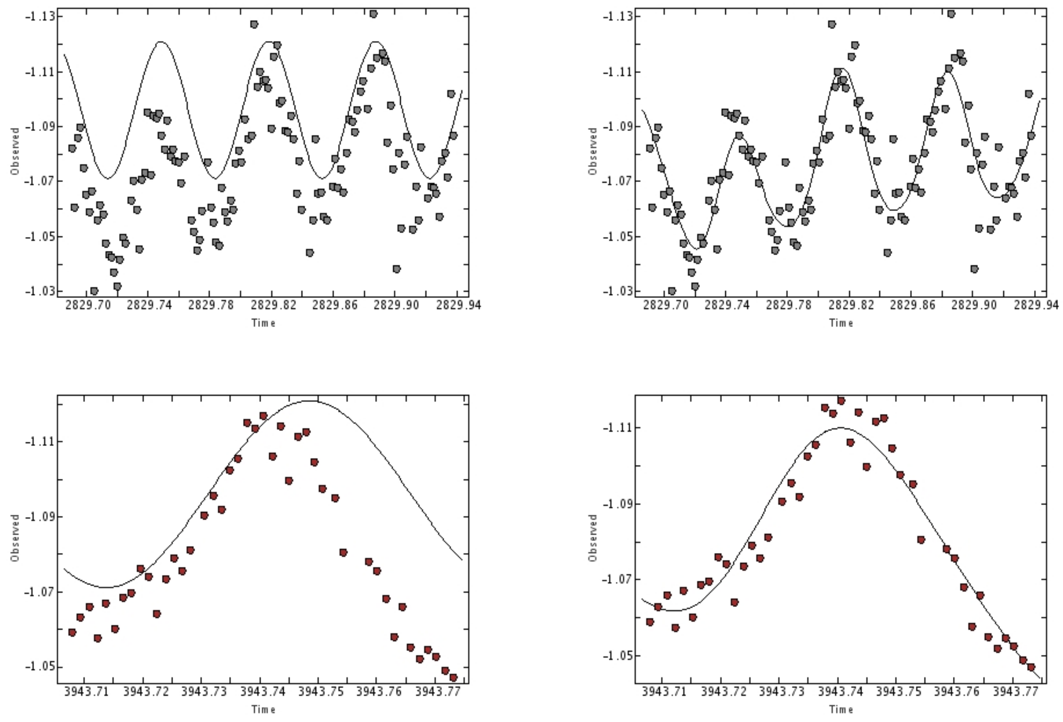
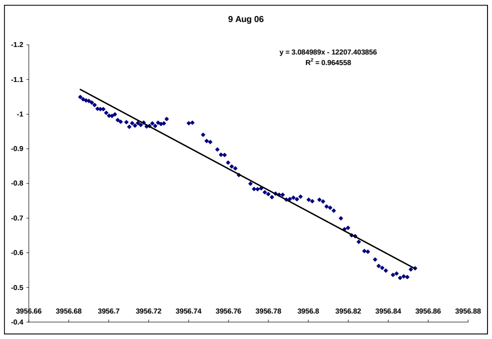


Figure 3.6: These four graphs show two different fits from Period04, along with the data from two different nights. The left two graphs have a fit line with only one frequency. The right hand graphs have a fit line with the frequencies listed in Table 3.1.

occur. If the data points are scattered randomly around zero, the diagram shows that the calculated period is correct. If there is a linear trend in the O-C diagram, it indicates that the calculated period is wrong; if the slope is positive, the calculated period is too long and if the slope is negative, the calculated period is too short. A parabolic O-C diagram, shows that the actual period is increasing or decreasing, depending on whether the curve is concave up or down, respectively. If the trend is sinusoidal, it means that the actual period is changing in a regular manner. Any other trend in the O-C diagram indicates that the period is changing irregularly.

To calculate the times of maximum light, we fit parabolic lines to each light curve containing a maximum in our data set. We then differentiated the resulting equations and set them equal to zero to find the HJD where the peak occurs. We also included several maxima that occurred during the primary eclipse. To remove the binary component of the light curve, we fit a curve to the data, and then subtracted the values of the curve from the data points. We used a linear fit for one night, 9 August 2006, since the data corresponds to the beginning of the eclipse. Since V577 Oph is not a total eclipsing binary, the primary minimum is parabolic shaped. We used a parabolic fit for three other nights, 12 July 2001, 28 July 2006, and 3 August 2006, since these three nights correspond to the eclipser primary minimum. Fig. 3.7 shows the linear fit and the resulting light curve. We used the published ephemeris equation to calculate the whole cycle number. Using our first night as a zero-point, we shifted the cycle numbers and then plotted them vs HJD (see Fig. 3.8). From that we calculated our own ephemeris equation, and used that to generate our calculated values. We then subtracted those values from the observed values and plotted the difference vs HJD to create the O-C diagram (see Fig. 3.9).



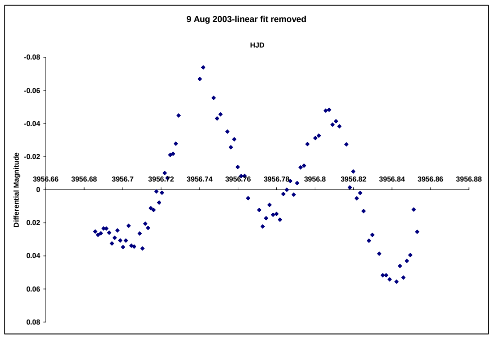


Figure 3.7: The graph on the left shows the linear fit and the data. The graph on the right shows the data after the linear values have been subtracted, leaving only the pulsation of the δ Scuti.

Since the data is spread out over so many years, it is difficult to see any short term trend in the O-C diagram. To fix this, we phased the O-C diagram as detailed

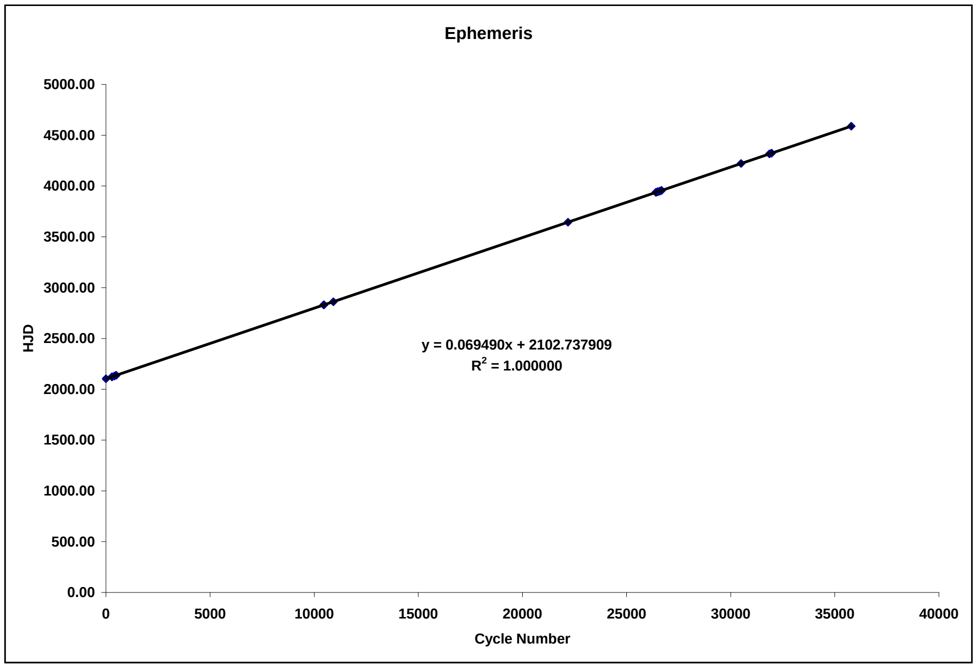


Figure 3.8: This graph shows the whole cycle number plotted against cycle number. The fit line is the line used to calculate the ephemeris equation.

in Sec. 3.2.2 using the binary ephemeris equation. Fig. 3.10 shows the observed-calculated difference plotted vs the binary cycle number instead of HJD. We included as many maxima as possible in our original phased O-C diagram (Fig. 3.10), and we suspect many of the points are not completely reliable. Fig. 3.11 includes only times of maximum light calculated with more than 15 data points in the parabolic fit and with an R^2 value of 0.8 or higher. The O-C diagram appears to have a sinusoidal pattern to it. The points on the first half the graph are above zero and those on the second half are below zero. We suspect this is due to the orbit of the δ Scuti star around its companion. However, there is enough scatter in the data to make this conclusion tenuous.

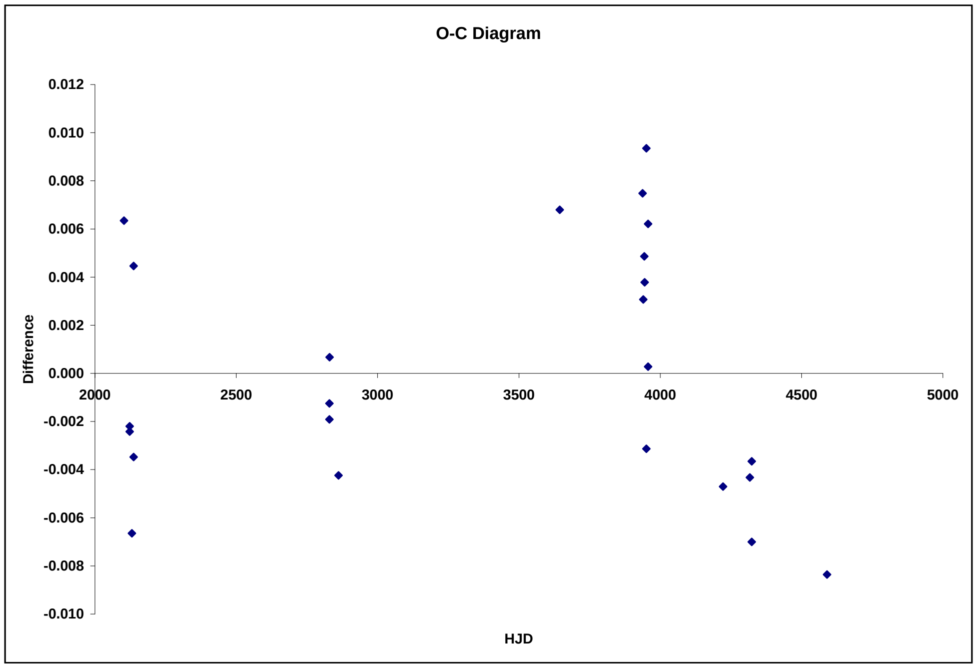


Figure 3.9: This graph shows the difference between the observed and the calculated maxima plotted vs HJD.

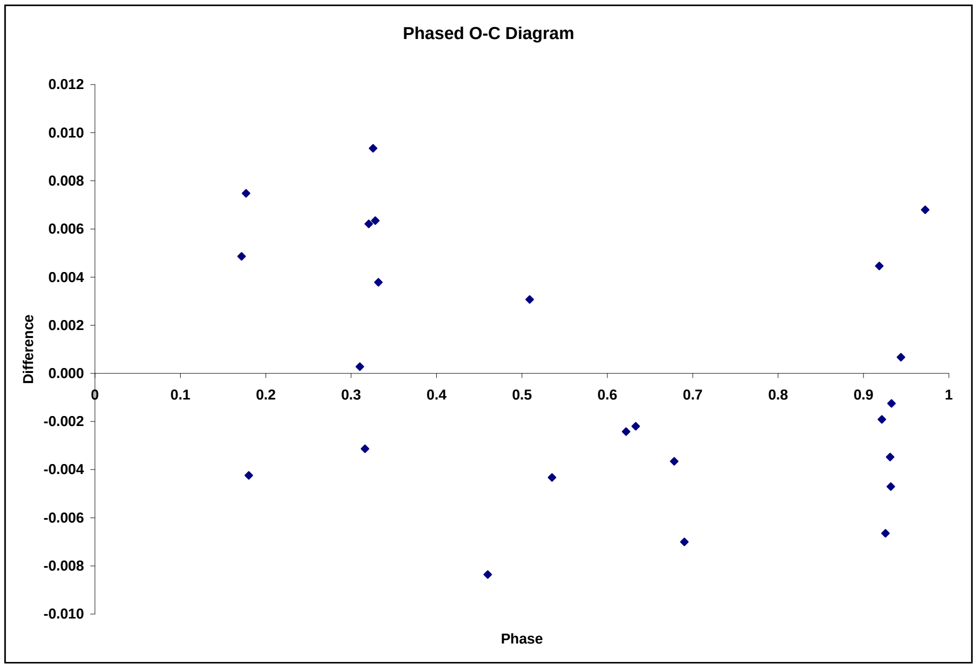


Figure 3.10: This graph shows the difference between the observed and the calculated maxima after it has been phased using the binary period.

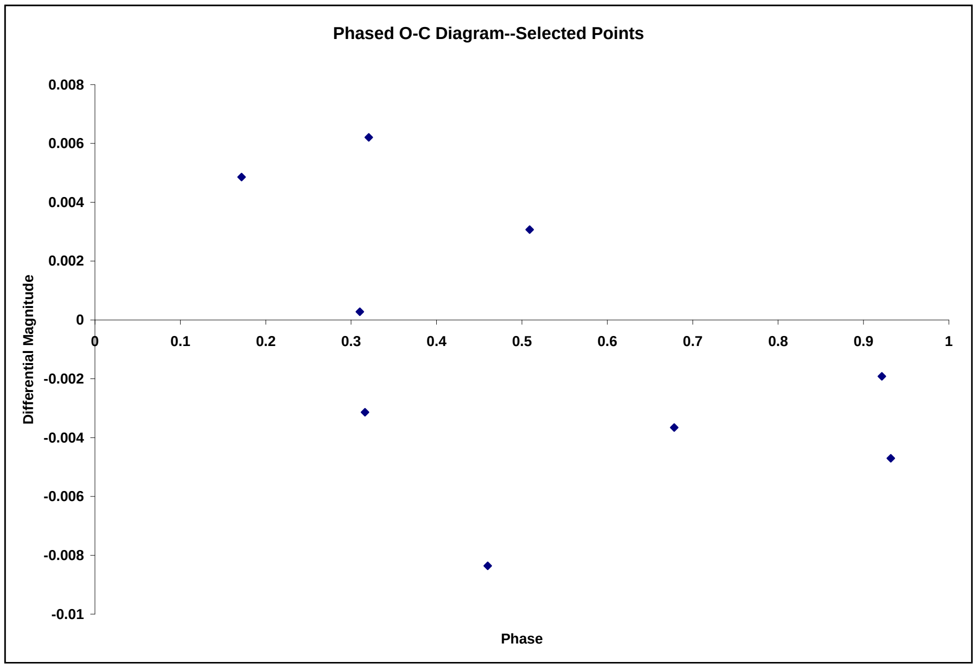


Figure 3.11: This graph shows the difference between the observed and the calculated maxima after it has been phased using the binary period. The points plotted are those maxima calculated with a large number of data points in the linear fit and with an R^2 value of 0.8 or higher.

Chapter 4

Conclusions

From the binary phased data, we are able to confirm that the period of 6.079084 given by Volkov et al. (1990) is accurate. In Fig. 3.3, the primary eclipse consists of four nights of data from two different years, 2001 and 2006, and two months within 2006, July and August. The overlap of each separate light curve is almost exact, the slight difference probably due only to variations caused by the δ Scuti component. The secondary eclipse consists of three nights from three different years, 2001, 2006, and 2008. The overlap of the secondary eclipse is also almost exact. Based on the accuracy of this period, we also conclude that the orbit of V577 Oph is stable. The stars are not slowly approaching one another and causing the period to decrease over time.

We also confirm that V577 Oph has a δ Scuti variable as its primary component, based on the shape of the light curve during an eclipse. The primary eclipse light curve contains variations due to the δ Scuti, whereas the secondary eclipse does not. We measured a semi-amplitude of 0.0223 mag in the V filter, which is similar to the published value of 0.0289 mag.

From the pulsator phased data we are able to confirm that the published period for the δ Scuti corresponds to the primary frequency of the star's pulsation, though we suspect the presence of multiple frequencies. Based on the analysis of Period04, we have considerable evidence of this. We suspect that one of these minor frequencies is due to the orbit of the δ Scuti component around its companion. We believe the slight sinusoidal trend in the phased O-C diagram is evidence of this.

References

Diethelm, R., Information Bulletin on Variable Stars (IBVS), 1993, 3894, 1

General Catalogue of Variable Stars 2004,
(ftp://cdsarc.u-strasbg.fr/pub/cats/II/250/vartype.txt)

Hintz, E. G. and Joner, M. D. and McNamara, D. H. and Nelson, K. A. and Moody, J. W. and Kim, C.,PASP, 1997, 109, 15-20

Lenz P., Breger M. 2005, CoAst, 146, 53

Volkov, I. M., Information Bulletin on Variable Stars (IBVS), 1990, 3493, 1

Zhou, A. Y., Information Bulletin on Variable Stars (IBVS), 2001, 5087, 1

Appendix A

Scripts

This appendix contains the scripts used in reduction and photometry.

A.1 Data Reduction Scripts

A.1.1 Header Modification

The following two scripts edit the headers of both the image frames and the calibration frames. The first edits the headers of the image frames and the second edits the calibration frames.

tdateobhead.cl

```
#tdateobhead.cl modified slightly from Joner’s ten1headfix.cl by Paul Iverson,
altered by Alie Porter

    hedit v577*.fits OBSERVAT “esc” add+ ver-
    hedit v577v*.fits SUBSET “V” add+ ver-
    hedit v577b*.fits SUBSET “B” add+ ver-
    hedit v577r*.fits SUBSET “R” add+ ver-
    hedit v577*.fits IMAGETYP object ver-

    hedit v577*.fits OBSERVER “BYU-Natalie Porter” addonly- ver-

    hedit *v577*.fits* RA “12:06:0.82” add+ ver-
    hedit *v577*.fits* RA “12:06:0.82” add+ ver-
    hedit *v577*.fits* DEC “23:12:16.77” add+ ver-
```

```
        hedit *v577*.fits* DEC “+23:12:16.77” add+ ver-

        hedit v577*.fits EPOCH ‘2000.0’ add+ ver-
        hselect v577*.fits $I,DATE-OBS yes > datalist1
        !sed “s/[0-9-]*[T]//g” datalist1 > datalist2
        list = “datalist2”
        while (fscan (list, s1, s2) != EOF)
        hedit (s1, “UT”, s2, add+, ver-)
        del datalist1
        del datalist2

        !echo “st = mst(@‘DATE-OBS’, UT, obsdb (observat, \“longitude\”))” >
st.cmds
        asthedit v577*.fits st.cmds table=“” verbose+
        del st.cmds

        setairmass v577*.fits
        setjd v577*.fits
```

tcalhead.cl

```
        #tcalhead.cl by Paul Iverson, edited by Alie Porter

        #hedit flat*.fits SUBSET “V” ver-
        hedit flatv*.fits SUBSET “V” ver-
        hedit flatb*.fits SUBSET “B” ver-
        hedit flatr*.fits SUBSET “R” ver-

        hedit zero*.fits IMAGETYP zero ver-
        hedit dark*.fits IMAGETYP dark ver-
```

```
hedit flat*.fits IMAGETYP flat ver-

hedit zero*.fits OBJECT Zero ver-
hedit dark*.fits OBJECT Dark ver-

hedit flatv*.fits OBJECT "Flat V" ver-
hedit flatb*.fits OBJECT "Flat B" ver-
hedit flatr*.fits OBJECT "Flat R" ver-

hedit flatv*.fits FILTER "V" ver-
hedit flatb*.fits FILTER "B" ver-
hedit flatr*.fits FILTER "R" ver-
```

A.1.2 Image Reduction

The following script applies the calibration frames to the image frames as detailed in Sec. 2.2.

tproc.cl

```
#tproc.cl by Paul Iverson, altered by Alie Porter

print ("Combining zeros")
zerocombine zero*.fits
beep
beep
disp Zero.fits

ccdproc.ccdtype = "dark"
ccdproc.zero = "Zero.fits"
```

```
print ("Correcting dark frames")
ccdproc dark*.fits zerocor+ darkcor- flatcor-

print ("Combining darks")
darkcombine dark*.fits

beep
beep
disp Dark.fits

ccdproc.ccdtype = "flat"
ccdproc.zero = "Zero.fits"
ccdproc.dark = "Dark.fits"

print ("Correcting flat frames")
ccdproc flat*.fits zerocor+ darkcor+ flatcor-

print ("Combining flats")
flatcombine flat*.fits subsets+

beep
beep
disp Flat.fits
#disp FlatV.fits
#disp FlatB.fits
#disp FlatR.fits

ccdproc.ccdtype = "object"
ccdproc.zero = "Zero.fits"
ccdproc.dark = "Dark.fits"
ccdproc.flat = "Flat*.fits"

print ("Correcting images frames")
ccdproc v577*.fits zerocor+ darkcor+ flatcor+

beep
beep
disp v577*01.fits
#disp v577v*01.fits
#disp v577b*01.fits
#disp v577r*01.fits
```

A.2 Differential Photometry Scripts

A.2.1 Removing Bad Frames

This script displays each individual frame and then gives the option of keeping or deleting that frame.

alldisp.cl

```
#display all the frames in a folder, and deletes the bad ones.

#variables string s1 = ""
int tmp1
struct *list1

sections ("v577*.fits",>"imagesfile")

list1 = "imagesfile"
```

```
while (fscan(list1,s1) != EOF){
print ("Image:",s1)
disp (s1,1)
print ("Do you want to keep this image? 0 if no, 1 if yes)")
scanf ("%d",tmp1)
if (tmp1 != 0 && tmp1 != 1){
print ("Try again!")
}
;
if (tmp1 == 0){
del (s1)
tmp = 1
}
;
}

del imagesfile
```

A.2.2 Preparation for Aligning Frames

The following scripts are used to calculate various parameters of each individual frame, which values are then put into the headers to be used by the aligning script. The first script collects the gain and read noise for several sections of each frame, which is then averaged in the second script and the third puts the value into the headers. The fourth script calculates the sky value and FWHM for each frame and puts the values into the headers.

gainrdnoisecol.cl

```
#ap_gain_rdnoise_collector_v_1.cl by Paul Iverson altered by Alie Porter
```

```
procedure gainrdnoisecol (fnum,bnum,pick)

string fnum {prompt = “Enter the number of flats frames to use
(minimum = 2)”}
string bnum {prompt = “Enter the number of bias frames to use
(minimum = 2)”}
string pick {prompt = “Enter the filter to collect (B,V,R or All)”,
enum=“B|V|R|All”}

begin
string flatnum
string biasnum
string anspick
flatnum = fnum
while (int(flatnum) < 2){
clear
flatnum = fnum
}
biasnum = bnum
while (int(biasnum) < 2){
clear
biasnum = bnum
}
anspick = pick
end

#variable declarations
int n=1
int f1, f2,fn
int b1,b2,bn
int xlen, ylen
int x1, x2, x3, x4, x5, x6
int y1, y2, y3, y4, y5, y6
int numflatB, numflatV, numflatR
int numbias
string s1=“”
string flat, bias
string flat1, flat2
string bias1, bias2

fn = int(flatnum)
bn = int(biasnum)

hselect(“*.fits”, “$I,FILTER”, yes, >>“datalistfile”)
sections @datalistfile
if (sections.nimages == 0){
hselect (“*.FIT”, “$I,FILTER”, yes, >>“datalistfile”)
sections @datalistfile
if (sections.nimages == 0){
hselect (“*.fits”,“$I,FILTER”,yes,>>“datalistfile”)
}
;
}
;

match (“flat”,“datalistfile”,>>“datalistflat”)
sections @datalistflat
if (sections.nimages == 0){
```

```
match (“FLAT”,“datalistfile”,>>“datalistflat”)
}
;

match (“B$”,“datalistflat”,>>“datalistflatBtemp”)
sections @datalistflatBtemp
numflatB = sections.nimages
if (numflatB > 0){
while (n <= numflatB && n <= fn){
tabpar (“datalistflatBtemp”,1,n)
flat = tabpar.value
print (flat, >>“datalistflatB”)
n = n+1
}
numflatB = n - 1
}
;
n=1

match (“V$”,“datalistflat”,>>“datalistflatVtemp”)
sections @datalistflatVtemp
numflatV = sections.nimages
if (numflatV > 0){
while (n <= numflatV && n <= fn){
tabpar (“datalistflatVtemp”,1,n)
flat = tabpar.value
print (flat, >>“datalistflatV”)
n = n+1
}
```

```
numflatV = n - 1
}
;
n=1

match ("R$","datalistflat",>>"datalistflatRtemp")
sections @datalistflatRtemp
numflatR = sections.nimages
if (numflatR > 0){
while (n <= numflatR && n <= fn){
tabpar ("datalistflatRtemp",1,n)
flat = tabpar.value
print (flat, >>"datalistflatR")
n = n+1
}
numflatR = n - 1
}
;
n=1

match ("zero","datalistfile",>>"datalistbiastemp")
sections @datalistbiastemp
if (sections.nimages == 0){
match ("BIAS","datalistfile",>>"datalistbiastemp")
sections @datalistbiastemp
if (sections.nimages == 0){
match ("bias","datalistfile",>>"datalistbiastemp")
}
;
```

```
}
;

sections @datalistbiastemp
numbias = sections.nimages
if (numbias > 0){
while (n <= numbias && n <= bn){
tabpar ("datalistbiastemp",1,n)
bias = tabpar.value
print (bias, >>"datalistbias")
n = n+1
}
numbias = n - 1
}
;
n=1
del *temp
clear

if (anspick == "B"){
numflatV = 0
numflatR = 0
}
;
if (anspick == "V"){
numflatB = 0
numflatR = 0
}
;
```

```
if (anspick == “R”){
numflatB = 0
numflatV = 0
}
;

print (“Using ”,numflatB,“ flat B frames”)
print (“Using ”,numflatV,“ flat V frames”)
print (“Using ”,numflatR,“ flat R frames”)
print (“Using ”,numbias,“ bias frames”)

sleep (2)

if (numflatB > 0) {
for (f1=1;f1<=numflatB-1;f1+=1){
for (f2=f1+1;f2<=numflatB;f2+=1){
for (b1=1;b1<=numbias-1;b1+=1){
for (b2=b1+1;b2<=numbias;b2+=1){
tabpar (“datalistflatB”,1,f1)
flat1 = tabpar.value

imgets.param.p_mode = “h”

imgets (flat1, param=“i_naxis1”)
xlen = int(imgets.value)
imgets (flat1, param=“i_naxis2”)
ylen = int(imgets.value)

x1 = 50
```

```
x2 = 150
x3 = xlen - 150
x4 = xlen - 50
if ((xlen % 2) == 0){
x5 = (xlen/2)-50
x6 = (xlen/2)+50
}
;
if ((xlen % 2) == 1){
x5 = ((xlen - 1)/2)-50
x6 = ((xlen - 1)/2)+50
}
;

y1 = 50
y2 = 150
y3 = ylen - 150
y4 = ylen - 50
if ((ylen % 2) == 0){
y5 = (ylen/2)-50
y6 = (ylen/2)+50
}
;
if ((ylen % 2) == 1){
y5 = ((ylen - 1)/2)-50
y6 = ((ylen - 1)/2)+50
}
;
```

```
    tabpar (“datalistflatB”,1,f2)
    flat2 = tabpar.value
    tabpar (“datalistbias”,1,b1)
    bias1 = tabpar.value
    tabpar (“datalistbias”,1,b2)
    bias2 = tabpar.value

    clear

    print (“There are ”,numflatB,“ flat B frames”)
    print (“There are ”,numbias,“ bias frames”)
    print (“”)
    print (“Flat 1: ”,flat1)
    print (“Flat 2: ”,flat2)
    print (“Bias 1: ”,bias1)
    print (“Bias 2: ”,bias2)
    ap_findgain (flat1,flat2,bias1,bias2,section=“[“//x1//”:“//x2//”,“//y1
//”:“//y2//”]”,verbose-,>>“gainrdnoiseB.info”)
    ap_findgain (flat1,flat2,bias1,bias2,section=“[“//x1//”:“//x2//”,“//y3
//”:“//y4//”]”,verbose-,>>“gainrdnoiseB.info”)
    ap_findgain (flat1,flat2,bias1,bias2,section=“[“//x3//”:“//x4//”,“//y1
//”:“//y2//”]”,verbose-,>>“gainrdnoiseB.info”)
    ap_findgain (flat1,flat2,bias1,bias2,section=“[“//x3//”:“//x4//”,“//y3
//”:“//y4//”]”,verbose-,>>“gainrdnoiseB.info”)
    ap_findgain (flat1,flat2,bias1,bias2,section=“[“//x5//”:“//x6//”,“//y5
//”:“//y6//”]”,verbose-,>>“gainrdnoiseB.info”)
    }
    }
    }
```

```
}
}
;

if (numflatV > 0) {
for (f1=1;f1<=numflatV-1;f1+=1){
for (f2=f1+1;f2<=numflatV;f2+=1){
for (b1=1;b1<=numbias-1;b1+=1){
for (b2=b1+1;b2<=numbias;b2+=1){
tabpar ("datalistflatV",1,f1)
flat1 = tabpar.value

imgets.param.p_mode = "h"

imgets (flat1, param="i_naxis1")
xlen = int(imgets.value)
imgets (flat1, param="i_naxis2")
ylen = int(imgets.value)

x1 = 50
x2 = 150
x3 = xlen - 150
x4 = xlen - 50
if ((xlen % 2) == 0){
x5 = (xlen/2)-50
x6 = (xlen/2)+50
}
;
if ((xlen % 2) == 1){
```

```
x5 = ((xlen - 1)/2)-50
x6 = ((xlen - 1)/2)+50
}
;

y1 = 50
y2 = 150
y3 = ylen - 150
y4 = ylen - 50
if ((ylen % 2) == 0){
y5 = (ylen/2)-50
y6 = (ylen/2)+50
}
;
if ((ylen % 2) == 1){
y5 = ((ylen - 1)/2)-50
y6 = ((ylen - 1)/2)+50
}
;

tabpar ("datalistflatV",1,f2)
flat2 = tabpar.value
tabpar ("datalistbias",1,b1)
bias1 = tabpar.value
tabpar ("datalistbias",1,b2)
bias2 = tabpar.value

clear
```

```
        print ("There are ",numflatV," flat V frames")
        print ("There are ",numbias," bias frames")
        print ("")
        print ("Flat 1: ",flat1)
        print ("Flat 2: ",flat2)
        print ("Bias 1: ",bias1)
        print ("Bias 2: ",bias2)
        ap_findgain (flat1,flat2,bias1,bias2,section="["//x1//":"//x2//","//y1
//":"//y2//"]",verbose-,>>"gainrdnoiseV.info")
        ap_findgain (flat1,flat2,bias1,bias2,section="["//x1//":"//x2//","//y3
//":"//y4//"]",verbose-,>>"gainrdnoiseV.info")
        ap_findgain (flat1,flat2,bias1,bias2,section="["//x3//":"//x4//","//y1
//":"//y2//"]",verbose-,>>"gainrdnoiseV.info")
        ap_findgain (flat1,flat2,bias1,bias2,section="["//x3//":"//x4//","//y3
//":"//y4//"]",verbose-,>>"gainrdnoiseV.info")
        ap_findgain (flat1,flat2,bias1,bias2,section="["//x5//":"//x6//","//y5
//":"//y6//"]",verbose-,>>"gainrdnoiseV.info")
        }
        }
        }
        }
        }
        ;

        if (numflatR > 0) {
        for (f1=1;f1<=numflatR-1;f1+=1){
        for (f2=f1+1;f2<=numflatR;f2+=1){
        for (b1=1;b1<=numbias-1;b1+=1){
        for (b2=b1+1;b2<=numbias;b2+=1){
```

```
tabpar (“datalistflatR”,1,f1)
flat1 = tabpar.value

imgets.param.p_mode = “h”

imgets (flat1, param=“i_naxis1”)
xlen = int(imgets.value)
imgets (flat1, param=“i_naxis2”)
ylen = int(imgets.value)

x1 = 50
x2 = 150
x3 = xlen - 150
x4 = xlen - 50
if ((xlen % 2) == 0){
x5 = (xlen/2)-50
x6 = (xlen/2)+50
}
;
if ((xlen % 2) == 1){
x5 = ((xlen - 1)/2)-50
x6 = ((xlen - 1)/2)+50
}
;

y1 = 50
y2 = 150
y3 = ylen - 150
y4 = ylen - 50
```

```
if ((ylen % 2) == 0){
y5 = (ylen/2)-50
y6 = (ylen/2)+50
}
;
if ((ylen % 2) == 1){
y5 = ((ylen - 1)/2)-50
y6 = ((ylen - 1)/2)+50
}
;

tabpar (“datalistflatR”,1,f2)
flat2 = tabpar.value
tabpar (“datalistbias”,1,b1)
bias1 = tabpar.value
tabpar (“datalistbias”,1,b2)
bias2 = tabpar.value

clear

print (“There are ”,numflatR,“ flat R frames”)
print (“There are ”,numbias,“ bias frames”)
print (“”)
print (“Flat 1: ”,flat1)
print (“Flat 2: ”,flat2)
print (“Bias 1: ”,bias1)
print (“Bias 2: ”,bias2)
ap_findgain (flat1,flat2,bias1,bias2,section=“[“//x1//”:“//x2//”,“//y1
//”:“//y2//”]”,verbose-,>>“gainrdnoiseR.info”)
```

```
    ap_findgain (flat1,flat2,bias1,bias2,section="["//x1//":"//x2//","//y3
//":"//y4//"]",verbose-,>>"gainrdnoiseR.info")
    ap_findgain (flat1,flat2,bias1,bias2,section="["//x3//":"//x4//","//y1
//":"//y2//"]",verbose-,>>"gainrdnoiseR.info")
    ap_findgain (flat1,flat2,bias1,bias2,section="["//x3//":"//x4//","//y3
//":"//y4//"]",verbose-,>>"gainrdnoiseR.info")
    ap_findgain (flat1,flat2,bias1,bias2,section="["//x5//":"//x6//","//y5
//":"//y6//"]",verbose-,>>"gainrdnoiseR.info")
    }
    }
    }
    }
    }
    ;

    del data*

    beep
    beep
```

gainrdnoisecalc.cl

```
    #apgainrdnoisecalculator_v_1.cl by Paul Iverson altered by Alie Porter

    #variable declarations
    string s1 = ""
    string s2 = ""
    struct *list1
    struct *list2
```

```
sections (“*.info”,>“datalistfile”)

list1 = “datalistfile”
while (fscan(list1,s1) != EOF){
copy (s1,s1//“.temp”)
}
del data*

if (access(“gainrdnoiseB.info.temp”)){
list1 = “gainrdnoiseB.info.temp”
while (fscan(list1,s1,s2) != EOF){
clear
print (“Calculating gain and rdnoise for the B filter”)
print (s1, >> “gainB.info.temp”)
print (s2, >> “rdnoiseB.info.temp”)
}

type “gainB.info.temp” | average > “gainB.info”
type “rdnoiseB.info.temp” | average > “rdnoiseB.info”
} ;
if (access(“gainrdnoiseV.info.temp”)){
list1 = “gainrdnoiseV.info.temp”
while (fscan(list1,s1,s2) != EOF){
clear
print (“Calculating gain and rdnoise for the V filter”)
print (s1, >> “gainV.info.temp”)
print (s2, >> “rdnoiseV.info.temp”)
}
type “gainV.info.temp” | average > “gainV.info”
type “rdnoiseV.info.temp” | average > “rdnoiseV.info”
}
;

if (access(“gainrdnoiseR.info.temp”)){
list1 = “gainrdnoiseR.info.temp”
while (fscan(list1,s1,s2) != EOF){
clear
print (“Calculating gain and rdnoise for the R filter”)
print (s1, >> “gainR.info.temp”)
print (s2, >> “rdnoiseR.info.temp”)
}

type “gainR.info.temp” | average > “gainR.info”
type “rdnoiseR.info.temp” | average > “rdnoiseR.info”
}
;

del *temp

beep
beep
```

gainrdnoiseobhead.cl

```
#ap_gain_rdnoise_obhead_v_1.cl by Paul Iverson altered by Alie Porter
#purpose of this script is to edit headers adding RDNoise and Gain;
values for RDnoise and Gain taken from Cody Short’s thesis
```

```
#variable declarations
real Bgain, Brdnoise
real Vgain, Vrdnoise
real Rgain, Rrdnoise
string s1
string s2
struct *list1

#collects gain and rdnoise values
if (access(“gainB.info”)){
tabpar (“gainB.info”,1,1)
Bgain = real(tabpar.value)
}
;
if (access(“rdnoiseB.info”)){
tabpar (“rdnoiseB.info”,1,1)
Brdnoise = real(tabpar.value)
}
;
if (access(“gainV.info”)){
tabpar (“gainV.info”,1,1)
Vgain = real(tabpar.value)
}
;
if (access(“rdnoiseV.info”)){
tabpar (“rdnoiseV.info”,1,1)
Vrdnoise = real(tabpar.value)
}
;
```

```
if (access("gainR.info")){
tabpar ("gainR.info",1,1)
Rgain = real(tabpar.value)
}
;
if (access("rdnoiseR.info")){
tabpar ("rdnoiseR.info",1,1)
Rrdnoise = real(tabpar.value)
}
;

#edits header fields "READNOISE" and "GAIN"
hselect ("v577*.fits","$I,SUBSET",yes,>>"datalist1")
hselect ("flat*.fits","$I,SUBSET",yes,>>"datalist1")
match "B$" "datalist1" > "datalistB"
match "V$" "datalist1" > "datalistV"
match "R$" "datalist1" > "datalistR"

if (access("datalistB")){
if (access("gainB.info")){
list1 = "datalistB"
while (fscan(list1,s1,s2) != EOF){
hedit (s1, "RDNOISE", Brdnoise,add+,addonly+,ver-)
hedit (s1, "GAIN", Bgain,add+,addonly+,ver-)
}
}
;
}
;
```

```
if (access("datalistV")){
if (access("gainV.info")){
list1 = "datalistV"
while (fscan(list1,s1,s2) != EOF){
hedit (s1, "RDNOISE", Vrdnoise,add+,addonly+,ver-)
hedit (s1, "GAIN", Vgain,add+,addonly+,ver-)
}
}
;
}
;

if (access("datalistR")){
if (access("gainR.info")){
list1 = "datalistR"
while (fscan(list1,s1,s2) != EOF){
hedit (s1, "RDNOISE", Rrdnoise,add+,addonly+,ver-)
hedit (s1, "GAIN", Rgain,add+,addonly+,ver-)
}
}
;
}
;

del data*

beep
beep
```

fwhm.cl

```
#ap_fwhm_obhead_v_3.cl by Paul Iverson altered by Alie Porter
#the purpose of this cl script is to compute and add FWHM to the
header using the psfmeasure tool in IRAF

#variable declarations
int i
int n = 1
int xlen
int ylen
int x1, x2
int y1, y2
real num, avg, dev
real skyvalue, standdev
real FWHMcheck
string files, trimfiles, dimfiles
string temp
string s1 = ""
string s2 = ""
string s3 = ""
string FWHM
struct *list1
struct *list2

sections ("v577*.fits", opt="fullname", > "datalist1")
tabpar ("datalist1",1,1)
dimfile = tabpar.value
```

```
imgets.param.p_mode = "h"

imgets (dimfile, param="i_naxis1")
xlen = int(imgets.value)
imgets (dimfile, param="i_naxis2")
ylen = int(imgets.value)

x1 = 15
x2 = xlen - 15
y1 = 15
y2 = ylen - 15

imcopy ("v577-*.fits"//"["//x1//":"//x2//","//y1//":"//y2//"]",
"v577-*//.trim")

sections ("v577*.trim*", opt="fullname", > "datalist2")

joinlines ("datalist1,datalist2",> "datalistfilenames")

list1 = "datalistfilenames"
while (fscan(list1,s1,s2) != EOF){
files = (s1)
trimfiles = (s2)

if (access("fwhm.reg") == yes || access("sky.reg") == yes){
del fwhm.reg
del sky.reg
}
```

```
;

findpars.threshold = 10
datapars.sigma = 50
datapars.datamin = INDEF
datapars.datamax = INDEF

findpars.threshold.p_mode = "h"
datapars.fwhmpsf.p_mode = "h"
datapars.sigma.p_mode = "h"
datapars.datamin.p_mode = "h"
datapars.datamax.p_mode = "h"

daofind (trimfiles,output="skydao.reg",verif-)
txdump ("skydao.reg","xcenter,ycenter",yes,> "sky.reg")
del skydao.reg

# display (trimfiles, 1)
# tvmark (1, coords="sky.reg")

phot.coords.p_mode = "h"
phot.output.p_mode = "h"

phot (trimfiles,coords="sky.reg",interac-,verif-)

clear

hedit (files,"FWHM","INDEF",add+,ver-)
hedit (trimfiles,“FWHM”,“INDEF”,add+,ver-)

if (access(trimfiles//“.mag.1”)){
txdump (trimfiles//“.mag.1”,“msky”,yes,>>“msky.info”)
txdump (trimfiles//“.mag.1”,“stdev”,yes,>>“stde.info”)

type “msky.info” | average > “sky.info”
type “stde.info” | average > “stdev.info”

tabpar (“sky.info”,1,1)
skyvalue = real(tabpar.value)

hedit (files,“SKYAVG”,skyvalue,add+,addonly+,ver-)
hedit (trimfiles,“SKYAVG”,skyvalue,add+,addonly+,ver-)

tabpar (“stdev.info”,1,1)
standdev = real(tabpar.value)

hedit (files,“SKYDEV”,standdev,add+,addonly+,ver-)
hedit (trimfiles,“SKYDEV”,standdev,add+,addonly+,ver-)

del *sky*.info
del *stde*.info

findpars.threshold = (skyvalue/standdev) + 2
datapars.sigma = standdev
datapars.datamin = INDEF
datapars.datamax = INDEF
```

```
        daofind (trimfiles,output="fwhmdao.reg",verif-)
        txdump ("fwhmdao.reg","xcenter,ycenter",yes,> "fwhm.reg")
        del fwhmdao.reg

        # display (trimfiles, 1)
        # tvmark (1, coords="fwhm.reg")
        print ("q",>>"nomoreqs")
        psfmeasure (trimfiles,display-,imagecur="fwhm.reg",graphcur=
"nomoreqs",> (trimfiles)//".fwhm")
        del nomoreqs

        copy ((trimfiles)//".fwhm","datafwhmA")
        sed -e '/NOAO/d' -e '/Image/d' -e '/Average/d' -e 's/" "/d/g'
-e 's/obj-.........../ddddddddddddddd/g' "datafwhmA" >> "datafwhmB"

        sections @datafwhmB
        clear

        list2 = "datafwhmB"
        while (fscan(list2,s3) != EOF){
        temp = (s3)
        i = strlen(temp)
        if (n !=1 && n <=sections.nimages+1){
        temp = substr(temp,41,i-15)
        print(temp,>>"datafwhmC")
        }
        ;
        n = n +1
        }
```

```
n = 1

sed -e ‘s/d//g’ “datafwhmC” >> “datafwhmD”

type “datafwhmD” | average > “datafwhmE”

tabpar (“datafwhmE”,1,1)
avg = real(tabpar.value)

list2 = “datafwhmD”
while (fscan(list2,s3) != EOF){
num = real(s3)
dev = 1*avg
if (num >= avg && num <= avg + dev){
print (num, >>“datafwhmF”)
}
;
if (num <= avg && num >= avg - dev){
print (num, >>“datafwhmF”)
}
;
}

type “datafwhmF” | average > “datafwhmG”

tabpar (“datafwhmG”,1,1)
avg = real(tabpar.value)

list2 = “datafwhmF”
```

```
while (fscan(list2,s3) != EOF){
num = real(s3)
dev = 1*avg
if (num >= avg && num <= avg + dev){
print (num, >>"datafwhmH")
}
;
if (num <= avg && num >= avg - dev){
print (num, >>"datafwhmH")
}
;
}

type "datafwhmH" | average > "datafwhmI"

tabpar ("datafwhmI",1,1)
avg = real(tabpar.value)

list2 = "datafwhmH"
while (fscan(list2,s3) != EOF){
num = real(s3)
dev = 0.66*avg
if (num >= avg && num <= avg + dev){
print (num, >>"datafwhmJ")
}
;
if (num <= avg && num >= avg - dev){
print (num, >>"datafwhmJ")
}
```

```
;
}

type “datafwhmJ” | average > “datafwhmK”

tabpar (“datafwhmK”,1,1)
avg = real(tabpar.value)

list2 = “datafwhmJ”
while (fscan(list2,s3) != EOF){
num = real(s3)
dev = 0.33*avg
if (num >= avg && num <= avg + dev){
print (num, >>“datafwhmL”)
}
;
if (num <= avg && num >= avg - dev){
print (num, >>“datafwhmL”)
}
;
}

type “datafwhmL” | average > “datafwhmM”

tabpar (“datafwhmM”,1,1)
temp = tabpar.value
i = strlen(temp)
if (i > 6){
avg = real(substr(temp,1,6))
```

```
}
;
if (i <= 6){
avg = real(temp)
}
;

hedit ((files),“FWHM”,avg,add+,addonly+,ver-)
hedit ((trimfiles),“FWHM”,avg,add+,addonly+,ver-)

del data*
}
;
}

#extracts filename and fwhm value from object frames
hselect (“v577*.fits”,“$I,FWHM”,yes,>>“datatotal”)

fields (“datatotal”,1,>“datafiles”)
fields (“datatotal”,2,>“dataFWHM”)

#calculates non-INDEF-included fwhm
match (“INDEF”,“dataFWHM”,stop+,>“datanoINDEFFWHM”)
type “datanoINDEFFWHM” | average > “dataaverageFWHM”
tabpar (“datanoINDEFFWHM”,1,1)
i=strlen(tabpar.value)
if (i>6){
avg=real(substr(tabpar.value,1,6))
}
```

```
;

#removes INDEFs from fwhm values and replaces them with
group average
list1="dataFWHM"
while (fscan(list1,s1)!=EOF){
FWHM=(s1)
FWHMcheck=5*avg

if (FWHM!="INDEF"){
if(real(FWHM)>FWHMcheck){
FWHM=avg
}
;
print (FWHM,>>"datacorrected")
}
;
if (FWHM=="INDEF"){
FWHM=avg
print (FWHM,>>"datacorrected")
}
;
}

joinlines ("datafiles,datacorrected",>> "datacorrtotal")

list1="datacorrtotal"
while (fscan(list1,s1,s2) !=EOF){
files=(s1)
avg=real(s2)

#edits header field “FWHM”
hedit (files, “FWHM”,avg,add+,ver-)
print (files//“ ”//avg, >> “avgFWHM.info”)
}

del *mag*
del *fwhm*
del sky*
del data*

beep
beep
```

A.2.3 Aligning the Frames

The first script uses the information put into the headers by the previous four scripts. It then aligns the frames so that the stars do not appear to move from frame to frame. The second and third scripts are used in the aligning script.

simplalign.cl

```
#ap_align.cl by Paul Iverson, altered by Alie Porter
#simplifies call to Eran Ofek’s autoalign.cl

string s1
string s2
string s3
string s4
```

```
        #autoalign (“ilist”, “prefix”, fwhm, readnoise, gain, xytol, objectn,
“bug_log”)
        hselect (“v577*trim.fits”,“$I,FWHM,RDNOISE,GAIN,SKYDEV”,yes, >
“datalist1”)
        fields (“datalist1”,1,> “datalistfiles”)
        fields (“datalist1”,2,> “datalist2a”)
        fields (“datalist1”,3,> “datalist3a”)
        fields (“datalist1”,4,> “datalist4a”)
        type “datalist2a” | average >> “datalist2b”
        type “datalist3a” | average >> “datalist3b”
        type “datalist4a” | average >> “datalist4b”
        fields (“datalist2b”,1,> “datalistFWHM”)
        fields (“datalist3b”,1,> “datalistRDNOISE”)
        fields (“datalist4b”,1,> “datalistGAIN”)

        joinlines (“datalistFWHM,datalistRDNOISE,datalistGAIN”, >>
“datatotal”)

        list = “datatotal”
        while (fscan(list,s1,s2,s3) != EOF){
        nautoalign (“datalistfiles”,“a-”,real(s1),real(s2),real(s3),1.5,50,
“bug_log”)
        }

        del data*
        del input_shift
        del tmp*
        del v577*.fits.*afnl*
        del v577*.fits.*coo*
del v577*.fits.*smag*
del v577*.fits.*rmag*
del bug_log

beep
beep
```

nautoalign.cl

```
procedure nautoalign (ilist, prefix, fwhm, readnoise, gain, xytol, objectn,
bug_log, succeed)
#————————————————————————————————————
# autoalign.cl -
# Documentation
# ————-
# the list:
# #field name, and list of images in the following lines, etc.
# Example:
# #GRB990316
# 990316.015
# 990316.016
# 990316.017
# #AD Leo
# 990316.018
# 990316.019
# .
# .
# .
# INSTALL: edit and add the following lines to the login.cl
# task xyshift =/home/wise-cdr/eran/iraf/bin/xyshift
# task autodaofind = /home/wise-cdr/eran/iraf/script/autodaofind.cl
# Written By Eran Ofek, October 1998, Last update: 061098
#——————————————————————————————

string ilist {“”,prompt=“list of images to align”}
string prefix {“a”,prompt=“prefix for shifted output images”}
real fwhm {1.5,prompt=“PSF FWHM in pixels”}
real readnoise {29.0609,prompt=“CCD read out noise in electrons”}
real gain {4.0364,prompt=“CCD gain in electrons per count”}
real xytol {1.5, min=0.0,prompt=“matching tolerance for pgshift”}
int objectn {50, prompt=“Max. Number of stars to match”}
string bug_log {“buglog”,prompt=“logfile name”}
bool succeed {no,prompt=“succeeded to find astrometric solution”}

struct *lis1
struct *lis2
struct *lis3
struct *lis4

begin

string imname
string refimage
string magfile real avshiftx # shift in X axis.
real avshifty # shift in Y axis.
real pershift # number of stars used to shift the image.
bool last_ast
int match_n # number of stars matched
```

```
if (access("rdnoiseR.info")){
delete (bug_log,verify=no,>>&"/dev/null")
}
;

lis1 = ilist
while (fscan(lis1, imname)!= EOF)
{
nautodaofind(imname=imname,out_file="default",fwhm=fwhm,
readnoise=readnoise,gain=gain,threshold_sig=50.0)
}

end

# find shifts between images
lis3 = ilist
last_ast = yes
while (fscan(lis3, imname)!=EOF)
{
print ('————————————————')
print (' Field Line : ',imname)
print ('————————————————')

if (substr(imname,1,1) == '#')
{
# next field
last_ast = yes
}
else
```

```
{
if (last_ast==yes)
{
last_ast = no
# set image to be reference image
refimage = imname
print ('=====================================')
print ('Reference Image : ', refimage)
print ('=====================================')
magfile = imname // '.coo.1'
#——————————————————-
# Prepare the image list file for pgshift
#——————————————————-
print("Prepare catalog for findshift")
delete ('tmp_object',verify=no,>>&"/dev/null")
print(imname, > 'tmp_object')

#creating the .smag file
delete (imname//'.smag.1',verify=no,>>&"/dev/null")
txdump(textfile=magfile,fields="ID,XCENTER,YCENTER,MAG,
MERR,MSKY,NITER,SHARPNESS,CHI",expr="MAG[1] !=INDEF",headers=yes,
>>imname//'.smag.1')

delete (imname//'.rmag.1',verify=no,>>&"/dev/null")
txdump(textfile=magfile,fields="ID,XCENTER,YCENTER,MAG,
MERR,MSKY,NITER,SHARPNESS,CHI",expr="MAG[1] !=INDEF",headers=no,
>>imname//'.rmag.1')

print(' sorting '//imname//'.rmag.1')
```

```
        delete (imname//'.afnl.1',verify=no,>>&"/dev/null")

        # creating the .afnl file
        # sort the rls file by decreasing magnitude
        sort(input_fi=imname//'.rmag.1',column=4,numeric=yes, >>
imname//'.afnl.1')

        imcopy (input=imname, output=prefix//imname)
        }
        else
        {

        magfile = imname // '.coo.1'

        #—————————————————————————-
        # Prepare the image catalog file for pgshift
        #—————————————————————————-
        print("Prepare catalog for findshift")
        delete ('tmp_object',verify=no,>>&"/dev/null")

        print(imname, > 'tmp_object')

        #creating the .smag file
        delete (imname//'.smag.1',verify=no,>>&"/dev/null")
        txdump(textfile=magfile,fields="ID,XCENTER,YCENTER,MAG,
MERR,MSKY,NITER,SHARPNESS,CHI",expr="MAG[1] !=INDEF",headers=yes,
>>imname//'.smag.1')

        delete (imname//'.rmag.1',verify=no,>>&"/dev/null")
```

```
    txdump(textfile=magfile,fields="ID,XCENTER,YCENTER,MAG,
MERR,MSKY,NITER,SHARPNESS,CHI",expr="MAG[1] !=INDEF",headers=no,
>>imname//'.rmag.1')

    print(' sorting '//imname//'.rmag.1')
    delete (imname//'.afnl.1',verify=no,>>&"/dev/null")

    # creating the .afnl file
    # sort the rls file by decreasing magnitude
    sort(input_fi=imname//'.rmag.1',column=4,numeric=yes, >>
imname//'.afnl.1')

    #——————
    # compute shifts
    #——————
    print (' Computing Shifts for image : ',imname)
    delete ('input_shift',verify=no,>>&"/dev/null")
    print (imname//'.afnl.1', >> 'input_shift')
    print (refimage//'.afnl.1', >> 'input_shift')
    print (xytol, >> 'input_shift')
    print(imname//'.afnl.1','\n',refimage//'.afnl.1','\n',xytol,'\n',objectn) |
xyshift | scan(avshiftx, avshifty, pershift, match_n)
    print("—————- Shift in pixels ——————-")
    print(" X = ", avshiftx)
    print(" Y = ", avshifty)
    print(" % = ", pershift)
    print(" n = ", match_n)
    print('Image : ',imname,' shift% ',pershift, >> bug_log)
```

```
        # shift the image
        print (‘ ===> Shifting image : ’, imname)
        imshift(input=imname, output=prefix//imname, xshift=-avshiftx,
yshift=-avshifty)
        }
        }
        }
        succeed = yes
```

nautodaofind.cl

```
        procedure nautodaofind
        #————————————————————————————————
        # autodaofind.cl - Automatic daofind for image.
        # By : Eran O. Ofek, altered by Alie Porter
        # Written: August 1998, Last Update: Aug 10th, 1998
        #————————————————————————————————

        string imname {“”,prompt=“image name”}
        string out_file {“default”,prompt=“output file name”}
        real fwhm {3.0,prompt=“PSF FWHM in pixels”}
        real readnoise {6.50,prompt=“CCD read out noise in electrons”}
        real gain {8.42,prompt=“CCD gain in electrons per count”}
        real threshold_sig {5.0,prompt=“threshold on sigma above
background”}

        struct *lis1

        begin
```

```
real sky_noise

hselect (imname,“$I,SKYDEV”,yes, > “dataskydev”)
tabpar (“dataskydev”,2,1)
sky_noise = real(tabpar.value)

print (’Find stars using Daofind’)

delete (imname//‘.coo.*’,verify=no,>>&“/dev/null”)

daofind(image=imname,output=out_file,verify=no, verbose=yes,
sigma=sky_noise, scale=1, fwhmpsf=fwhm, readnoi=readnoise, epadu=gain,
thresho=threshold_sig)

del dataskydev

print(“END autodaofind”)

end
```

A.2.4 Obtaining Differential Magnitudes

This script uses a coordinate file to calculate the magnitude of specific stars in the frame. Those magnitudes are later converted into differential magnitudes.

anightphot4.cl

```
# NIGHTPHOT – IRAF script designed to make photing a million files a
lot easier
# without having to do each frame individually.
```

```
procedure anightphot4 (images, center_file)

string images {prompt="Root name of images to phot"}
string center_file {prompt="File of approximate centers (ds9.reg)"}
struct *list, *list2

begin
# Local variables
string imagelist
string img
string imgroot
string coordfile, tmp5
int i, end1, end2, tmp1, tmp2, tmp3, tmp4, tmp6
real temp, FWHM, temp3, intfwhm, inthwhm #temp2

# Make sure the noao and apphot or daophot packages are loaded
if (! defpac ("digiphot")) {
print ("You have not loaded the apphot or daophot package.")
print ("One of these packages must be loaded before continuing.")
bye
}
# Create a text file list of images to phot.
imagelist = mktemp ("tmp$night")
imgroot = images
sections (imgroot//"*", > imagelist)

# Open the list of images and scan through it.
list = imagelist
coordfile = center_file
```

```
#print ("What is your starting FWHM value from IMEXAM")
#scanf ("%f", intfwhm)
while (fscan (list, img) != EOF) {

# Display the current image frame in frame 1. This allows the
# frame to appear fresh without markings, etc.
display (img, 1)

# Call the tvmark command to allow the user to see if his/her stars
# are correctly centered.
tvmark (1, coords=coordfile)

# Ask the user to input whether to continue or not.
end1 = 0
while (end1 == 0){
print ("Are your stars marked and labeled correctly? (0 if no, 1 if yes)")
scanf ("%d",tmp3)
if (tmp3 != 0 && tmp3 != 1)
print ("You entered an incorrect response!")
if (tmp3 == 0){
while (tmp3 == 0) {
display (img, 1)
print ("Please re-mark the stars and save as `ds9.reg"")
print ("Once you are done please enter 1 to continue.")
scanf ("%d", tmp4)
if (tmp4 != 1)
print ("You entered an incorrect response!")
if (tmp4 == 1)
tmp3 = 1
}
}
if (tmp3 == 1){
coordfile = “ds9.reg”
end1 = 1
}
}
# PSFMeasure Command
imgets (img,param=“FWHM”)
intfwhm = real(imgets.value)
if (intfwhm == INDEF){
intfwhm = 5.5
}
if (intfwhm != INDEF) {
temp = intfwhm
temp3 = 3*intfwhm
}

# Get rid of the “.fits” extension
i = strlen (img)
if (substr (img, i-3, i) == “.fits”)
img = substr (img, 1, i-4)

# Call the phot command
fitskypars.annulus = temp3
phot (img, “”, coords=coordfile, output=“default”, verify=no,
update=yes, verbose=yes)
```

```
    # Ask the user to input whether they want to keep this
“.mag.1” file.
    end2 = 0
    while (end2 == 0){
    print (“Are you satisfied with your results for this frame (no errors)?”)
    print (“Enter 1 to continue, 0 to re-phot”)
    scanf (“%d”, tmp1)
    if (tmp1 != 0 && tmp1 != 1)
    print (“You entered an incorrect response!”)
    if (tmp1 == 0){
    delete img//“.mag.1”
    while (tmp1 == 0){
    display (img, 1)
    print (“Please re-mark the stars and save as ‘ds9.reg”)
    print (“Once you are done please enter 1 to continue.”)
    print (“This will re-phot the image frame with the new coordinates”)
    scanf (“%d”, tmp2)
    if (tmp2 != 1)
    print (“You entered an incorrect response!”)
    if (tmp2 == 1){
    coordfile = “ds9.reg”
    phot (img, “”, coords=coordfile, output=“default”, verify=no,
update=yes, verbose=yes)
    print (“Are you satisfied with your results for this frame (no errors)?”)
    print (“Enter 1 to continue, 0 to re-phot”)
    scanf (“%d”, tmp6)
    if (tmp6 != 0 && tmp6 != 1)
    print (“You entered an incorrect response!”)
    if(tmp6 == 1)
```

```
        tmp1 = 1
        if(tmp6 == 0){
        tmp1 = 0
        delete img//“.mag.1”
        }
        }
        }
        }
        if (tmp1 == 1)
        end2 = 1
        }

        # Delete the coordinate file
        delete “ds9.reg”

        # Text dump the coordinates from the above image’s mag file
        # into a new coordinate file
        txdump (img//“.mag.1”, “xcenter,ycenter”, “yes”, headers=no,
parameters=yes, > coordfile)
        }

        # Create the text file for Varstar.
        print (“Nightphot will now create the text file needed for Varstar4 or
Varstar5”)
        print (“What would you like to name the file? (e.g. starB.lst)”)
        scanf (“%20s”, tmp5)
        txdump (imgroot//“*.mag.1”, “id,mag,otime,xairmass,ifilter”, “yes”,
headers=no, parameters=yes, > tmp5)
```

```
# Clean up
delete (imagelist, ver-, >& “dev$null”)

end
```