DEVELOPMENT OF AN AUTOMATED SEARCH ENGINE FOR VARIABLE STARS IN LARGE CLUSTERS

by

Oliver L. Woodland

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

April 2006

Copyright © 2006 Oliver L. Woodland

All Rights Reserved

BRIGHAM YOUNG UNIVERSITY

DEPARTMENT APPROVAL

of a senior thesis submitted by

Oliver L. Woodland

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

Date	Eric G. Hintz, Advisor
Date	Jean-François Van Huele, Research Coord.
Date	Scott D. Sommerfeldt, Chair

ABSTRACT

DEVELOPMENT OF AN AUTOMATED SEARCH ENGINE FOR VARIABLE STARS IN LARGE CLUSTERS

Oliver L. Woodland

Department of Physics and Astronomy

Bachelor of Science

We develop an automated variable star search engine named VARFIND and coded in the MATLAB language. Using a least squares method, VARFIND fits an exponential function to an error vs. magnitude diagram for any given star cluster. It then calculates a robust median statistic for each star and outputs the likely variable star candidates based on this statistic. The search engine was tested on clusters NGC 6882/85, NGC 188, and NGC 6709, which have been previously searched for variable stars. Results indicate that VARFIND is effective in narrowing a search to the likely variable candidates, which can then be further analyzed to find variability. VARFIND is especially suited to large clusters.

ACKNOWLEDGMENTS

I would like to thank Dr. Eric Hintz for his advice and direction in making this project happen and his wife Maureen for her valuable help in editing and revising. I appreciate Dr. Jean-Francois Van Huele and Dr. Justin Peatross for their help in formatting and editing. I also thank Andrew Davis for the inspiration and framework of VARFIND, Ben Rose for his patience and hard work in aligning his program with my needs, Kathleen Moncrieff for the use of her data in testing, my family for their support, and most of all my wife, Kelsey, and son, Jonas, for making it all worth it.

Contents

List of Figures

List of Tables

Chapter 1

Background and Motivation

1.1 Accurate Distance Measurements

Modern astronomy depends upon accurate distance measurements. The most accurate method involves measuring the motion of a star in the sky due to the earth's revolution around the sun and then calculating geometrically the distance to that star. Unfortunately, as we study more distant stars, this motion called parallax becomes increasingly small until measuring it becomes impossible due to instrumentation limitations. It is this problem that leads us to seek other methods of determining distances. As we'll see in this work, studying variable stars in stellar clusters helps us to find distances to objects that might be otherwise unmeasurable and is therefore critical to improving our understanding of the universe.

Clusters are a good place to look for variable stars because they consist of stars at a similar distance and of similar age and composition. However, clusters may consist of thousands of stars and finding variable stars individually takes a lot of time. Technology has improved aspects of variable star research, yet large clusters remain difficult to study as variable stars are found solely by rigorously studying

individual light curves. An example of a light curve from a typical variable star is shown in Fig. 1.2. BYU focuses on variable star research and searches in clusters are common yet time consuming. This research develops an automated search engine that locates variable star candidates and improves the efficiency of cluster searches. These candidates must then be examined rigorously to confirm variability and determine type. This search engine named VARFIND is based on a statistical approach and is coded in the MATLAB language.

1.2 CCDs and the Magnitude Scale

While the human eye has been the primary method for astronomical imaging for thousands of years, more efficient techniques have been developed recently. The most significant of these uses a Charge-Coupled Device (CCD), a semi-conducting electronic device that counts photons as they come in.

Currently, the CCD is nearly universally used for counting photons that enter a telescope. There are several reasons for this: efficiency and linearity in counting photons. The human eye is able to detect about 1 out of every 100 photons giving it a quantum efficiency of 1%. In contrast, a CCD boasts a quantum efficiency of nearly 100%. [1] Smaller telescopes become much more valuable with CCD cameras that capture more light. CCDs have a linear response across a wide range of frequencies, so intensities can be measured very accurately.

A CCD is a matrix of pixels that act as "wells" for collecting electrons. It works by a process very similar to the photoelectric effect in that when a photon strikes a semi-conducting pixel, it excites an electron which is then stored in the well. The number of electrons stored in the well is proportional to the brightness of the image at that particular location on the CCD. A single pixel on a CCD chip may store as

many as 70,000 electrons before becoming saturated. [2] The brightness of multiple stars can be found simultaneously using CCD photometry, making it an invaluable tool in the search for and study of variable stars.

The human eye has a logarithmic response to changes in brightness and is therefore fairly insensitive to changes in brightness. The Greek astronomer Hipparcos established the original magnitude scale by ranking stars by the numbers one through six; one being the brightest and six being the faintest. Traditions die slowly in astronomy, and so we continue this $\log_{10}$ scale for all stars: two stars that differ by five magnitudes actually differ 100 times in brightness. The CCD's linear response explains its popularity with astronomers.

1.3 Variable Stars

A variable star is a star whose brightness changes over time. We know a few reasons why this can happen and so variable stars are classified accordingly. Variability is generally tracked by the creation of light curves, which plot the star's magnitude over time. Astronomical times are usually recorded using the Heliocentric Julian Date (HJD): a time system recorded by the number of days since the start of the Julian calendar that is also corrected for the earth's motion about the sun. The two main types of variable stars are extrinsic and intrinsic variable stars.

The brightness of an extrinsically variable star only changes as a result of our line of sight to the star. For example, a common type of an extrinsic variable is an eclipsing binary. An eclipsing binary system is a set of two or more stars that orbit each other and happen to cross each other's path from our point of view. The total brightness of the system changes as one star passes in front of another and blocks some of the light. Often, these objects are much too distant to resolve the stars

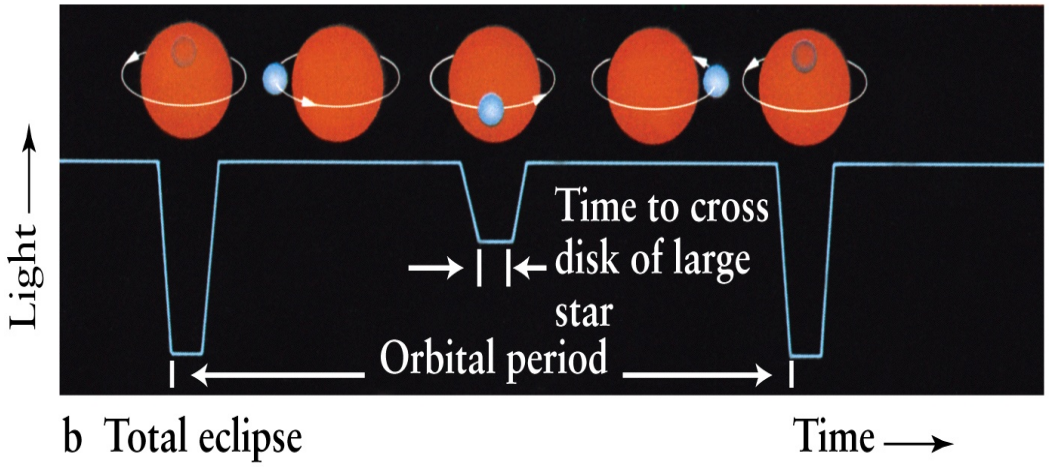

Figure 1.1 An example light curve of a full eclipsing binary (http://homepage.smc.edu).

visually and so we find that they are eclipsing binaries from their light curves. Full eclipsing light curves will generally be flat with sudden drops where one star passes behind another as is shown in Fig. 1.1. If the stars are of different sizes, the light curve will have two different minimum heights as shown. It may seem unlikely that many stars line up so perfectly as to eclipse each other. The chances of running into eclipsing variables is quite high in a variable star search because approximately half of the visible stars belong to multiple star systems.

Intrinsically variable stars vary in brightness because the stars themselves are inherently unstable and change brightness on their own. Intrinsically variable stars can be further categorized into cataclysmic variables and pulsating variables.

Cataclysmic variability occurs when a star goes through some type of violent change that rapidly affects its brightness. These variable stars include supernovae and recurring novae and are very valuable to understand the distance ladder of the universe and its composition. They also give us information on stellar evolution, as the stars approach the end of their lives and run out of elements to fuel their nuclear

fusion. Unfortunately, the likelihood of catching one on a CCD from start to finish is quite slim. Automated sky searches are aiding in studying these objects.

Pulsating variable stars are fascinating; they change brightness due to actual physical pulsation going on within the stellar atmosphere. Gravity pulls to implode the star into a black hole while the outward pressure from the nuclear fusion tends to explode the star into the surrounding area. Pulsators are stars that are unable to achieve a steady equilibrium between gravity and radiation pressure but pulsate on some regular interval. An example of a pulsating variable of the RR Lyrae type can be found in Fig 1.2. Note that the magnitude scale on the left of the plot is inverted; indicating that the peak is a time of maximum light. The curve is asymmetric as is common in pulsating variables, indicating complicated physical processes in the atmosphere of the star. High amplitude pulsating stars are generally easily recognizable, while short-period and smaller-amplitude pulsators require a much closer look.

Most stars go through several phases of variability as they evolve. For some stars, one of these phases may be the Cepheid stage named after δ Cephei; the prototype of this type of variability. Cepheid variables are important because of their key role in finding stellar distances.

In 1912 while at Harvard University, astronomer Henrietta Leavitt discovered that in the Small Magellanic Cloud (SMC), a small companion galaxy that orbits our own, visible from the southern hemisphere, the Cepheid's apparent magnitudes were related to their periods of pulsation. Because these stars were all approximately the same distance away in the SMC, this meant that their luminosity was related to their period. This relationship between the period and luminosity of a Cepheid variable is simply known as the Period-Luminosity (P-L) relation. The relation was calibrated by measuring the parallax of Polaris, the nearest typical Cepheid. The result is that one can find the brightness of a Cepheid by simply measuring its period. By comparing

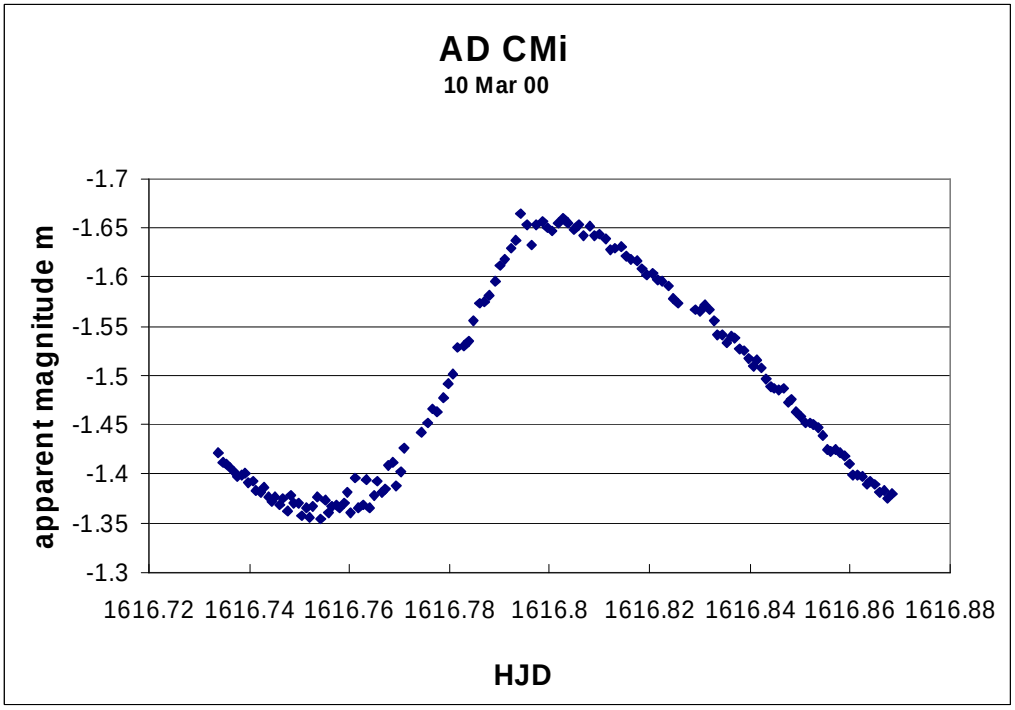

Figure 1.2 A light curve of pulsating variable AD CMi by Mana Vautier.

the apparent brightness of a star to its actual brightness if it were ten parsecs away, distances can be calculated according to the following relation,

$$m - M = 5 \log (d) - 5. \tag{1.1}$$

The quantity $m - M$ in Eq. 1.1 is the distance modulus, where M is the absolute magnitude, or the magnitude the star would be if it were 10 parsecs away, m is the apparent magnitude as observed through the telescope, and d is the distance to the star in parsecs.

1.4 Star Clusters

A star cluster is simply a group of stars that formed around the same time, from the same condensing gas cloud. The members of the cluster are gravitationally bound to

each other and are at approximately the same distance from us. Star clusters contain stars that are roughly the same age and chemical composition, yet of all different masses and evolutionary stages. They provide us with a great opportunity to study stellar evolution and how size and mass influence that evolution. Finding variable stars and using the P-L relation gives distances to clusters. Knowing the distance can give luminosity values for the surrounding stars, helping determine accurate sizes for stars.

There are two types of star clusters: open or galactic clusters and globular clusters. Open clusters are loosely bound gravitationally and generally contain on the order of a few hundred stars. They typically contain young, hot stars; less than 10^8 years old with temperatures of 20,000 to 50,000 K. [2] Open clusters should have fewer stars approaching the unstable stages of evolution than globular clusters. They are located generally in the spiral arms of our galaxy. One well known example of an open cluster is the Pleiades, as shown in Fig. 1.3.

Globular clusters are much older, some dating back billions of years. They can contain thousands of stars and are much more densely packed than open clusters. There are around 200 known globular clusters in our galaxy and they may each contain hundreds of thousands of stars. They are generally located in the galactic halo, the spacious region outside the disk of the galaxy, or in the galactic center. One example of such a cluster may be seen in Fig. 1.4.

1.5 Current Challenges in Variable Star Searches

Traditionally, variable star searches in clusters are conducted by collecting data on a cluster over a few different nights, performing photometry on the images, and then studying the individual light curves. Each star for each night must me analyzed for

Figure 1.3 The Pleiades, a well-known open cluster (NASA/ESA/AURA/Caltech).

trends of variability. Noisy data and long period variability can make identifying variability a difficult task indeed. Sometimes an error magnitude plot can be created for the cluster (detailed in section 1.6) and stars with high error can be given specific attention.

Difficulties arise in searching for variable stars in star clusters for a number of reasons. First, the sheer number of stars in some clusters can be overwhelming. If the cluster is a globular cluster or some other unusually crowded cluster, one can spend months sorting through the multitude of light curves for each star in the cluster and reviewing and analyzing each curve. Second, low amplitude variable stars are easily overlooked as they are upstaged by larger amplitude stars. Current methods make it difficult to pick out the small amplitude variables from a large data set as

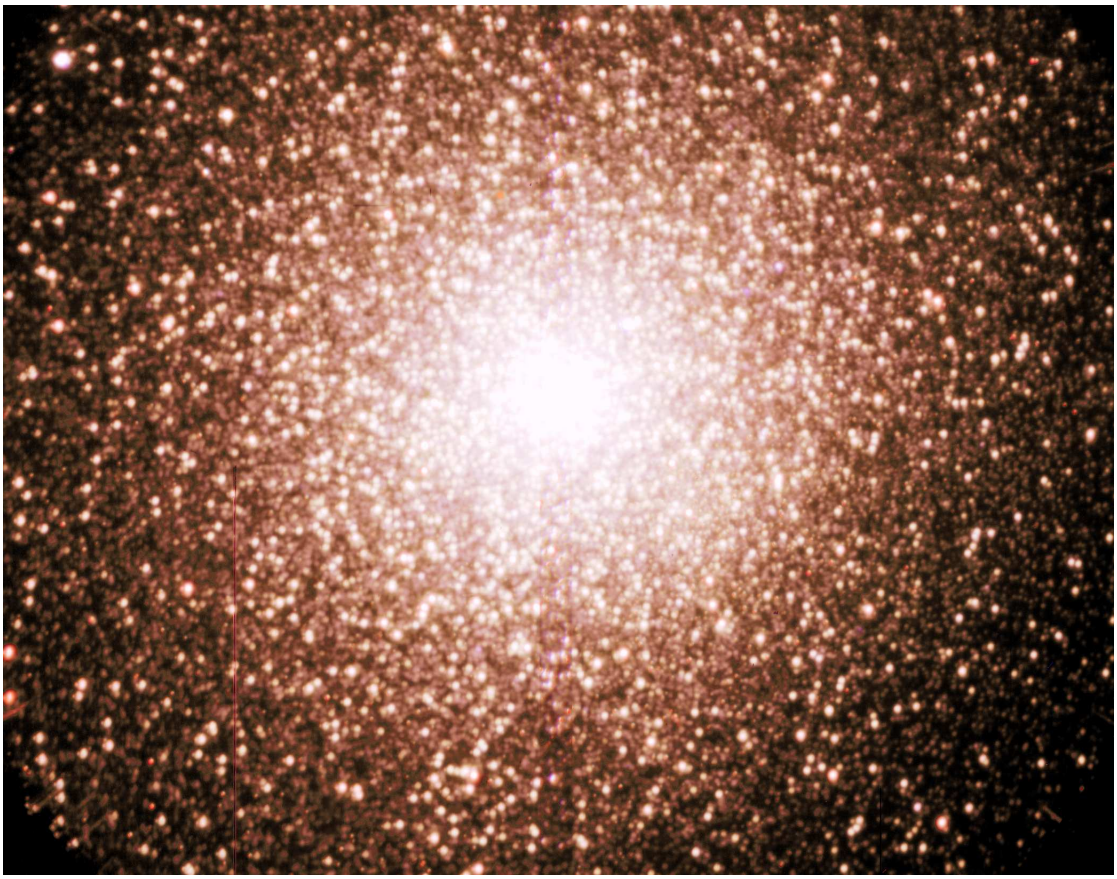

Figure 1.4 47 Tucanae, a bright globular cluster (APOD and South African Large Telescope).

the methods are based on the amount of error in the magnitude as will be discussed in the section 1.6. Third, statistical methods for finding variable stars in clusters are only beginning to be developed.

1.6 Work Underway at BYU

The BYU astronomy research group is heavily involved in variable star research. One emphasis is on finding low-amplitude variable stars in clusters. The primary approach to finding these variables is through differential photometry. Differential photometry

involves comparing light curves to the intensity of a few stars deemed as non-variable and then coming up with magnitudes that are relative to this standard. These non-varying stars form an ensemble which is averaged and subtracted from the magnitudes of each star in the field to give differential magnitudes. VARSTAR5 is a program created by Eric Hintz in order to find these differential magnitudes. [3] An updated version of the program was written by Ben Rose and is entitled CLUSTER. CLUSTER works by allowing the user to choose the ensemble or set of stable stars based on an initial error per observation calculation for each star in the cluster. "Error" as it is used in this work will be defined as the standard deviation per observation

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(m_i - \bar{m})^2}{N-1}}, \tag{1.2}$$

where m_i is the observed magnitude of the individual star and $\bar{m}$ is the average observed magnitude of the star. N is the number of observations or data points.

The stars with the lowest error are then selected as the ensemble stars. Their brightness is averaged so that they can be used as the basis for comparison for all the other stars. CLUSTER then outputs a file including the star number, final error, and average differential magnitude for each star. From this information, an error versus magnitude plot may be created. See Fig. 1.5 for an example of such a plot. Note how the average error begins to increase as stars become fainter and fainter, producing a well defined curve. This occurs because fainter stars provide fewer photons for the detector to count, so the error bars in the observations will be greater. From this point forward, m can be assumed to be differential magnitude unless otherwise specified.

Error magnitude curves are frequently used to find variable stars in a cluster as there is a distinct curve to the graph and any star with an unusually high error will

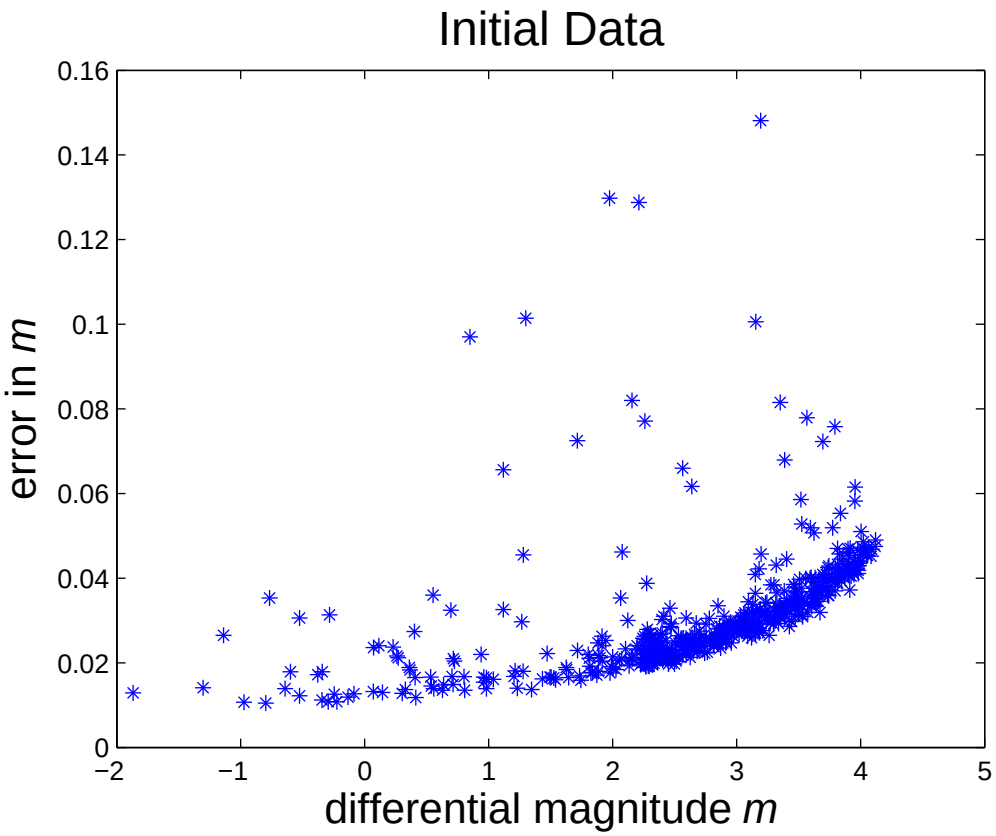

Figure 1.5 An Error vs Magnitude plot created from data analyzed using CLUSTER.

be well above the curve and may be variable. This approach is helpful for finding variables of large amplitude. It is less effective for finding variables of low amplitude as they will not be far off the curve and may be mixed in with stars that are not variable but simply have a large amount of noise in their observed differential magnitudes.

Enoch et al. (2003) developed a Robust Median Statistic (RoMS) further explained in Section 2.4 for the purpose of finding variability in L/T dwarfs. [4] L/T dwarfs are considered brown dwarfs, or stars that never had enough mass to burn hydrogen, so they do not radiate strongly and are very faint. [2] The RoMS is therefore adept at finding variability in low amplitude and faint stars. We have used MATLAB to create a program called VARFIND that builds on what is achieved by CLUSTER

and uses output files from CLUSTER. VARFIND finds a function for the bottom of the curve of an error vs. magnitude plot for a cluster, applies the RoMS to the data, and then gives the statistic which can be used to determine which stars are good variable candidates. VARFIND is especially well suited for large clusters.

1.7 Outline of Program Development

The next chapter outlines the development of VARFIND. It discusses MATLAB and the techniques used to create the program. It covers curve finding and curve fitting as well as more detail about the RoMS. Finally it discusses complications and difficulties, including the functional limitations of the program. In chapter 3, we explain the procedures for testing the program on actual clusters. We then give our conclusions and discuss the impact on future research at BYU and elsewhere as well as notes for further development of the ideas behind VARFIND.

Chapter 2

Development

2.1 MATLAB and Programming Techniques

The software developed in this project was coded in MATLAB, a powerful programming language produced by The MathWorks, Inc. MATLAB was selected for reasons including its ability to handle large arrays of data with relative ease, its versatility and adaptability, and our familiarity with the software from previous coursework. These factors made it an invaluable tool in analyzing large clusters.

The main advantage gained in using MATLAB in this project as opposed to C or other more traditional languages is its capability to manipulate large sets of data. In other languages, large arrays require complicated code to be written. MATLAB treats nearly everything as an array or matrix and receives its name for that reason. It can easily perform intricate operations on large matrices without a hiccup. This is especially useful for our project since we seek to make finding variable stars in a crowded cluster simpler and more accurate. Analyzing 1000+ light curves by hand can prove a very tedious process and using MATLAB as a tool saves hundreds of hours.

While MATLAB is most useful for working with arrays and matrices, it is also a very versatile stand alone programming language. It can be coded to do nearly anything other languages can do, although working with strings can be challenging because all variables in MATLAB are treated as matrices and so we must be very mindful of the dimensions of those matrices in order to perform operations with them. Despite this, MATLAB's adaptability allowed us to create a user-friendly interface, capable of fitting a variety of different clusters.

VARFIND was created keeping in mind that all clusters differ in the number of stars they contain and in the shapes of error magnitude curves. Therefore, it requires a fair amount of interaction from the user in order to tailor it to a particular cluster. Default values work generally for the clusters tested, however, the strength of this project lies in its adaptability to all clusters. The actual MATLAB code written for VARFIND can be found in appendix B.

2.2 Curve Finding

The main purpose of VARFIND is to identify the likely variable star candidates in a star cluster. For VARFIND to function correctly, raw observation data must first be processed through CLUSTER, mentioned in section 1.6, to produce an error versus average magnitude data file and an HJD versus magnitude file. Error versus magnitude plots such as Fig 1.5 show a distinct curve as stars become fainter and the standard of deviation in the observations increases. This happens because fewer photons are collected for the fainter stars and so the error bars in the magnitudes become greater. The bottom of this curve is of particular interest as it represents the line along which the most stable stars should lie. Stars that lie above this curve have a larger error in their measured magnitudes and therefore may be variable stars.

Programming VARFIND to find the bottom of this curve was the first task.

When VARFIND is run, the error vs magnitude file created by CLUSTER is loaded. It then plots the error vs the magnitude and displays the plot for the user in a format like Fig. 1.5. Next, the program asks the user to choose a magnitude interval for finding the bottom of the curve. The default interval in m is set to 0.18 as this proved a convenient interval for most clusters.

VARFIND approximates the bottom of the curve by finding the minimum error value from the data set in each m interval set by the user and saving it to a new array. This new array is then plotted in red circles on top of the original error magnitude diagram as shown in Fig. 2.1.

The goal is to have the red circles mimic the bottom of the actual data set, but since this method is simply an approximation of the bottom of the curve—depending on the interval chosen—some of the red points may not be right along the bottom of the error curve. A number of error checks have been put in place to correct for this problem.

Once the new plot is automatically opened, the user is asked if this interval is acceptable. The user is able to change m interval repeatedly until a suitable interval can be found and a decent approximation of the bottom of the curve has been found.

Next, the user is given the opportunity to correct for large discontinuities in the approximated curve. Fig. 2.2 is an example of a plot with such discontinuities. VARFIND asks the user to give a maximum error difference value that can exist between two consecutive points on the graph. Any large discontinuities in the plot will then not be plotted. In the current version, this maximum error value cannot be increased without rerunning the program from the beginning.

Alternately, individual points can be removed from the curve plot if they seem to make the plot discontinuous. This is done by counting the number of points from

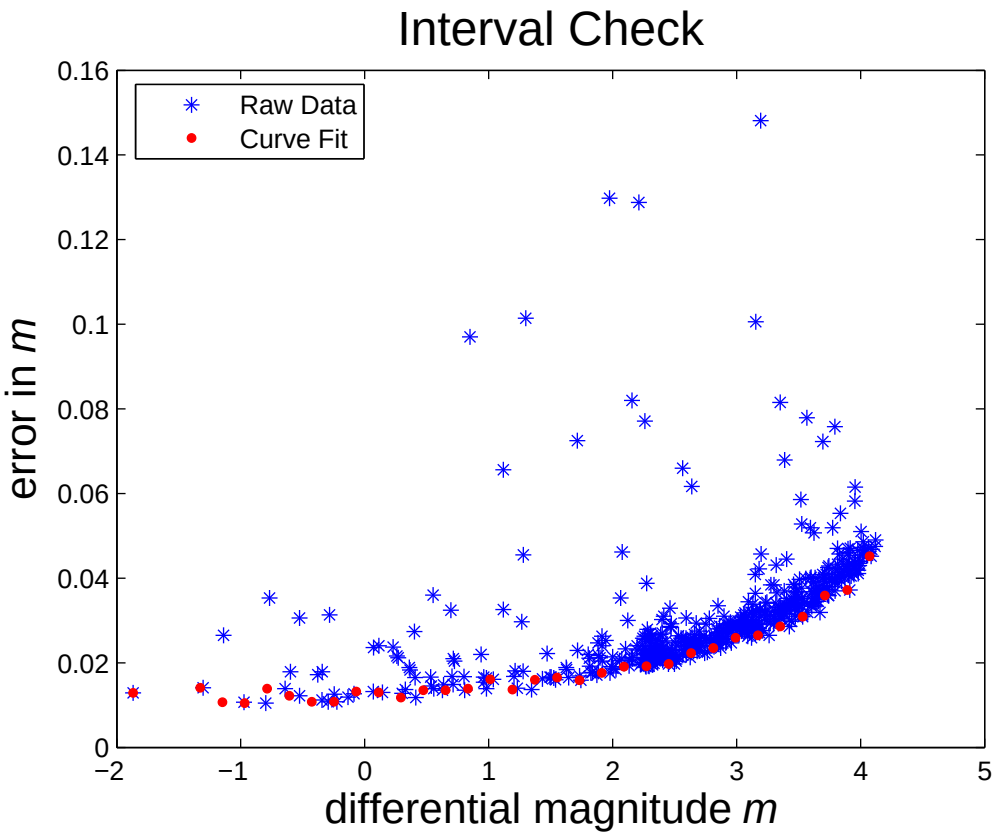

Figure 2.1 An error magnitude diagram of a cluster superimposed with points found along the bottom of the curve. The m interval is set by the user.

the left to the point to be removed. This number is then input into the program by the user and that particular point will be removed. As with the other discontinuity removal, once a point has been removed it cannot be put back in without rerunning the program. At this point, a suitable approximation plot for the bottom of the curve should be in place.

Traditionally, an error magnitude plot has what is considered a flat part on the left where the faintness of the star is not affecting the standard deviation in its magnitude. In Fig. 2.3, the actual curving doesn't usually begin until around $m = 1$. Therefore, at this point VARFIND splits the flat and curved portions of the plot so that actual

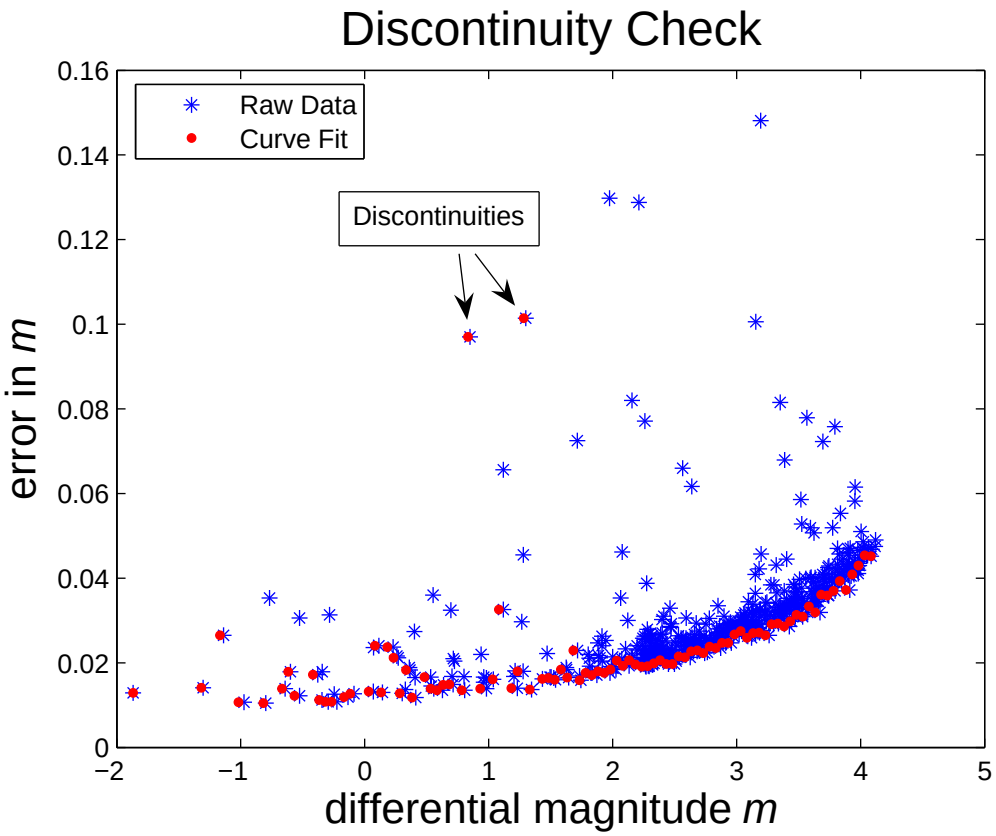

Figure 2.2 An error diagram fit with a plot that contains discontinuities removable by setting a max error value.

mathematical functions may be fit to the curve. VARFIND splits the curve at a default m of 2.5 plus the m of the brightest star. It then opens a new graph with the flat portion plotted in black dots and the curved portion plotted in red dots as can be seen in Fig. 2.3. The user is then asked if the default split value is acceptable and may test different split values until an appropriate split m is established. The plot is now ready to be fit with an exponential function on the right and a flat function on the left.

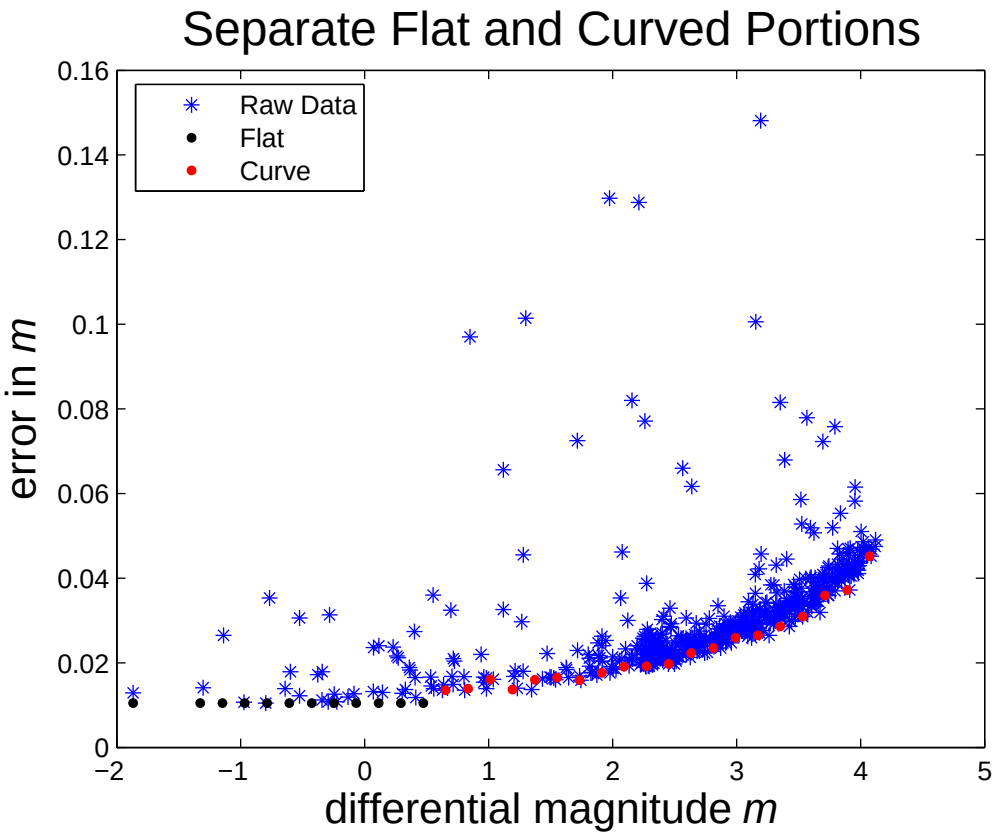

Figure 2.3 An error diagram fit with a plot separated into flat and curved portions taken from VARFIND.

2.3 Curve Fitting

To obtain a mathematical function for the error curve, a least squares approach was used. Adapted from "An Introduction to Matlab," two additional MATLAB scripts are called in the curve fitting process: leastsq.m and leastsqb.m. [5] The curve is approximated using coefficients

$$a = [a_1, a_2, a_3] \tag{2.1}$$

in the following expression

$$\sigma_i = a_1 e^{a_2 x^{a_3}}. \tag{2.2}$$

We found a can usually be initially approximated to be $a = [1, 1, 1]$, corresponding to $\sigma_i = e^x$, provided that enough at least 10 data points exist on the curve. VARFIND can then find reasonable values for the actual coefficients of a from this general approximation. With fewer data points, more trials may be needed in order to fit the curve. $a = [.01, 1, 1]$ is a good choice to try if the default $[1, 1, 1]$ doesn't work.

The flat portion is also approximated in a similar but less complex fashion: the error value of the star with the lowest error. The function value for the flat portion is saved to a variable b. Fig. 2.4 shows an example of an error curve successfully fit with VARFIND. If the fit is successful, the red curve will match the underlying data points quite well as in Fig. 2.4. Note that the flat and curved parts of the approximation will not always meet exactly. This occurs because the flat fit is simply the minimum m value of the stars on that side of the plot. The curve approximation cannot be forced to meet this minimum value. The jump is exaggerated by the fact that this plot is on a narrower scale.

If VARFIND is unable to successfully fit a function to the curve then two things will happen. VARFIND will output an error message and the plots that are automatically opened will show a function in red that obviously doesn't fit the data points. In this case, new approximations for a will need to be made until a proper fit is achieved based upon the plot opened by VARFIND. Once this fit is achieved and the user is satisfied, the equations found will be plotted onto the error magnitude plot as is shown in Fig 2.5 .

2.4 Robust Median Statistic

When a function has been successfully fit to the curve and the coefficients in a and b have been identified, the RoMS may be used to find the likely variable star candidates

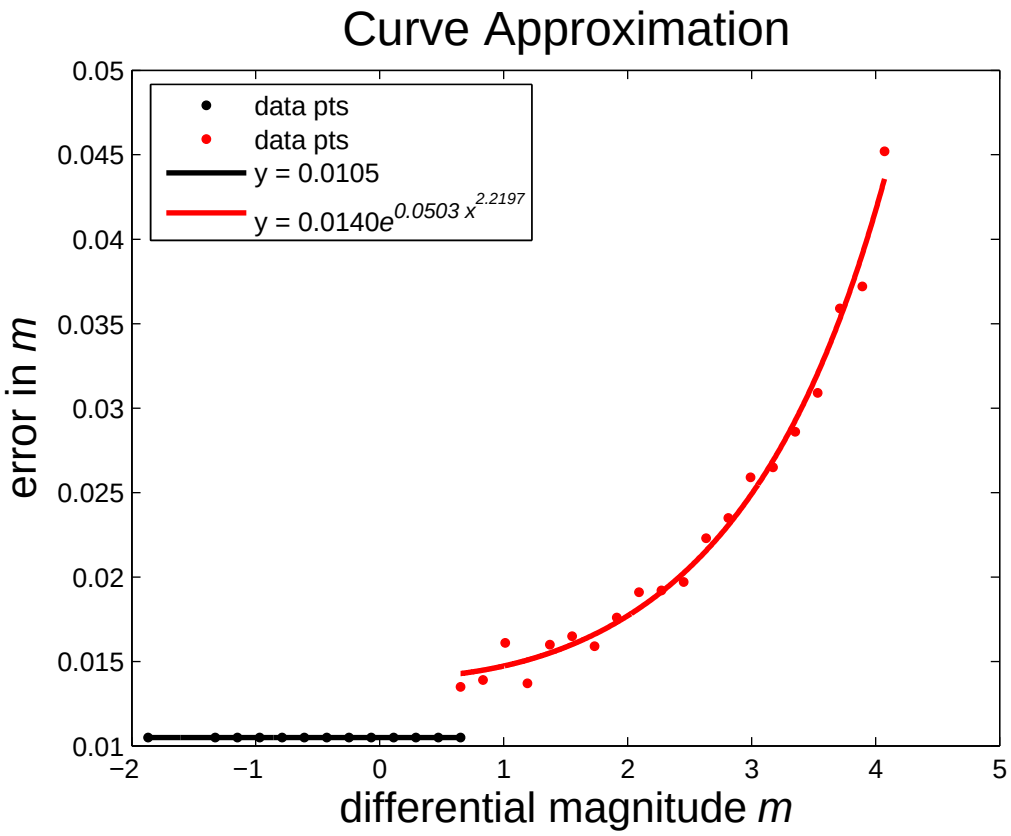

Figure 2.4 An error curve fit with VARFIND. The equations found by VARFIND are listed on the plot.

from the data set. This statistic was first outlined in Ref. [4]. Enoch et al.(2003) developed the statistic for detecting small variations in magnitude of L and T dwarf stars. These are stars of cooler temperature for which the amplitude of variability is small enough to prove very challenging to detect by conventional methods. In 2005, Eric Hintz and Ben Rose expanded the use of the RoMS by applying it in a general search for variable stars and found it to be very effective. [6]

Enoch et al.(2003) define the RoMS in the following manner:

$$\eta = \sum_{i=1}^{N} \left| \frac{m_i - median(m)}{\sigma_i} \right|. \tag{2.3}$$

The m_i refers to the differential magnitude value of the i^{th} observation of a star and

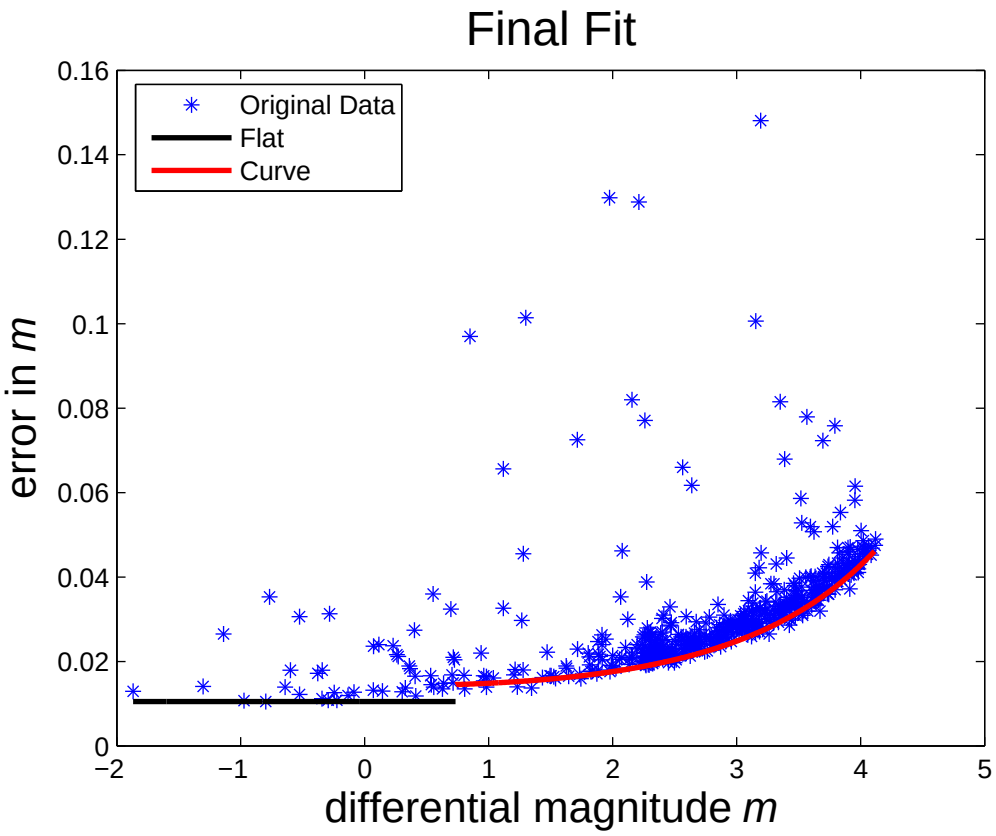

Figure 2.5 An error magnitude plot superimposed with plots of the equations found by VARFIND for use in the RoMS

$median(m)$ is the median differential magnitude value for the whole set of observations of that star. σ_i is defined in Eq 2.2 and is the value of the error curve at the average m for that star. N is the number of observations in the set. Reduced η is defined as $\tilde{\eta} = \eta/d$ where d is the number of degrees of freedom or $N - 1$, and Enoch et al.(2003) establish that those stars with a reduced RoMS $\tilde{\eta}$ less than one are not likely to be varying. A value of $\tilde{\eta} \approx 1$ or larger indicates that the star is probably a variable star. [4] Hintz et al. state also that an $\tilde{\eta}$ value of 0.8 has a 50% chance of being variable. [6] For VARFIND, some calibration of these previously published values may be needed as is explained further in section 3.4.

This robust statistic is implemented in VARFIND by first asking the user to in-

put another CLUSTER output file that contains one column of data containing the HJDs for the observation period followed by columns for each star containing their differential magnitude m at their corresponding HJDs. Using this file in accordance with equation 2.3, VARFIND can then take the m_i for each each star and subtract from it the median(m) and divide that value by the corresponding σ_i using the coefficients in a for the curved portion and b for the flat portion. For more details on the implementation of this statistic, see the MATLAB code in appendix B.

A value for $\tilde{\eta}$ is then calculated for each of the stars in the cluster and entered into a text file named by the user. The file is opened automatically and formatted with columns containing Star ID, $\tilde{\eta}$ (called reta in VARFIND for reduced eta), average m, and likelihood of being variable. Currently, a $\tilde{\eta} > 1$ is listed as "probable", $0.9 \leq \tilde{\eta} \leq 1$ produces the "suspected" designation, and $\tilde{\eta} < 0.9$ is left blank. An example of the output file produced by VARFIND can be found in Tab. 2.1.

At the conclusion of VARFIND, a histogram of the $\tilde{\eta}$ values may be produced at the user's discretion. This information can be useful if any calibration of the values is needed as is discussed in Chapter 3.

2.5 Functional Limitations

VARFIND is capable of handling a cluster with as many stars as can be found in clusters, provided that CLUSTER can do the same beforehand. There is definitely a limit to how few stars would be statistically rigorous. Obviously, enough stars need to be present so that a well defined error curve can be established. This value may vary between clusters, but approximately 50 stars are needed in order to define this curve.

VARFIND is severely limited by the fact that the user must give an approximation

Table 2.1 Typical output file upon completion of VARFIND

Star ID	$\bar{m}$	$\tilde{\eta}$	Variable?	Star ID	$\bar{m}$	$\tilde{\eta}$	Variable?
1	3.1507	1.09071	probable	24	2.3173	1.09777	probable
2	2.4162	0.84505		25	0.2598	1.60064	probable
3	3.5232	0.8162		26	3.7471	0.85024	
4	2.7542	0.92241	suspected	27	2.9237	0.89644	
5	2.3144	0.91508	suspected	28	3.3683	0.89445	
6	3.7408	0.79904		29	2.931	0.9415	suspected
7	3.468	0.83966		30	2.4633	1.32931	probable
8	1.9134	1.20959	probable	31	2.6543	0.93264	suspected
9	3.9467	0.83797		32	3.7937	0.79237	
10	3.7877	0.86946		33	0.343	1.32823	probable
11	3.2315	0.92119	suspected	34	3.0797	0.95643	suspected
12	3.9679	0.81646		35	2.5926	1.16327	probable
13	3.9937	0.81298		36	3.2034	0.85572	
14	2.5053	0.96033	suspected	37	2.951	0.97539	suspected
15	2.2367	1.02476	probable	38	2.8363	0.87362	
16	2.3628	0.95076	suspected	39	2.3874	1.08156	probable
17	1.472	1.10869	probable	40	2.0733	0.89723	
18	3.4708	0.84807		41	2.1044	1.01862	probable
19	2.3062	1.1053	probable	42	0.5316	1.28535	probable
20	3.4257	0.801		43	3.0117	0.91956	suspected
21	2.7803	1.04239	probable	44	3.0673	0.98187	suspected
22	3.3403	0.84735		45	1.8817	1.14339	probable
23	3.9581	0.866		46	1.3041	1.07977	probable

for the coefficients in a. Approximating the function can be somewhat of an art and may require some practice on the part of the user. With trial and error in approximation, we are confident that this program will be able to fit error curves for nearly all types of clusters. Multiple attempts at the approximation may be needed, but the plots that VARFIND produces demonstrate the success or lack of success of the fit with clarity. If the fit is unsuccessful, it will be obvious from the plot because the red line of the fit will not match up at all with the data points beneath it. At this point, another guess should be attempted.

Working with strings and letters in MATLAB can be more of a challenge than with other programming languages. MATLAB treats everything it deals with as a vector, matrix, or other type of array. A string is not a word or sentence, it is an array containing numbers corresponding to specific characters. This treating brings problems as matrix operations require that the dimensions of each array correspond exactly. Error checking and input commands become more difficult due to these factors. Through trial and error, sufficient error checking and string handling was coded into VARFIND.

Chapter 3

Results and Conclusion

3.1 Analysis Methods

We chose to analyze several clusters that were previously well studied and researched for variable stars and compare our $\tilde{\eta}$ values with previously published variable stars and $\tilde{\eta}$ values. These clusters were originally analyzed using traditional methods which include studying light curves on an individual basis and looking for variability in the data. Where possible, we reprocessed the exact data used for the original study of the cluster. This was done in an effort to avoid inconsistencies due to the method of data collection, photometry, or data reduction.

Potential problems in the analysis lie in preparation of the data for VARFIND. As is mentioned earlier, data must be processed through a program called CLUSTER prior to being analyzed with VARFIND. CLUSTER establishes differential magnitudes and the final standard deviation in the magnitudes of each star. This information is key to the proper function of VARFIND and so problems with this software may be propagated through our results.

We mention the above because CLUSTER is yet in the development stages and has

not been fully debugged or finalized. CLUSTER is designed to be an improved version of VARSTAR5, the traditional differential photometry software used at BYU in variable star research. While flaws in CLUSTER will propagate through VARFIND, VARFIND is designed to function as if the data provided were correct and so as improved versions of CLUSTER arise, data collected from VARFIND should have improved accuracy as well. Some of the data were also analyzed after being run through VARSTAR5 to ensure that VARFIND is functioning as designed.

3.2 Analysis of NGC 6882/85

This cluster is an interesting one in that it is unclear at this point whether all the stars in the field belong to one cluster or if it is actually two clusters in the same line of sight. Despite this, the cluster has been well researched and documented for variable stars by Hintz et al.(2005). Also, the RoMS was utilized in finding variable stars, however it was not calculated automatically and therefore provides valuable comparison values.

Hintz et al.(2005) studied over 92 stars in the cluster and established three new variable stars on top of the three already published. In our analysis, we only compare the first 41 stars because the data file provided us only contained data for those stars.

Using VARFIND, 14 of 41 stars were found to have $\tilde{\eta} > 0.9$. VARFIND identified five of six previously established variable stars as potential candidates: stars 1, 8, 23, 28 and 41. However, it also identified two of six previously established non-variables as candidates: stars 9 and 16. It identified two of four previously suspected variables, stars 25 and 38, as suspected candidates. Fig. 3.1 shows a histogram of $\tilde{\eta}$ values.

The peak in the histogram occurs at approximately 0.7, which indicates that the values are calibrated quite well because this is the predicted value where the peak

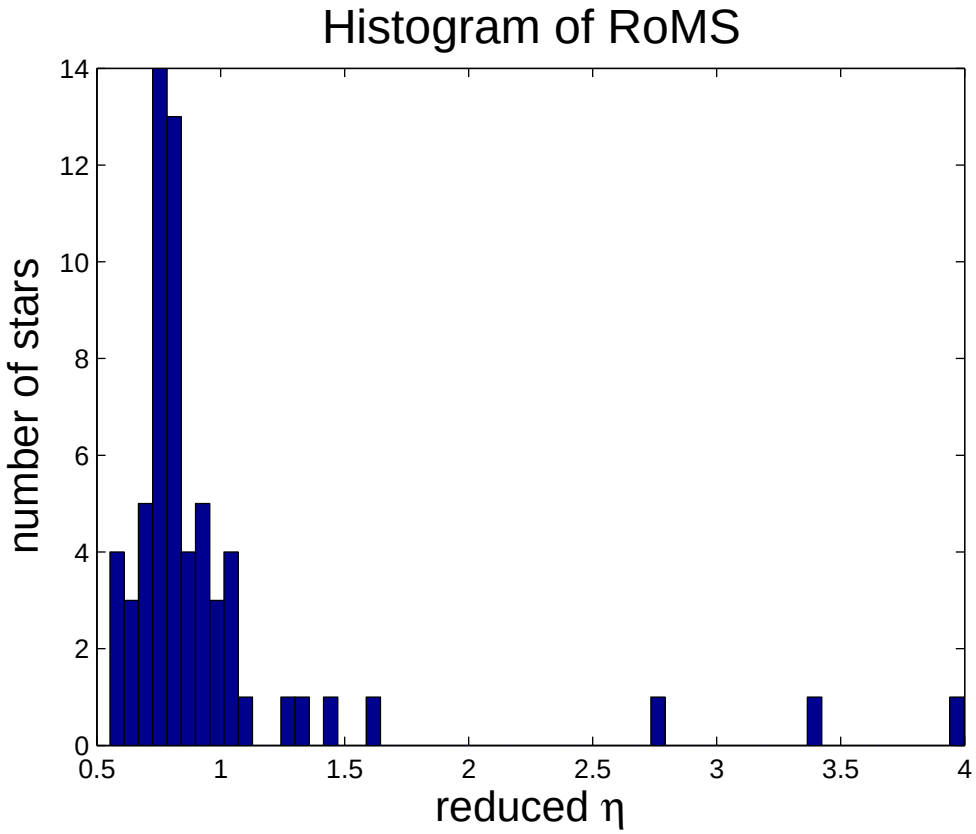

Figure 3.1 A histogram produced by VARFIND of the $\tilde{\eta}$ values for NGC 6882/6885.

should occur. In order to further compare our automatically calculated values with those manually calculated by Hintz et al(2005), see Tab. 3.1.

3.3 Analysis of NGC 188

NGC 188 is an open cluster in the northern sky and is one of the oldest open clusters known. The cluster is fairly close, having a distance modulus of $m-M = 11.43 \pm 0.08$ or approximately 1.93 kpc. [7] Fig 3.2 shows a photograph of the cluster. As of July 2005, only 9 variable stars are known in the cluster, however it is noted that lower amplitude variable stars may have been overlooked. [8] We used the data obtained by

Table 3.1 $\tilde{\eta}$ Comparison: Manual vs. Automated

Star ID	$\tilde{\eta}$ [6]	$\tilde{\eta}$(VARFIND)	Variable? [6]
1	1.022	1.27655	Yes
8	0.993	0.93743	Yes
9	0.801	1.06877	No
13	0.912	0.68098	Yes
15	0.847	0.8653	No
16	0.848	1.12161	No
19	0.71	0.73575	No
23	0.857	0.88722	Susp.
25	0.844	1.03164	Susp.
28	1.02	3.40058	Yes
29	0.811	0.72867	Susp.
34	0.834	0.80853	No
38	0.851	0.90874	Susp.
39	0.844	0.73877	No
41	0.869	55.80936	Susp.

Figure 3.2 NGC 188, an open cluster used in the development and testing of VARFIND.

Davis et al. to compare the previously known variables with those that are suspected by VARFIND.

This cluster was heavily used in the design of this program as it contains 513 stars and has a well defined error curve. Most of the figures taken from VARFIND in this work are error vs. magnitude plots for NGC 188. Andrew Davis, who performed a search in this cluster, first attempted a semi-automated method to calculate the RoMS using this cluster [8]. VARFIND was inspired by Davis' work, and with permission we adapted his MATLAB code for the least squares approximation portion of VARFIND.

Unfortunately, Davis et al.(2005) does not specify how the variable stars he found correspond to the Star IDs from his data files and we are therefore unable to compare our results effectively with his. VARFIND found 104 of 513 stars had a $\tilde{\eta} > 1$. Further study is needed to determine if this is in error or if there are low-amplitude variable

stars that have been overlooked. A histogram of these values can be found in Fig. 3.3. Note that the peak bin is centered around about 0.8, a reasonable value.

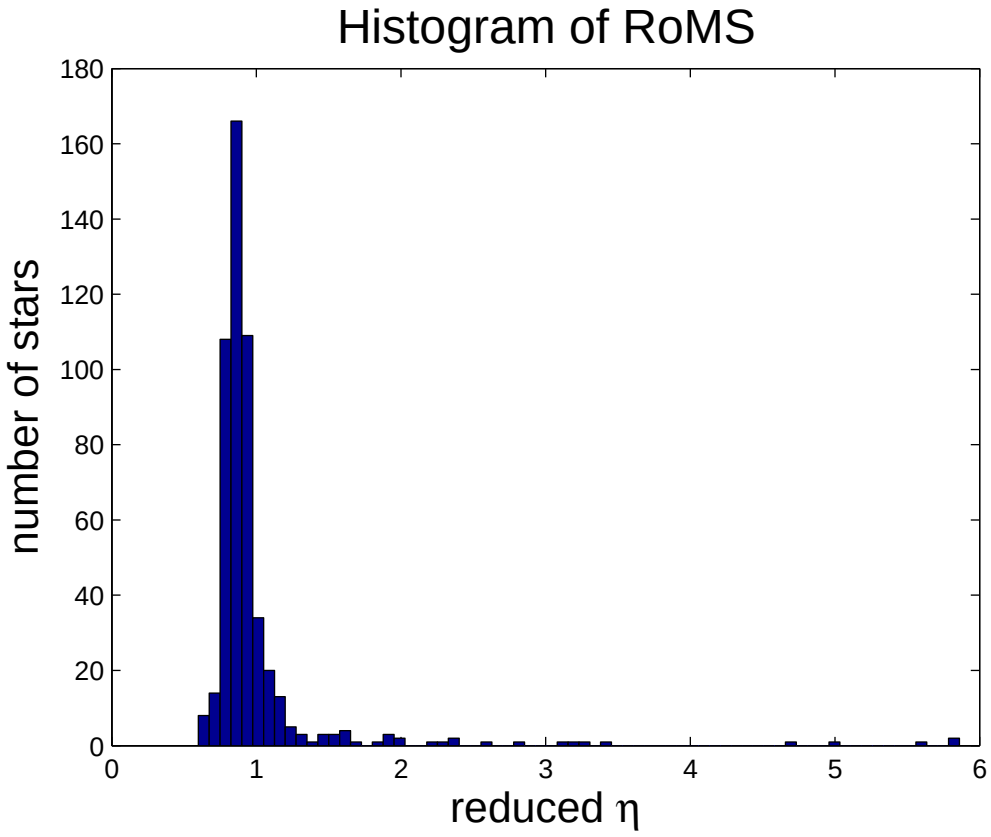

Figure 3.3 A histogram produced by VARFIND of the $\tilde{\eta}$ values for NGC 188.

3.4 Analysis of NGC 6709

Located in the constellation of Aquila, NGC 6709 is a young star cluster. It is a fairly densely packed cluster with 136 stars included in a 2005 survey conducted by Kathleen Moncrieff. [9] Moncrieff et al. found 24 suspected variable stars with two pulsating variables previously known by individually studying each light curve in the field . Again, the same data used to conduct this 2005 survey was processed through

VARFIND to establish a comparison.

A problem seemed to arise with this cluster that did not occur with the others. The $\tilde{\eta}$ values from VARFIND seem to have an offset from what would be expected. In Fig. 3.4, note that the peak value is centered around 1.5 and the width of the spread seems to be much broader than in the other histograms. More analysis and research

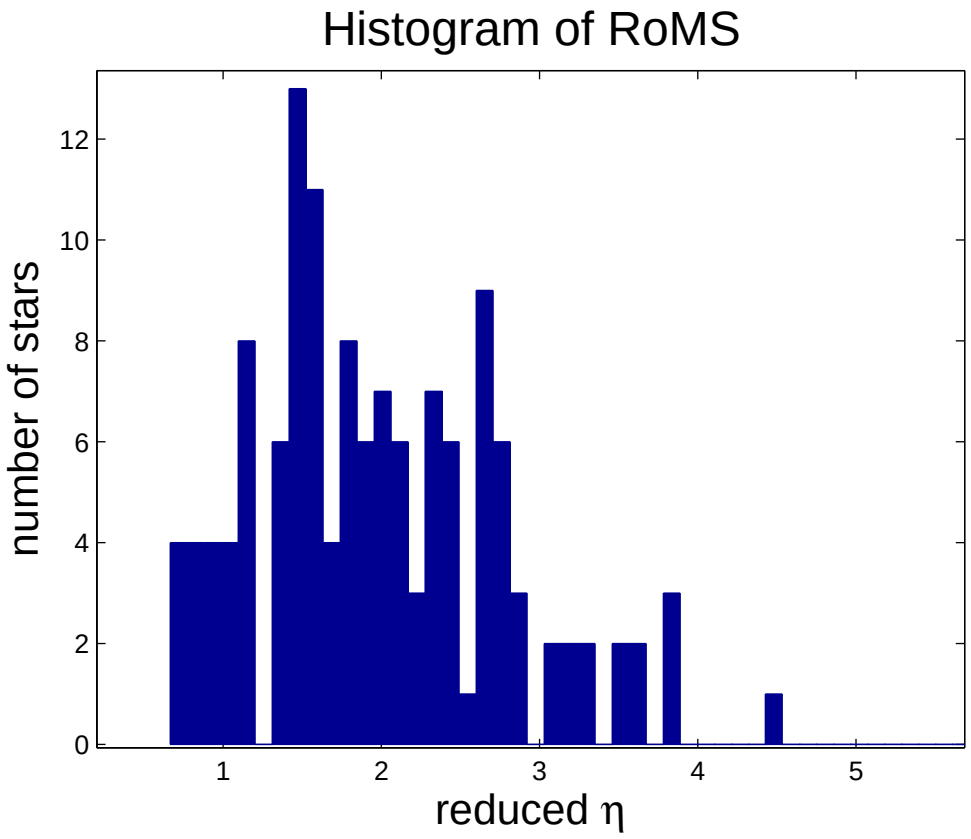

Figure 3.4 A histogram produced by VARFIND of the $\tilde{\eta}$ values for NGC 6709.

is needed to understand if these values can simply be calibrated and trusted, or if the data are unusable. The 14 of 24 the previously established variable stars exhibited a $\tilde{\eta}$ value of approximately 2.0 which is much higher than the histogram peak of 1.5 or the median value of 1.734. This may indicate that a simple calibration could render the data useful for variable star searches after all. The remaining previously established

variables had values closer to 1.5.

CLUSTER found errors much higher in NGC 6709 than those of the other two clusters used. Perhaps excessive noise in the data are offsetting our statistics.

3.5 Conclusions

While VARFIND proves a valuable tool for finding variable stars in clusters, more testing and analysis is required in order to validate the significance of the RoMS used, and the accuracy of VARFIND. At this point this program will not replace the traditional method of scrutinizing light curves, however we have established that VARFIND is effective in narrowing down the search to stars that have an increased probability of being variable.

Due to the automated nature of the software developed, equations found for an error curve may be offset from the actual curve. This produces an offset in the $\tilde{\eta}$ values for each star in the cluster from $\tilde{\eta}$ values calculated manually. NGC 6882/85 and NGC 188 showed reasonable peaks in their $\tilde{\eta}$ histograms, while the histogram of NCG 6709 may be an example of such an offset.

An interesting project would be to look for trends in variability type based on the RoMS. This addresses the following questions: Do eclipsing binaries lie on the histogram at a different location than do pulsators? How will novae or other cataclysmic variables affect results? If trends are found, VARFIND may be even more valuable than previously anticipated.

Further research and testing also might include using VARFIND on a well-studied globular cluster. This could affirm our claims that the software is well-suited for very large clusters and that it can actually handle thousands of stars, as we were unable to verify in this research.

Bibliography

[1] B. W. Carroll and D. A. Ostlie, *An Introduction to Modern Astrophysics* (Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1998).

[2] M. Zeilik and S. A. Gregory, *Introductory Astronomy and Astrophysics*, 4th ed. (Harcourt Brace College Publishers, Orlando, FL, 1998).

[3] E. G. Hintz, M. D. Joner, D. H. McNamara, K. A. Nelson, J. W. Moody, and C. Kim, "Time-Series Ensemble Photometry of SX Phoenicis Stars. I. BL Camelopardalis," Publications of the Astronomical Society of the Pacific **109,** 15–20 (1997).

[4] M. L. Enoch, M. E. Brown, and A. J. Burgasser, "Photometric Variability at the L/T Dwarf Boundary," The Astronomical Journal (2003).

[5] R. Spencer, *An Introduction to MATLAB* (Brigham Young University, Provo, UT, 2000).

[6] E. G. Hintz and M. B. Rose, "Variable Stars in the Field of NGC 6882/6885: The Case of V381 Vulpeculae and V382 Vulpeculae," Publications of the Astronomical Society of the Pacific (2005).

[7] T. von Hippel and A. Sarajedini, "WIYN Open Cluster Study. I. Deep Photometry of NGC 188," Astronomical Journal **116,** 1789 (1998).

[8] A. J. Davis, *A CCD Photometric Search for Variable Stars in the Open Cluster NGC 188* (Brigham Young University, Provo, UT, 2005).

[9] K. E. Moncrieff, *A Photometric Search for Variable Stars in the Open Cluster NGC 6709* (Brigham Young University, Provo, UT, 2005).

Appendix A

Users Guide for VARFIND

Appendix A is a short tutorial for using VARFIND. It discusses what files are included on the VARFIND Data CD, and gives a few tips on approximating curves.

A.1 The VARFIND Data CD:What is included

The data CD includes everything that is needed in order to get started with VARFIND, complete with example data and other data formatting programs. The following files are include on the data CD:

- `UsersGuide.pdf` (This document)
- `VARFIND1_3.m`
- `flatfit.m`
- `funcfit.m`
- `leastsq.m`
- `leastsqb.m`

- `cluster_5.0.c`
- `MagError.txt`
- `HJDmag.out`
- `varstarFormat.m`

A.1.1 Running VARFIND

`VARFIND1_3.m` is the main program script which must be run inside MATLAB. You can then follow the instructions on the screen. The scripts `flatfit.m`, `funcfit.m`, `leastsq.m`, `leastsqb.m` are other MATLAB scripts that must be in your project folder, but do not need to be run independently.

A.1.2 CLUSTER

`cluster_5.0.c` is the updated version of VARSTAR5 entitled CLUSTER, written by Ben Rose. CLUSTER receives an ".lst" file and outputs an average magnitude versus error file and an HJD versus magnitude file in the format needed by VARFIND.

When VARFIND is run, it will prompt you to input these two files from CLUSTER. The magnitude versus error file will be asked for first, and the HJD versus magnitude file will be asked for later on in the program. This `.c` script can be compiled in LINUX by the following command:

"`gcc -lm -o cluster cluster_5.0.c`."

This will compile CLUSTER in your current folder. To run it, type "`.\cluster`" while in that folder, or type the path to the folder in which CLUSTER is located (i.e. "`\home\research\cluster`". It will then prompt you to input your ".lst" file

as well as the names for the two output files mentioned above. For more information on CLUSTER, contact Ben Rose.

A.1.3 Sample Data

`MagError.txt` is an example magnitude versus error file created by CLUSTER and `HJDmag.out` is an example HJD versus magnitude file created by CLUSTER. These files may be used as sample or test data in order to learn to use VARFIND.

A.1.4 Formatting VARSTAR5 Data for Use in VARFIND

The file `varstarFormat.m` is a MATLAB script I created in order to reformat VARSTAR5 data for use in VARFIND. It opens each of the ".dat" files created by VARSTAR5 and compiles them into one single HJD versus magnitude file as is needed by VARFIND. CAUTION: this script is a bit rough and may require some knowledge of MATLAB to run effectively.

A.2 Tips For a Good Curve Fit

A.2.1 Magnitude Intervals

When selecting a magnitude interval overwhich to find the bottom of the error curve, choose one that gives the most possible data points and yet still represents the curve effectively. This allows VARFIND to fit the data points more effectively and your coefficient approximation to be much less accurate. Remember that you can easily remove stray points later by setting a maximum error between consecutive points or manually removing specific points.

A.2.2 Coefficient Approximation

Approximating the coefficients for the curve is definitely the most difficult part of using this program. Trial and error is really the only way to always get a good fit, however I have provided a few suggestions on where to begin.

A good place to start is always with "[1,1,1]". If this guess yields a flat red line on the graph or if VARFIND notifies you that the fit was unsuccessful, the next guess I suggest you try is "[.01,1,1]". You will know that the fit is successful when the red plot fits the red data points and the VARFIND doesn't say anything weird to you.

GOOD LUCK!!!

Appendix B

MATLAB Code for VARFIND

```
clear all; close all; clc; fprintf('\n')
    disp('   ________  _________________________________ ')
    disp('  | __    / /__    ____  ____  _  _  _  ___   |')
    disp('  | \ \  / //  \  |  o \|  __|| || \| || o \  |')
    disp('  |  \ \/ // /\ \ |    /|  _| | ||    ||   /  |')
    disp('  |   \__//_/  \_\|_|\ \|_|   |_||_|\_||__/   |')
    disp('  |___________________\ \_____________________|')

fprintf('\n      Welcome to VARFIND version 1.3!\n') fprintf('
by Oliver Woodland\n\n') fprintf('This software will help you find
and fit the bottom of your\n') fprintf('error vs magnitude curve and
utilize the Robust Median\n') fprintf('Statistic to calculate the
probable variable star candidates. \n') fprintf('It is especially
useful for large clusters or fields of many \n') fprintf('stars.\n')

%loads the data file from the user and saves it to the variable 'magVerror'
fid=0; while fid < 1
   fprintf('\nInput error vs magnitude file name')
   datafile=input('(.err from CLUSTER): ', 's');
   [fid,message] = fopen(datafile, 'r');
   if fid == -1
    fprintf('Filename not found in project folder.\n\n')
   end
end copyfile(datafile,'magVerror'); load magVerror;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% The following section receives the data file from the user and fits the
% bottom of the data curve with a flat portion of the graph and a curved
```

```
% portion.
%
%*************************************************************************
%*************************************************************************

sorted=sortrows(magVerror,[2 3]); mag=sorted(:,2); err=sorted(:,3);

plot(mag,err,'b*') title('Initial Data','fontsize',18)
xlabel('differential magnitude \itm','fontsize',16); ylabel('error
in \itm','fontsize',16);

magmin=min(mag); magmax=max(mag); createcurve=0;

while createcurve == 0
    fprintf('Enter a magnitude interval overwhich to fit curve(def 0.18)')
    dx=input(': ');
    if isempty(dx) == 1
        dx = .18;
    end
    close
    curvex=magmin:dx:magmax;
    curvey=zeros(1,int16((magmax-magmin)/dx));

    i=1; % increments along 'sorted' array locations
    ii=1; % increments through 'small' array locations
    n=1; % increments through 'curvey' array locations
    step=magmin; % increments along stepsize set by user

    % This loop loads the array 'curvey' with stars from the bottom of the
    % curve
    while step <= magmax
        ii=1;
        %small=ones(1,50);
        clear small
        small=10;
        while sorted(i,2) < step+dx & sorted(i,2) >= step
            small(ii)=sorted(i,3);
            i=i+1;
            ii=ii+1;
            if i > length(sorted)
                break
            end
        end
```

```
        curvey(n)=min(small);
        n=n+1;
        step=step+dx;
    end

    % corrects for discontiuities caused by matrix prealloccation to 10
    % and deletes each point with a y value of 10
    i=1;
    while i <= length(curvey)
        if curvey(i)==10
           curvey(i)=[];
           curvex(i)=[];
        else
           i=i+1;
        end
    end

    plot(mag,err,'b*')
    hold on
    plot(curvex,curvey,'ro','MarkerFaceColor','r','MarkerSize',3.2)
    legend('Raw Data','Curve Fit','Location','NW')
    title('\itm Interval Check','fontsize',18)
    xlabel('differential magnitude \itm','fontsize',16)
    ylabel('error in \itm','fontsize',16)
    hold off
    fprintf('\nAre you satisfied with this step size?(y or n)')
    createcurve=strncmpi(input(': ','s'),'yes',1);
    fprintf('\n')
end

close plot(mag,err,'b*')
    hold on
    plot(curvex,curvey,'ro','MarkerFaceColor','r','MarkerSize',3.2)
    legend('Raw Data','Curve Fit','Location','NW')
    title('Discontinuity Check','fontsize',18)
    xlabel('differential magnitude \itm','fontsize',16)
    ylabel('error in \itm','fontsize',16)
    hold off

% corrects for other discontinuities caused by spacing of dx
discont=1; while discont == 1
    i=1;
```

```
    fprintf('Enter max err between consecutive pts')
    fprintf(' to correct large abnormalites.\n');
    fprintf('If no large abnormalities exist press ENTER\n')
    fprintf('(WARNING, this max err cannot be ')
    fixit=input('increased afterwards): ');

    if isempty(fixit) == 1
        fixit = 10;
    end

    while i < length(curvey)
        if (curvey(i+1)-curvey(i)) > fixit
            curvey(i+1)=[];
            curvex(i+1)=[];
        else
            i=i+1;
        end
    end

    fprintf('\n')
    close
    plot(mag,err,'b*')
    hold on
    plot(curvex,curvey,'ro','MarkerFaceColor','r','MarkerSize',3.2)
    legend('Raw Data','Curve Fit','Location','NW')
    title('Discontinuity Check','fontsize',18)
    xlabel('differential magnitude \itm','fontsize',16)
    ylabel('error in \itm','fontsize',16)
    hold off

    fprintf('Do you want to decrease your max err?(y or n)')
    discont=strncmpi(input(': ','s'),'yes',1);
    fprintf('\n')
end

specific=1; while specific == 1
    close
    plot(mag,err,'b*')
    hold on
    plot(curvex,curvey,'ro','MarkerFaceColor','r','MarkerSize',3.2)
    legend('Raw Data','Curve Fit','Location','NW')
    title('Discontinuity Check','fontsize',18)
    xlabel('differential magnitude \itm','fontsize',16)
```

```
    ylabel('error in \itm','fontsize',16)
    hold off

    fprintf('Are there any other specific points that must be deleted?')
    specific=strncmpi(input('(y or n): ','s'),'yes',1);
    if specific == 1
        close
        plot(mag,err,'b*')
        hold on
        plot(curvex,curvey,'ro','MarkerFaceColor','r','MarkerSize',3.2)
        legend('Raw Data','Curve Fit','Location','NW')
        title('Discontinuity Check','fontsize',18)
        xlabel('differential magnitude \itm','fontsize',16)
        ylabel('error in \itm','fontsize',16)
        hold off
        i=input('Enter point # counting from left(pos integer): ');
        curvey(i)=[];
        curvex(i)=[];
    end
    fprintf('\n')
end

close plot(mag,err,'b*') hold on
plot(curvex,curvey,'ro','MarkerFaceColor','r','MarkerSize',3.2)
title('Discontinuity Check','fontsize',18) xlabel('differential
magnitude \itm','fontsize',16) ylabel('error in \itm','fontsize',16)
hold off


%************************************************************************
% This portion splits the fit into a flat portion and a curve portion. The
% user then can accept the default split, or enter a magnitude value where
% the split should occur.
%************************************************************************
close splitcurve=0; seppoint=magmin + 2.5; j=1; while splitcurve==0
    clear newcurvex flatx
    clear newcurvey flaty
    i=1;
    while curvex(i) <= seppoint
        flatx(i)=curvex(i);
        i=i+1;
    end
```

```
ii=1;
while curvex(i) <= magmax
    newcurvex(ii)=curvex(i);
    i=i+1;
    ii=ii+1;
    if i > length(curvex)
        break
    end
end


flatytemp=curvey(1:length(flatx));
errormin=min(flatytemp);
for i=1:length(flatx)
    flaty(i)=errormin;
end


newcurvey=curvey((length(flatx)+1):length(curvey));

plot(mag,err,'b*')
hold on
plot(flatx,flaty,'ko','MarkerFaceColor','k','MarkerSize',3.2)
plot(newcurvex,newcurvey,'ro','MarkerFaceColor','r','MarkerSize',3.2)
legend('Raw Data','Flat','Curve','Location','NW')
title('Separate Flat and Curved Portions','fontsize',18)
xlabel('differential magnitude \itm','fontsize',16)
ylabel('error in \itm','fontsize',16)
hold off

fprintf('Separate the flat portion from the curved portion.\n')
fprintf('Be careful not to move the split too far left, as\n')
fprintf('the curve may become difficult to fit.\n')
if j==1
    fprintf('Is the default split satisfactory? (y or n)')
    splitcurve=strncmpi(input(': ','s'),'yes',1);
    fprintf('\n')
else
    fprintf('Is this split satisfactory? (y or n)')
    splitcurve=strncmpi(input(': ','s'),'yes',1);
    fprintf('\n')
end
```

```
    j=j+1;

    if splitcurve==0
        seppoint=input('Enter magnitude value where split should occur: ');
        if isempty(seppoint) == 1
            seppoint = magmin + 2.5;
        end
    end
    close
end

%************************************************************************
% This portion fits the flat part with a function
% ************************************************************************
plot(mag,err,'b*') hold on
plot(flatx,flaty,'ko','MarkerFaceColor','k','MarkerSize',3.2)
plot(newcurvex,newcurvey,'ro','MarkerFaceColor','r','MarkerSize',3.2)
legend('Raw Data','Flat','Curve','Location','NW') title('Separate
Flat and Curved Portions','fontsize',18) xlabel('differential
magnitude \itm','fontsize',16) ylabel('error in \itm','fontsize',16)
hold off

%this part connects the dots so that we don't have a gap in the fit
flatx(length(flatx)+1)=flatx(length(flatx))+dx;
flaty(length(flaty)+1)=flaty(length(flaty));


%************************************************************************
%************************************************************************
%    End of curve finding portion
%************************************************************************
%************************************************************************

% this fits the curved portion
x=newcurvex;y=newcurvey; xmin=min(x);xmax=max(x); npts=1001;
dx=(xmax-xmin)/(npts-1); xplot=xmin:dx:xmax;

fit=0; ahoy=1; while fit == 0
    if ahoy==1
        fprintf('Enter an initial guess for the function.\n')
        fprintf('Fit is of the form a1*exp(a2*x)^a3 .\n')
        fprintf('Enter parameters [a1,a2,a3] in vector form')
        a=input(' [...].(def [1,1,1]): ');
```

```
    fprintf('\n')
else
    fprintf('Enter another guess for the function.\n')
    fprintf('Fit is of the form a1*exp(a2*x)^a3 .\n')
    fprintf('Enter parameters [a1,a2,a3] in vector form')
    a=input(' [...].(def [1,1,1]): ');
    fprintf('\n')
end

if isempty(a) == 1
    a=[1,1,1];
end

option=optimset('TolX',1e-5,'MaxFunEvals',10000);
a=fminsearch(@leastsq,a,option,x,y);
yplot=funcfit(a,xplot);
%prints the equation of the curve
fprintf('\nCurve approximated to:\n\ty = %7.6f*exp(%7.6f x^%7.6f)',a)
fprintf('\n\n')

%the next section fits the flat data
j=flatx; k=flaty;
jmin=min(j);jmax=max(j);
dj=(jmax-jmin)/(npts-1);
jplot=jmin:dj:jmax;

b=1;
b=fminsearch(@leastsqb,b,option,j,k);
kplot=flatfit(b,jplot);
fprintf('Flat approximated to:\n\ty = %7.6f\n\n',b)
%this file should have the real data
close
mag1=magVerror(:,2);rmser=magVerror(:,3);
plot(j,k,'ko','MarkerFaceColor','k','MarkerSize',3.2)
hold on
plot(x,y,'ro','MarkerFaceColor','r','MarkerSize',3.2)
plot(jplot,kplot,'k-','LineWidth',2)
plot(xplot,yplot,'r-','LineWidth',2)
xlabel('differential magnitude \itm','fontsize',16)
ylabel('error in \itm','fontsize',16)
s=sprintf('y = %5.4f\\ite^{%5.4f \\itx^{%5.4f}}',a);
r=sprintf('y = %5.4f',b);
legend({'data pts','data pts',r,s},'fontsize',15,'Location','NW')
```

```
    title('Curve Approximation','fontsize',18)
    hold off

    fit=strncmpi(input('Are you satisfied with this fit?(y or n): ','s')...
        ,'yes',1);
    fprintf('\n')
    ahoy=ahoy+1;
end

close plot(mag1,rmser,'b*') hold on
plot(jplot,kplot,'k-','LineWidth',2)
plot(xplot,yplot,'r-','LineWidth',2)
    xlabel('differential magnitude \itm','fontsize',16)
    ylabel('error in \itm','fontsize',16)
    title('Final Fit','fontsize',18)
    legend('Original Data','Flat','Curve','Location','NW')
    hold off

%The following uses robust data i hope :)
fid=0; while fid < 1
   filename=input('Input HJD v mag (.OUT) file name from CLUSTER: ','s');
   [fid,message] = fopen(filename, 'r');
   if fid == -1
       disp(' ')
       disp('Filename not found in project folder.')
   end
end

copyfile(filename,'HJDfile'); load HJDfile;

[N,n]=size(HJDfile);
%************************************************************************
% The following contains the actual calculation of the Robust Median
% Statistic.  The stat is saved to the variable reta.
%************************************************************************
i=1; j=1; for i=1:n-1
    if mag1(i) < seppoint
        sigma(i) = b;
    else
        sigma(i)=a(1).*exp(a(2)*mag1(i).^(a(3)));
    end
    medm(i) = median(HJDfile(:,i+1));
    for j=1:N
```

```
        delm(j,i)=HJDfile(j,i+1);
        dif(j)=delm(j,i) - medm(i);
        tot(j)=abs(dif(j)/sigma(i));
    end
    eta(i)=sum(tot);
    reta(i)=eta(i)/(N-1);
    if reta(i) < .9
        vari(i,1:9)='         ';
    elseif reta(i) > 1
        vari(i,1:9)='probable ';
    else
        vari(i,1:9)='suspected';
    end
end
%Printing output to file in format -- star id, mag, reta, variable?
disp(' ') outfile=input('Input file name for VARFIND output file:
',’s’); out(:,1)=1:n-1; out(:,2)=mag1; out(:,3)=reta;
fid=fopen(outfile,'w');
fprintf(fid,' %s\n',outfile);
fprintf(fid,'**********************************\n'); fprintf(fid,'*
Star   Avg    Reduced   Is it   *\n'); fprintf(fid,'*  ID    mag
eta   variable? *\n');
fprintf(fid,'**********************************\n'); for i=1:n-1
    fprintf(fid,'%5g %8.5f %8.5f %s\n',out(i,:),vari(i,1:9));
end fclose(fid);
fprintf('\nRobust statistic output to %s.\n\n',outfile);
open(outfile);

%creates a histogram of the results
fprintf('Would you like a histogram of the results?(y or n)')
histo=strncmpi(input(': ','s'),'yes',1); while histo==1
    N=input('Enter number of bins needed for histogram: ');
    figure
    hist(reta,N)
    title('Histogram of RoMS','fontsize',18)
    xlabel('reduced \eta','fontsize',16)
    ylabel('number of stars','fontsize',16)
    %xlim([0 10])
    %ylim([-1.75 3.25])
    fprintf('Are you satisfied with the number of bins?(y or n)')
    histo=strncmpi(input(': ','s'),'no',1);
    if histo == 1
        close
```

```
    end
    fprintf('\n')
end delete('magVerror') delete('HJDfile') fprintf('...Done\n\n')
```