The Search for Brown Dwarf Binary Systems:

Improving the Spectral Fitting Code

Savanah Turner

A senior thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Bachelor of Science

Denise Stephens, Advisor

Department of Physics and Astronomy

Brigham Young University

April 2020

ABSTRACT

The Search for Brown Dwarf Binary Systems:
Improving the Spectral Fitting Code

Savanah Turner
Department of Physics and Astronomy, BYU
Bachelor of Science

While many new brown dwarf observations have been made since the first was discovered in 1995, there is still much that remains unknown about these failed stars. Many stellar formation theories fail to explain the existence of objects with masses as low as brown dwarfs. One key to learning more about the formation of low-mass objects will be determining an accurate binary fraction, helping to either support or reject the different formation models based on their various binary fraction predictions. However, because brown dwarf binaries have small angular separations they can be difficult to visually resolve. A spectral fitting code has been developed here at Brigham Young University that works to identify unresolved brown dwarf binaries by statistically comparing the spectrum of the target object to model brown dwarf spectra. Spectral fitting is the only way to find many binary pairs, and thus is critically important to the efforts to find the binary fraction. I have improved upon this code by adapting it to use real brown dwarf spectral data rather than models in the fitting routine. The new code is cleaner, faster, and more user-friendly. I have verified my results by comparing them to the findings of others in the field. This improved spectral fitting code will enable our team to more efficiently search for unresolved binary brown dwarf systems and learn more about this universe we call home.

Keywords: Brown Dwarf, Infrared Astronomy, Spectroscopy, Spectral Fitting, Binary Pairs, Stellar Formation

ACKNOWLEDGMENTS

I would like to acknowledge my research advisor, Dr. Denise Stephens, for the many hours she has spent teaching, guiding, and supporting me in both my coursework and this research project. I hope to be half the woman she is one day and look forward to continuing to learn from her in the future.

I would also like to thank the other minds that have worked to develop the many iterations of this code that came before my time working with Dr. Stephens, mainly Dr. Tom Stephens and John Michael Eberhard. I would also like to thank Amber Berry for her work identifying data files to use as models.

Finally, I would like to thank my family for their never-ending support.

Contents

Table of Contents v

List of Figures vii

List of Tables vii

1 Introduction 1
- 1.1 Brown Dwarfs . . . 1
  - 1.1.1 Background . . . 1
  - 1.1.2 Discovery and Classification of Brown Dwarfs . . . 4
- 1.2 The Importance of Binary Systems . . . 7
  - 1.2.1 Questions About Brown Dwarf Formation . . . 7
  - 1.2.2 Proposed Adaptations and the Binary Fraction . . . 9
- 1.3 The Difficulty of Resolving Binary Pairs . . . 10

2 Spectral Fitting 13
- 2.1 Other Methods of Binary Identification . . . 13
- 2.2 Prior Work in Spectral Fitting . . . 14
  - 2.2.1 Burgasser's Work and Results . . . 14
  - 2.2.2 Bardalez-Gagliuffi's Work and Results . . . 15
  - 2.2.3 Past Work Done at BYU . . . 17

3 Improving the Code 20
- 3.1 Writing the Pipeline . . . 20
  - 3.1.1 The Old Code . . . 20
  - 3.1.2 First Things First: One Language . . . 21
  - 3.1.3 A Single Command . . . 22
  - 3.1.4 Multiple Objects . . . 23
- 3.2 Switching to Real Data . . . 24
  - 3.2.1 Choosing the Template Objects . . . 24
  - 3.2.2 Preparing the Spectral Templates . . . 25
  - 3.2.3 Adapting the Code to the Spectral Templates . . . 26

4 Results and Conclusions **27**
4.1 Results 27
4.1.1 Comparison to Previous Results 27
4.1.2 Speed of the Code 32
4.2 Conclusion and Future Work 36

Appendix A Workflow: Updated Instructions for Using the Code **38**

Appendix B massConvert2pt0.py **41**

Appendix C MassSmoothConvert2pt0.py **45**

Appendix D pipeline.py **49**

Appendix E WTII.py **53**

Bibliography **59**

Index **61**

List of Figures

1.1 Example spectra of a small star, two brown dwarfs, and the planet Jupiter. 2

1.2 Spectra from an L dwarf and a T dwarf 6

1.3 The progression of resolved to unresolved telescope images of a pair of objects placed close together. 11

1.4 Resolved vs Unresolved HST Images. 12

2.1 One of Burgasser's Final Plots. 16

2.2 An Example of Farnbach's Smoothing Process 18

4.1 New best fits for 10 of Eberhard's 12 objects. 31

4.2 New best fits for 13 of Burgasser's 20 objects. 36

List of Tables

4.1 Eberhard's table of identified spectral binary objects. 28

4.2 Results for 10 of Eberhard's 12 objects. 29

4.3 Results for 13 of Burgasser's 20 objects. 33

Chapter 1

Introduction

1.1 Brown Dwarfs

In 1995, a curious object with properties of both a star and a planet was discovered by a team based in Tenerife, Spain (Rebolo et al. 1995). This object was given the name "brown dwarf," and over 2500 brown dwarfs have been observed in the 26 years following that initial discovery. Certain characteristics of these odd objects fail to mesh with current understanding of the requirements for stellar formation, so it is hoped that an improved understanding of brown dwarfs will help us to better understand how stars are formed (Eberhard 2020). Thus, brown dwarf research is of critical importance to the astronomical community.

1.1.1 Background

When you look up at the night sky, you are looking at stars of all different brightnesses, colors, temperatures, and sizes. Many of these characteristics can be observed from the spectrum of a star, referring to how much light the star is emitting at various wavelengths (see Fig. 1.1).

For one thing, although stars are not perfect blackbodies, they can be approximated as such,

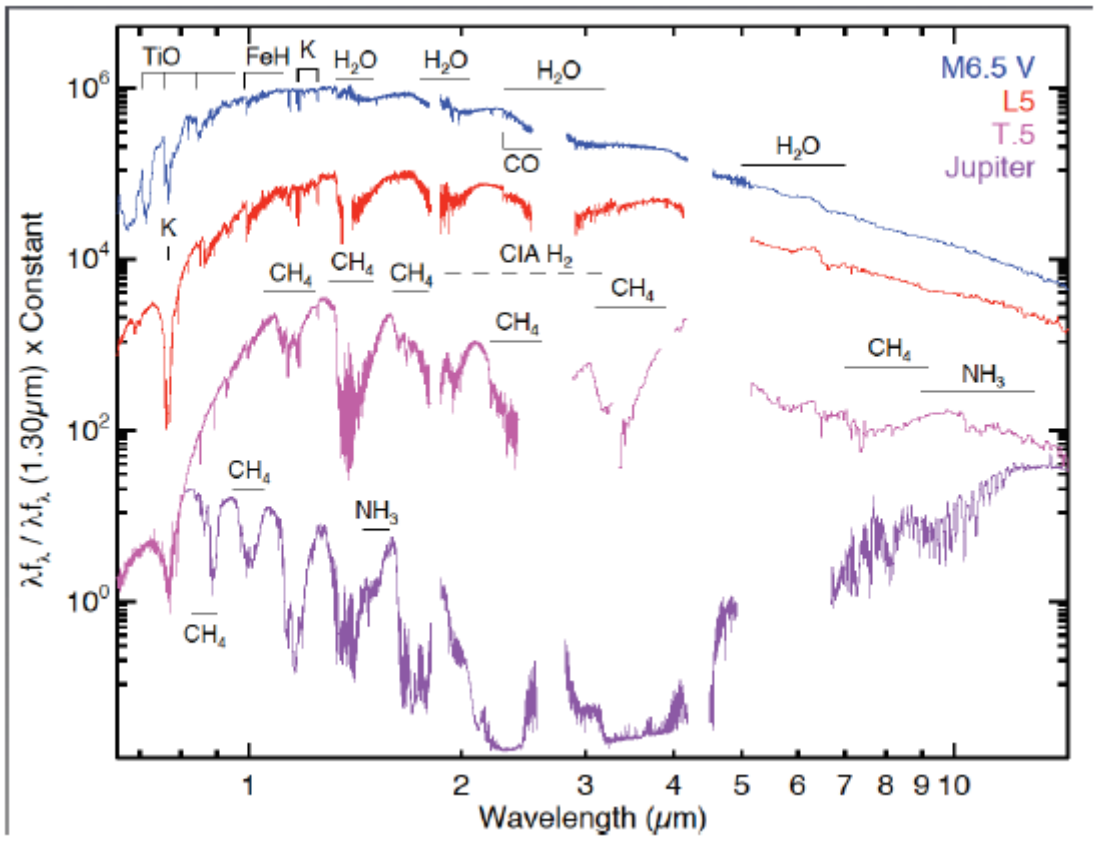


Figure 1.1 Example spectra of a small star (blue), two brown dwarfs (red and pink), and the planet Jupiter (indigo). The dips and spikes in the spectra are called absorption and emission lines and are caused by molecules present in the atmospheres of the objects. Different molecules absorb or emit light at different wavelengths. Note the labels on this graph explaining the sources of dominant spectral features. Figure taken from (Salway 2015), who adapted a figure from (Marley & Leggett 2009).

allowing the estimation of their temperature via Wein's Law, given in Equation 1.1. This law tells us that the temperature of a blackbody, T in Kelvin, is inversely proportional to the wavelength where the object's spectrum peaks, λ_{max} in meters.

$$\lambda_{max} = \frac{0.0029Km}{T} \tag{1.1}$$

Furthermore, much can be learned about the molecular composition of a celestial object by looking at the dips and spikes found in its spectrum, called absorption and emission lines, respectively. The processes of absorption and emission are possible due to the quantization of energy. Electrons within an atom can be in different states with different energies. The difference in energy between two states, call them states n_1 and n_2, is called the energy gap, or E_{gap} (Equation 1.2).

$$E_{gap} = E_{n1} - E_{n2} \tag{1.2}$$

In order for an electron to move up or down to a different state, it must either gain or lose an amount of energy equal to E_{gap}. The energies of the different states of an atom depend on its atomic number Z, where n is the state number (Equation 1.3).

$$E_n = \frac{-13.6Z^2}{n^2} \tag{1.3}$$

This loss or gain of energy occurs when an electron either absorbs or emits a photon with energy equal to E_{gap}. The Planck-Einstein Relation relates the frequency f or wavelength λ of a photon to its energy E, where h is Planck's constant and c is the speed of light (Equation 1.4). Photons can only be absorbed or emitted if their energy E is exactly equal to one of the possible values of E_{gap} for the atom in question. Because of the unique E_{gap} values for different atoms, the presence of these absorption and emission lines in an object's spectrum tells us what atoms the object contains. Lines denoting the presence of various molecules can also be seen in the spectrum of an object, although it is important to note that they are a result of changes in vibrational and rotational energy

levels in the molecules rather than electron transitions. This distinction explains why molecular spectral features span larger ranges of wavelengths than single atomic spectral features could.

$$E = \frac{h}{f} = \frac{hc}{\lambda} \tag{1.4}$$

Because spectra can tell us so much about an object, astronomers have developed a system for classifying stars based on their spectra. Each star is assigned a spectral type, denoted by a letter. The letter is followed by a number that further subdivides the spectral type into sub-types. The spectral types in order from hottest and biggest to coolest and smallest are O, B, A, F, G, K, M, L, T, and Y. The final three letters have been added in the last few years to describe brown dwarfs.

While there is much that remains mysterious about brown dwarfs, there are a few known characteristics. Unlike stars, which are powered by hydrogen fusion taking place in their hot, dense cores, brown dwarfs are too small and cool to facilitate hydrogen fusion. Some brown dwarfs are able to carry out deuterium fusion at the start of their lives, but eventually they cool and the nuclear activity stops. While stars maintain a steady temperature for most of their existence and show a mathematical relationship between their mass and luminosity (brightness), brown dwarfs cool and dim as they age (Burgasser 2008). In this regard, brown dwarfs act more like planets

1.1.2 Discovery and Classification of Brown Dwarfs

While the first brown dwarf was not discovered until 1995 (Rebolo et al. 1995), their existence was theorized years earlier by Shiv S. Kumar (Kumar 1963). He carried out a computational experiment in which he created mathematical models of the formation of stars. Kumar's models showed that, if the new star was below a mass of about 0.09 times the mass of the sun, the star would fail to "ignite," or begin its hydrogen fusion. Kumar initially named these failed stars "black dwarfs," but their name was later changed to "brown dwarf." His project was the beginning of mankind's study of these unique objects.

L and T Dwarfs

In 1999, four years after the discovery of the first brown dwarf, a team led by J. Davy Kirkpatrick searched through data taken by the 2-Micron All-Sky Survey (2MASS) for objects that looked cool enough to be brown dwarfs. The temperature of the object can be estimated from the color, which is found by subtracting the magnitude of the object found using a K telescope filter from the magnitude found using a J filter. This J-K magnitude difference is called the "color." After compiling this list of potential brown dwarfs, Kirkpatrick obtained spectra of these objects using the Low Resolution Imaging Spectrograph (LRIS) located at the W. M. Keck Observatory in Hawaii. The team noticed similarities in 25 of the spectra of these cool objects. Mainly, the spectral shape showed that the objects contained neutral alkali metals and metallic hydrides. The 26th object, a brown dwarf called Gl 229B originally discovered by another team (Golimowski et al. 1998) was the odd-one-out, containing more methane than the other spectra showed. The team decided to group the first 25 objects together into a new spectral type: Type L.

In his paper, Kirkpatrick proposed that brown dwarfs cooler than type L objects be called “T dwarfs” and noted that these brown dwarfs were likely to contain a high amount of methane similar to the odd 26th object, Gl 229B (Kirkpatrick et al. 1999). Further investigation into the T spectral type was performed by Adam Burgasser, Kirkpatrick, and others the following year, confirming the high abundance of methane that is now the main notable feature in T type spectra (Burgasser et al. 2000). For the first time, definite characteristics of these elusive objects had been determined.

Y Dwarfs

Recently, a group led by Michael Cushing discovered and defined the final spectral type: the Y dwarfs. Cushing’s team sorted through data obtained by the Wide-field Infrared Survey Explorer (WISE) and identified seven brown dwarfs that were notably cooler than L and T dwarfs (Cushing et al. 2011). The team discovered that these ultra-cool brown dwarfs could be spectrally differ-

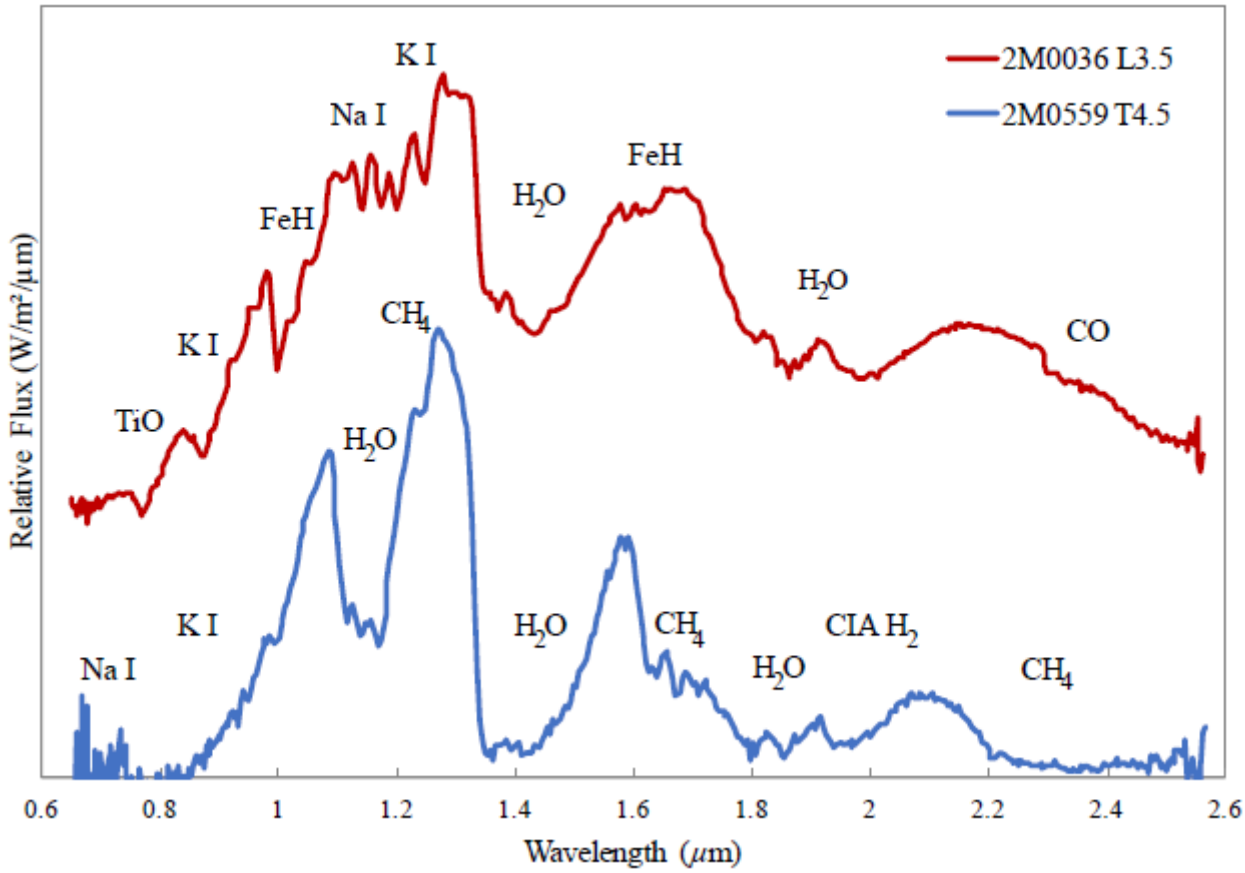


Figure 1.2 Spectra from an L dwarf (red) and a T dwarf (blue). Figure taken from (Farnbach 2017), who used data from (Burgasser et al. 2006) and (Chiu et al. 2006)

entiated from L and T dwarfs by their high abundance of CH_4, H_2O, and NH_3. However, in the few years since Cushing's work there has not been a telescope in space capable of obtaining the high-resolution spectral data needed to more deeply investigate the characteristics of these objects, the coolest and dimmest of the brown dwarfs and therefore the hardest to observe.

1.2 The Importance of Binary Systems

While brown dwarf research has come as far as to classify three types of brown dwarfs and identify prominent molecules found in their atmospheres, there are many questions that remain concerning these unique objects. In this section we address some of these questions, mainly how the current model of stellar formation fails to explain the existence of brown dwarfs and what theories have been proposed to resolve this issue.

1.2.1 Questions About Brown Dwarf Formation

The current well-known model of stellar formation involves the collapse of giant gas clouds found in young areas of galaxies. The mathematical background taught in beginning astronomy courses around the world is that there is some critical mass, called Jean's mass, that determines the stability of a gas cloud of radius R and temperature T. In this equation, k is Boltzmann's constant, G is the universal gravitational constant, μ is the mean molecular mass, and m_H is the mass of a hydrogen atom (Equation 1.5).

$$M_{Jean} = \frac{h}{f} = \frac{5kTR}{Gm_H\mu} \tag{1.5}$$

Once the mass of a cloud surpasses Jean's mass, the cloud collapses in on itself, fragmenting so that there are multiple centers of collapse. These centers of collapse form protostars, which gradually increase in mass and temperature until density, pressure, and temperature are high enough

in each of the protostars for nuclear reactions (particularly the fusion of hydrogen atoms) to ignite, forming a new star.

However, it has been noted that the idea of Jean's mass is more of a back-of-the-envelope calculation, explaining well the formation of larger objects in the universe but failing to account for the formation of smaller objects such as low-mass stars and brown dwarfs. Many questions need to be answered before a more sophisticated model of stellar formation can be developed.

For one, we know that fragmentation occurs in the collapse of gas clouds in order to form small objects, but when we analyze the physics of the collapse we see that there must be some limit to how small of an object can form via fragmentation. The smaller the fragment of the gas cloud, the greater the opacity of the fragment. Greater opacity means increased difficulty for photons to escape the fragment. At some point, photons are unable to escape the opaque fragment and the energy resulting from the gravitational collapse of the fragment is unable to be radiated away. The energy has to go somewhere according to the law of conservation of energy, and in this case it tends to go into turbulent motion of the gas in the fragment. The resulting outward pressure caused by the gas turbulence pushes back against the force of gravity and stops the collapse. In summary, there must be some lower mass limit where fragmentation stops due to turbulence, but we do not know exactly where this limit is, Fragmentation cannot continue indefinitely, so how small of an object can actually be formed in this manner?

Furthermore, a trend has been observed in the relative abundancies of different types of stars. Generally, the less massive spectral types are the most abundant in the universe, explained when we consider our fragmentation model where a gas cloud splits into a few large protostars, a few more medium protostars, and many small protostars. This trend is called the Initial Mas Function (IMF). However, in recent years it has been observed that the IMF drops off dramatically past M dwarfs. There are significantly fewer brown dwarfs than there are M dwarfs, breaking the trend that we see for the other spectral types, and the astronomical community has yet to learn why (Thies et al.

2015).

1.2.2 Proposed Adaptations and the Binary Fraction

Multiple research groups have tried to come up with adaptations to stellar formation theory that would explain how brown dwarfs are formed. Shortly after the discovery of brown dwarfs, Bo Reipurth and Cathie Clarke (Reipurth & Clarke 2001) proposed that brown dwarfs were normal protostars that were somehow catapulted from the center of the molecular cloud before they could accrete enough mass to become fully-fledged stars and begin hydrogen fusion. The next year, Paolo Padoan and Ake Nordlund (Padoan & Norlund 2002) proposed that the requirements of Jean's Mass could be waived when turbulence in the gas cloud triggered gravitational collapse of smaller-mass clouds. Another research duo theorized that the growth of small protostars forming near stars of the hottest spectral types, O or B stars, could be stunted by the radiation pressure of the large number of photons radiating out from the O or B star. They call this process "photo erosion," and propose that this could stop the small protostar from gaining enough mass to become a true star, leaving behind a brown dwarf instead (Whitworth & Zinnecker 2004). Other models exist, and more are on the way. All of these models seem to make sense in some way, but it is difficult to know which if any are true.

A key to supporting or rejecting these proposed adaptations to stellar formation theory is the brown dwarf binary fraction. This term refers to the number of brown dwarfs that exist in binary pairs orbiting each other divided by the total number of brown dwarfs. Each unique formation model implies its own brown dwarf binary fraction, as different methods of formation are more likely to result in the formation of two brown dwarfs versus one lone brown dwarfs. Therefore, if an accurate binary fraction can be estimated we will be much closer to understanding the role of brown dwarfs in stellar formation theory (Gagliuffi et al. 2014).

1.3 The Difficulty of Resolving Binary Pairs

As previously mentioned, finding an accurate binary percentage will help narrow-down the viable options for stellar formation theories. However, there is a physical concept called the Rayleigh criterion that makes visually observing brown dwarf binaries difficult. Brown dwarfs are smaller than your typical star, which thanks to the mathematics of orbits allows brown dwarf binaries to be much closer to each other. On top of this, brown dwarf binaries tend to be abnormally close to each other compared to other stellar binary systems, even when their small masses are taken into account. The smallest angular separation a telescope can resolve is limited by Rayleigh's equation where λ refers to the wavelength of light and D refers to the diameter of the telescope (Equation 1.6).

$$\theta = \frac{1.22\lambda}{D} \tag{1.6}$$

Many brown dwarf binaries have angular separations that fall under this cutoff angle, meaning that they appear as one overluminous brown dwarf rather than as two separate objects. See Fig. 1.3 for a visual of what happens when two objects are unresolved. As technology has improved to allow us to make better and bigger telescopes, we have been able to resolve smaller separation angles. See Fig. 1.4. However, there are still many brown dwarfs that have been flagged as "overluminous" that are thought to be unresolved binary pairs.

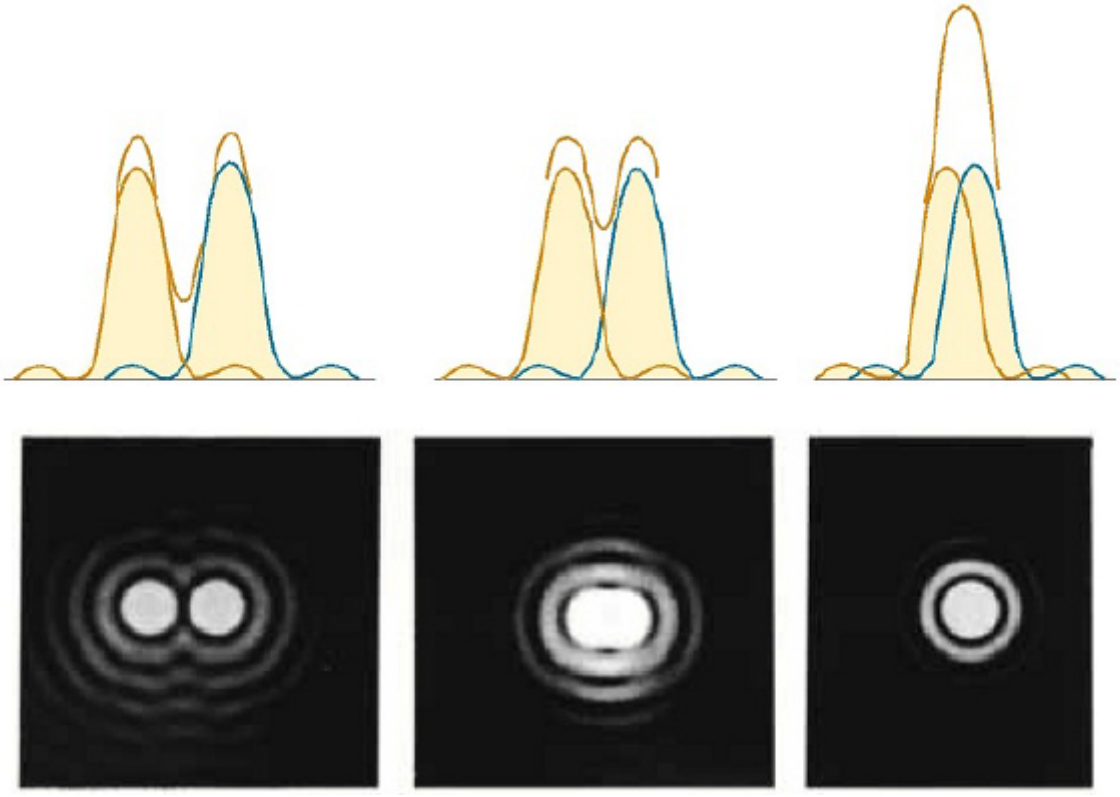


Figure 1.3 The progression of resolved to unresolved telescope images of a pair of objects placed close together. Figure taken from (Gardner 2015), who recreated this image from http://www.kshitijiitjee.com/Study/Physics/Part7/Chapter38/30.jpg

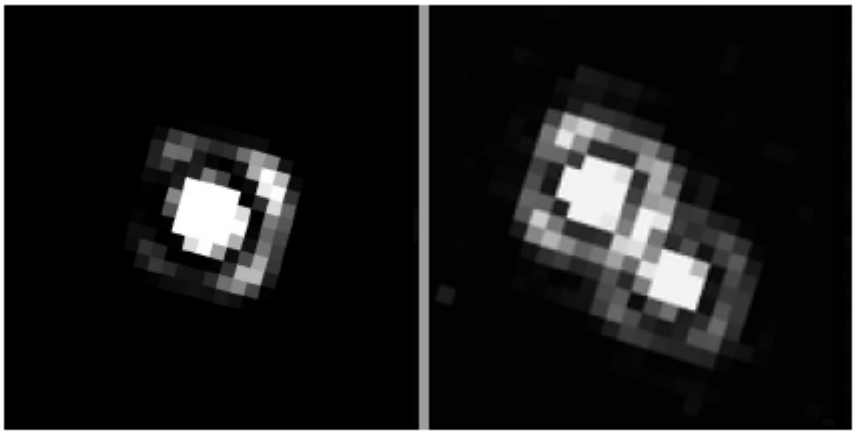


Figure 1.4 The same brown dwarf binary pair imaged in 1998 (left) and 2005 (right) by HST. Note that as technology improved the binary was resolved. Image obtained from (NASA et al. 2009) Credit NASA, ESA,and M. Stumpf (Max-Planck-Institute for Astronomy)

Chapter 2

Spectral Fitting

2.1 Other Methods of Binary Identification

Because of the difficulty of visually resolving brown dwarf binary pairs, a few other methods for observing brown dwarf binaries have emerged in the last few years.

One of these methods is Point Spread Function (PSF) fitting. This involves using the known characteristics of the telescope to determine how light from a distant source should spread out over the CCD, and then comparing the theoretical PSF function to the actual data obtained by the telescope in order to determine if it is more likely that one or two objects are creating the PSF. Other members of my research group have focused their work on PSF fitting (Beus et al. 2020; Matt 2017; Salway 2015)

The second main method is called spectral fitting, and involves statistical analysis of the spectral data of the target object as compared to model spectra. I will focus on the process of spectral fitting, as this is the method my project has revolved around.

2.2 Prior Work in Spectral Fitting

Spectral fitting was first used as a method for identifying brown dwarf binary systems by a group led by Adam Burgasser of UC San Diego. Since then, his work has been continued by his graduate student Dr. Daniella Bardalez-Gagliuffi. Using their work as a starting point, a statistical fitting code has been developed here at BYU iteratively by a chain of students working to improve on each others' results. Before presenting my own work in statistical spectral fitting, I will summarize the work done by others that has been the backbone of my research.

2.2.1 Burgasser's Work and Results

A team headed by Adam Burgasser of UC San Diego (Burgasser et al. 2010) tackled the binary percentage question via statistical spectral fitting in 2010. They compiled 253 brown dwarf spectra of spectral types L3-T8 both from their own observations and published data from the SpeX spectrograph, an instrument on NASA's Infrared Telescope Facility. After removing known binaries and other odd spectra from the mix, the total number of spectra used in the study was 189. Criteria based on the strength of various spectral lines found in each object's spectrum were used to identify 20 possible undiscovered binaries. Each of the candidate binaries was then fit to each of the 170 remaining SpeX spectra as well as the 13,581 different possible combinations of the 170 spectra, made by adding the fluxes together at each wavelength in order to create theoretical binary models.

Chi-squared statistical analysis was used to compare the goodness-of-fit of the binary candidate to each of the 13,751 fit options. See Equation 2.1 for the chi-squared equation used by Burgasser. $C[\lambda]$ refers to the flux of the candidate object at each wavelength λ, $T[\lambda]$ is the flux of the "template" or model object, $\omega[\lambda]$ are the weights used for each data point, α is a scaling constant, and $\sigma_c[\lambda]$ comes from the noise of the candidate object at each wavelength. Note that the fitting was only performed for wavelength ranges $\lambda = 0.95-1.35$, $1.45-1.8$, and $2.0-2.35\mu m$ due to the telluric

absorption bands in Earth's atmosphere.

$$\chi^2 = \Sigma_{[\lambda]}\omega[\lambda](\frac{C[\lambda]-\alpha T[\lambda]}{\sigma_c[\lambda]})^2 \tag{2.1}$$

Due to the fact that the single brown dwarf models were outnumbered by the binary models by a ratio of 72:1, just looking at the chai squared statistics would have resulted in too high of a binary percentage. In order to compensate for this, Burgasser used a one-sided F-test (see Equation 2.2) to find the likelihood that the null hypothesis that the candidate was not a binary could be rejected. If ηSB < 1.34, Burgasser said that the null hypothesis that the object was not a binary could be thrown out with a 99% confidence.

$$\eta SB = \frac{min(\chi^2_{single})/\nu_{single}}{min(\chi^2_{composite})/\nu_{binary}} \tag{2.2}$$

The fitting code was tested on six known binaries and found to agree with the known characteristics of these binaries for 5/6 of the objects. Burgasser's final result was a table of 17 new candidate spectral binary brown dwarfs.

2.2.2 Bardalez-Gagliuffi's Work and Results

Daniella Bardalez Gagliuffi of UC San Diego (Gagliuffi et al. 2014) continued Burgasser's work in the search for spectral binaries. Where Burgasser only included spectra of brown dwarfs of T and L spectral types, Bardalez Gagliuffi also included objects of M spectral types. More spectra had also been added to the SpeX database in-between Burgasser's work and this study. In the end, 815 spectra were chosen as binary candidates and 1110 spectra were used as models for the fitting process. The 815 candidates were further narrowed down using the same criteria employed by Burgasser, until a list of 35 were chosen.

This study was unique in that the composite models were generated so that binary pairs were always made of a M or L dwarf and a T dwarf, rather than looking at all of the possible combinations of template models. The rest of the methods and procedures used were similar to those used by

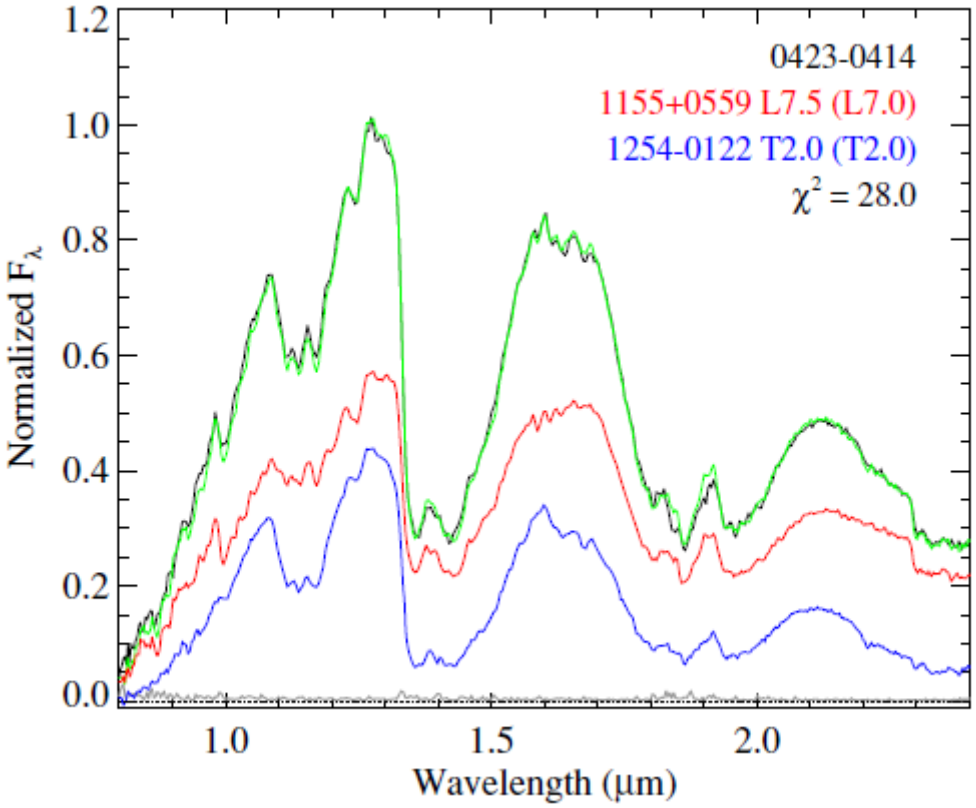


Figure 2.1 One of Burgasser's final plots depicting the best fit for brown dwarf 0423. The black line is the actual spectrum of the object. Red and blue are two real brown dwarf spectra that, when added together, create the green composite spectrum that is the best binary fit for 0423. Taken from (Burgasser et al. 2010)

Burgasser. In the end, 11 new binary candidates were found with the help of the fitting code. Four candidates from previous studies were also confirmed to be binaries. The team also learned that blue L dwarfs can contaminate spectral fitting when they are included in the collection of objects used as models.

2.2.3 Past Work Done at BYU

Similar spectral fitting work has also been carried out here at Brigham Young University, directed by Professor Denise Stephens (Stephens et al. 2009).

Stephens' research assistant Taran Esplin used statistical spectral fitting to characterize the over-luminous brown dwarf 2MASS 0559 (Esplin 2010). Esplin developed code in the program MatLab to fit model brown dwarf spectra generated by D. Saumon and Mark Marley (Saumon & Marley 2008) to the observed spectrum of 2MASS 0559. Esplin's results showed that 2MASS 0559 was likely a binary system, supporting the theories of those before him that overluminosity is at least some times indicative of more than one object being present.

Spectral fitting had become such an important method of learning more about brown dwarfs that Katrina Wright, another of Stephen's students, devoted her senior thesis to rewriting Esplin's MatLab code in the more widely used language C++. Wright also generalized the code so that it could be used on any brown dwarf rather than just 2MASS 0559. Wright's code was used as a basis moving forward (Wright 2015).

Another of Stephens' research students, Leanne Farnbach, continued the spectral fitting efforts. Farnbach identified binary candidates and found spectra for each object on the Spitzer Heritage Archive. Like Burgasser, Farnbach included only T and L dwarfs in her fitting. Like Esplin, she used models produced by D. Saumon and Mark Marley (Saumon & Marley 2008), but in addition to these she also included more recently updated models from another paper (Morley et al. 2012). The models had much higher resolution than the Spitzer data, so Farnbach used a smoothing code

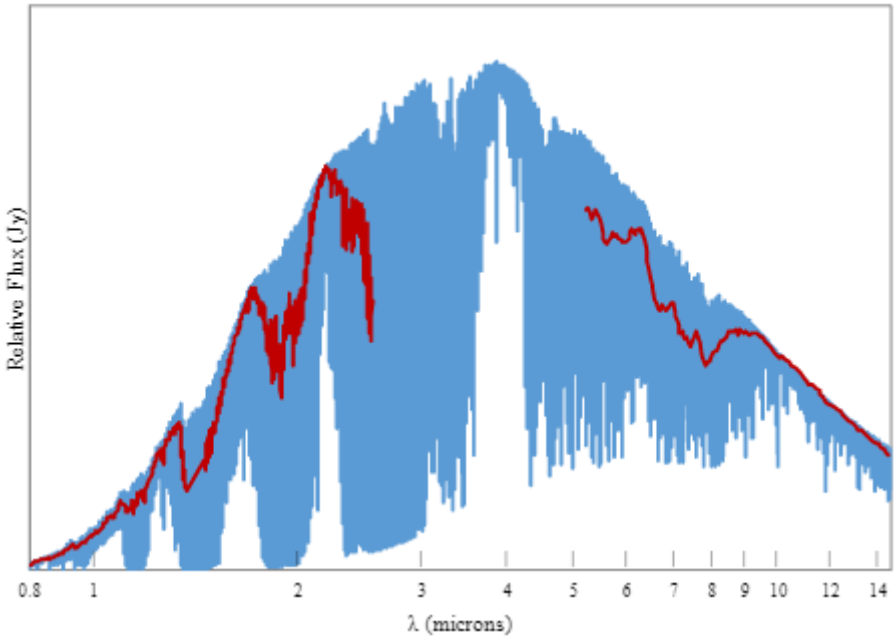


Figure 2.2 A plot depicting an original model spectrum (blue) and the spectrum after being smoothed (red) (Farnbach 2017)

to match the resolution of the models to that of the real data (See Fig. 2.2). After running the statistical spectral fitting code on her six binary candidates, Farnbach found all 6 of them to fit better with binary models than with single models. Two of the objects tested were known binary pairs used to verify the capabilities of the fitting code. An interesting result of Farnbach's project was her observation of the failure of the models to match certain spectral features of the real data. Farnbach was able to identify a few specific molecules that the models failed to accurately represent, telling us that the current models for brown dwarfs lack important molecular line lists and are thus inaccurate at the associated wavelengths. Farnbach's final conclusion was that her results supported the effectiveness of using spectral fitting to identify unresolved binary pairs (Farnbach 2017).

The most recent work completed by Stephens' group was done by her student John-Michael Eberhard. Eberhard's peers used data taken by the Gaia telescope to identify 117 brown dwarfs that appeared to be brighter than expected and therefore were likely to be unresolved binaries. Eberhard

was able to access spectral data for 26 of these objects from NASA's Spitzer Space Telescope (SST) and from NASA's ground-based InfraRed Telescope Facility (IRTF). He used the same models as Farnbach, and found that the code labeled 12 of his objects as binary to a 90% or greater confidence level (Eberhard 2020)

Chapter 3

Improving the Code

3.1 Writing the Pipeline

As previously mentioned, the spectral fitting code used by our team was a conglomerate of pieces written by different coders in different languages for different purposes. Because of this, the process of running the code involved many steps and various files of code. My first task, beginning from Eberhard's work, was to simplify and speed up the code.

3.1.1 The Old Code

In order to run the old code, spectral data first had to be obtained from two different sources in order to cover the desired wavelength range of 0.6 – 14 microns. Data from 0.6-2.5 microns, taken by NASA's SpeX Spectrograph (part of the InfraRed Telescope Facility) was obtained from the SpeX Prism Library, and was given in normalized units. Data ranging from 5-14.5 microns, taken with the InfraRed Spectrograph (IRS) on the Spitzer Space Telescope, was obtained from the InfraRed Science Archive (IRSA), and was not normalized. In order to use both data sets in the same spectral fitting process, the units needed to be the same. The Spex prism data was fed into a python code

that took in the magnitude values of the data set and un-normalized the data to put it in the same units as the Spitzer data.

Next, the models had to be re-generated for each new object. A Perl code calling a C++ regridding code rebinned and smoothed the models to match the wavelength grid of the main object's data set. As there were over 1400 models and they all had extremely fine resolution, this step was very lengthy. After preparing the spectral data file and regridding the models, the next step was to create a parameters file and a very specifically-named folder containing all of the needed files. Then, the binary fitting C++ code was called. This part of the code was well-written and fairly efficient considering the sheer amount of binary models it generated and examined. However, because there were so many models it still took a long time to run. On top of the large number of models, because the resolution of the models was matched to the resolution of the target object each time the code was run, the code took even longer than usual for target objects with high resolutions.

The next step was to run a Perl code that called a single fitting C++ code on each of the models. This was yet another lengthy process due to the large number of models, although the single fitting code was once again very well written. The sheer amount of models was the culprit again. Lastly, a python code needed to be run to sort and compare the binary and single fits and provide statistical data such as the best chai squared values and the η value. This code also generated multiple graphs illustrating various fits and notable features of the data sets. In order for the η value to be useful, a final and separate python code needed to be run that calculated the degrees of freedom for the data set and determined to what percentage of confidence the null hypothesis that the object was not a binary pair could be rejected.

3.1.2 First Things First: One Language

The first problem to fix was the fact that a user running the code would be required to interact with programs written in three different languages: Python, C++, and Perl. The initial plan was to

re-write everything in Python. However, as a main goal of my project was to improve the speed of the code, it was decided that C++ was still needed for the generation of the composite models and both the binary and single fitting programs. Thus, I kept the original C++ binary and single fitting codes, but everything else used in the new code is written in Python.

I rewrote both of the Perl codes in Python, a much more familiar and user-friendly language for most young astronomy students. I also rewrote the regridding code used to prepare the models in Python. As our new plan for the code did not require regridding the models over again each time the code was run, I was less concerned about the speed of the code and more concerned about ease of use, therefore Python was a better choice than C++.

3.1.3 A Single Command

The next step was to make the code more user-friendly. The old code had so many pieces that even after a year of working with it I still need to reference notes to remember the order to run each of the programs and how exactly to name files and call commands. The old code also required users to alter lines inside code files to match the name of the object and directory. I wanted the new code to do all of the folder making, file naming, and directory traversing on its own. I also wanted the new code to be able to be run with a single command in the terminal.

The final product was a single python script called "pipeline.py" (see App D) that can be run with the simple command "python3 pipeline.py ####" where #### represents the first four digits in the name of the target star. This single script regrids the target star's data file in order to match the models, calls the binary fitting C++ script, calls the single fitting C++ code, and then calls a final python script that sorts the fits, calculates the η value, generates multiple plots showing the fits, and calculates how statistically confident we can be that the null hypothesis is rejected.

I also included a line at the beginning of this pipeline function that runs the Makefile for the C++ programs. I included this line because the C++ programs included so many files, headers, and

libraries that manually compiling the code each time something was changed was a great source of frustration. I foresaw this issue arising any time a new user tried to run the code on their own machine in the future. Now, as long as a new user downloads the folder containing the complete set of files and downloads the external libraries per the instructions in the Workflow document (see App. A), the pipeline function will do the rest to ensure the code is compiled and ready to go.

3.1.4 Multiple Objects

While running the code on multiple objects to test its accuracy, I decided that another change would be very useful: I wanted to be able to feed the code a list of objects to best-fit rather than having to manually call the code for each object. I had a list of 26 objects to run the code on and found it much easier to get the code running on all of them at once and be able to walk away and work on something else while it ran rather than having to babysit the computer the entire time.

This addition was fairly simple. I took the pipeline code and adapted it to work as a function inside a larger code called "WTII.py" (See App. E). The pipeline function still takes the first four digits of the target object's name as an input and completes all the same processes. However, making the pipeline script into a function allowed me to call it multiple times from a parent code. The parent code takes a text file containing a list of objects as its only input, parses through the text file, and then runs the pipeline function on each object in turn.

Realizing quickly that running the code on multiple objects at a time resulted in a lot of clutter (too many output graphs and files to keep track of), I also added another element to the pipeline function that created a folder for each object and nicely stored all of the files and graphs generated by the code in the corresponding folder.

3.2 Switching to Real Data

Now that the code was simpler and more efficient to run, my next major project was to change from using Saumon and Marley's models in our fitting process to using real SpeX data as our spectral templates. As previously noted, in her own senior thesis work, Leanne Farnbach's results showed that the models failed to include certain characteristics seen in real brown dwarf spectra, namely the strong methane bands in T-dwarfs (Farnbach 2017). It was decided that using real data for the fitting process would remedy this problem and hopefully result in better fits.

3.2.1 Choosing the Template Objects

The first step was to identify which SpeX data files would be best to use as spectral templates, replacing the models used in the past. The SpeX Prism Library contains data for over 800 objects with magnitudes. However, the data files remain unpublished for hundreds of these objects.

Another member of my research group, Amber Berry, sorted through all of the known brown dwarfs with available spectra. She sorted these brown dwarfs by spectral type and then calculated the standard deviation of J-K magnitudes for each spectral type first in the optical and then in the infrared. She used the resulting data to identify the brown dwarfs that appeared to have the most well-behaved colors and luminosities for each spectral class. Specifically, modeling our strategy after a paper written by Bardalez-Gagliuffi (Gagliuffi et al. 2014), we decided to keep all of the objects that had colors that fell within two standard deviations of their spectral index. In this way, Amber was able to gather together a collection of brown dwarf objects that were most representative of each spectral class to be used as the spectral templates in our fitting routine.

This sorting step was important for a few reasons. First, if we included binary objects in our template sample, it would be likely that the fitting program would just identify a "single object fit" of our target star and one of the binary models rather than correctly finding a true binary fit. Second,

in her 2014 paper Bardalez-Gagliuffi reports that including L-dwarfs that were too blue resulted in poor fits. She called these objects "contaminants," and ended up solving the problem by rejecting any template candidates that fell outside of two standard deviations for their spectral index.

Because only one or two spectral files were available for some of the outlying spectral indices, a few overluminous objects slipped through the cracks during this sorting process. Because of this, the initial test fits were poor. Noticing the problem, I went through and manually removed any objects that were in either Eberhard's (Eberhard 2020) or Bardalez-Gagliuffi's (Gagliuffi et al. 2014) lists of likely binaries.

3.2.2 Preparing the Spectral Templates

The main time dump for the old code was the model preparation step. As the old code was designed to work for data files coming from any telescope with any resolution, the makers of the code decided to regrid all 1400 models to match the resolution and wavelength grid of the target data file each time the code was run.

Using SpeX data for both the target objects and the template objects made this step unnecessary. All of the SpeX data files have very similar resolutions and wavelength ranges. Nearly all of the available data files range from about 0.65 microns to about 2.5 microns, with a wavelength step size of about 0.001 microns. I chose a standard grid of 0.66 to 2.50 microns with a step size of 0.001 and wrote a python code to regrid all of our template objects to this standard. The regridding code can be found in the appendix (See App. B). These regridded models will work for all of the data files we will be fitting with this code. A subfunction is called in the pipeline function that regrids the data files to this same, standard grid. Thus, each time the code is run only one file has to be regridded rather than having to regrid all of the template data files every time.

3.2.3 Adapting the Code to the Spectral Templates

A few adaptations needed to be made to the rest of the code to work with the new spectral templates. First, some of the subfunctions used in the pipeline that originated from the old code expected a specific naming convention for the template files. This was a quick fix.

Next, the parameters used for the old models did not work well with the newer template objects. The Saumon and Marley models (Saumon & Marley 2008) had flux values in SI units that were calculated as if the observer were a fixed distance from the surface of the star, while the data files used for the old code had much smaller flux magnitudes as they were taken from further away. The parameters file takes in a maximum and minimum value for the scaling constant used in the spectral fitting, as well as a starting value and a minimum tolerance. The old parameters did not work with the new set of template objects and data files that were so much closer in flux magnitude. The code would time out and crash after exceeding its allowed number of iterations in its attempt to find a god fit within the tolerance. With trial and error I discovered a new set of parameter values that worked for each of the test objects. I also increased the allowed number of iterations in the fitting code to allow for higher accuracy. As the new code uses far fewer spectral templates than the old code, the slight increase in fitting time caused by the increase in allowed iterations was not a concern.

Lastly, the old code included functions to apply different telescope filter profiles to the template files corresponding with the telescope used to take the data files. As both the template and data files were taken using the same telescope, this step was no longer necessary and was removed from the code. All of the major python files I wrote and/or adapted for the code can be found in the appendix. Pieces of code written by others are cited in the beginning comments. The C++ files for the Binary and Single Fitting programs can be found on the data drive of the BYU astronomy computers. As I did not change the C++ code myself, I will not include them in this paper. A Workflow document explaining how to run the code is also included in the appendix (See App. A).

Chapter 4

Results and Conclusions

4.1 Results

The two main objectives of this project were to (1) make the code faster, simpler, and more user-friendly and (2) be able to reasonably replicate the results obtained by previous spectral fitting projects. Replicating previous results was a critical goal because it allowed us to ensure that the new iteration of the code still functioned the way it should.

4.1.1 Comparison to Previous Results

Two of the most exhaustive lists of positively-identified spectral binary objects came from Eberhard (Eberhard 2020) and Burgasser (Burgasser et al. 2010). First, I began by comparing my results to Eberhard's. In his paper he identifies seven spectral binaries with above a 99% confidence level, four above a 95% confidence level, and one object above a 90% confidence level. These objects and their corresponding confidence levels are listed in Table 4.1

I decided the best way to test the new code was to run it on these 12 objects and compare the results. I ended up only testing 10 out of the 12 because two of the objects did not have normalized

Name	η_{sb}	Confidence Level
2MASS 05591914-1404488	1.554	99%
SDSS J042348.57-041403.5	1.549	99%
2MASSW J0320284-044636	1.520	99%
2MASS 14313097+1436539	1.449	99%
DENIS-P J225210.73-173013.4	1.355	99%
2MASS 20282035+0052265	1.287	99%
SDSS J141624.08+134826.7	1.257	99%
SDSSp J125453.90-012247.4	1.323	95%
Gl 417 BC	1.244	95%
2MASS 09153413+0422045	1.181	95%
2MASS 20360316+1051295	1.161	95%
WDS J15200-4423A	1.146	90%

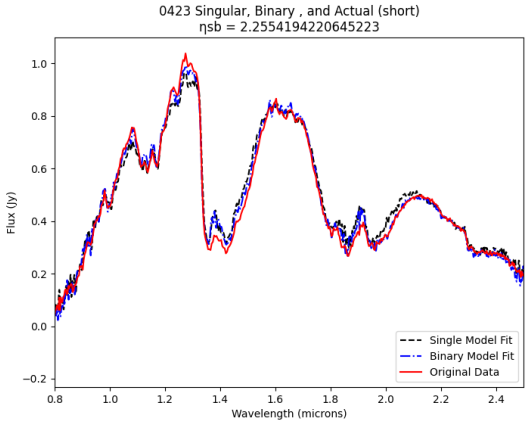
Table 4.1 Eberhard's table of identified spectral binary objects. Taken from (Eberhard 2020)

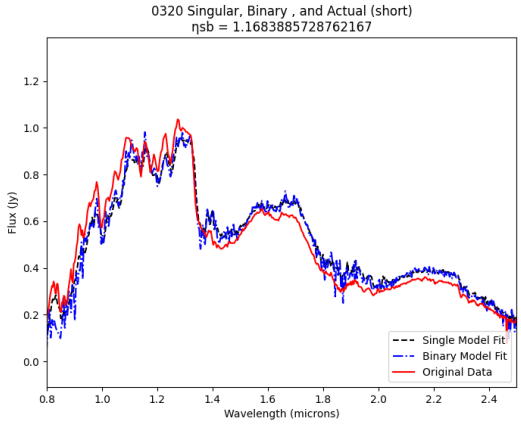
Name	η_{SB}	Confidence Level
SDSS J042348.57-041403.5	2.255419	99%
2MASSW J0320284-044636	1.168389	99%
2MASS 14313097+1436539	1.198202	99%
DENIS-P J225210.73-173013.4	1.673414	99%
2MASS 20282035+0052265	1.814395	99%
SDSS J141624.08+134826.7	1	<90%
GI 417 BC	1.072355	95%
2MASS 09153413+0422045	1.567773	99%
2MASS 20360316+1051295	2.959519	99%
WDS J15200-4423A	1.286718	99%

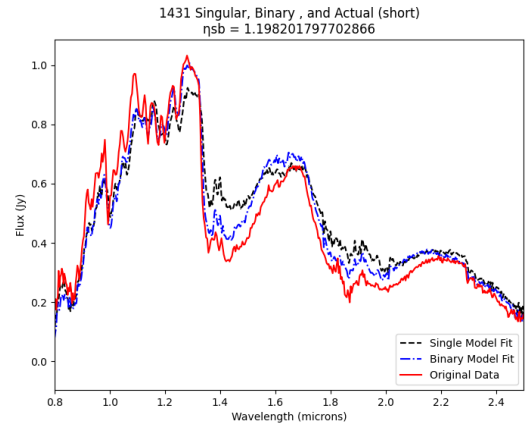
Table 4.2 Results for 10 of Eberhard's 12 objects. Note that all but one was successfully identified as a binary object by the new code.

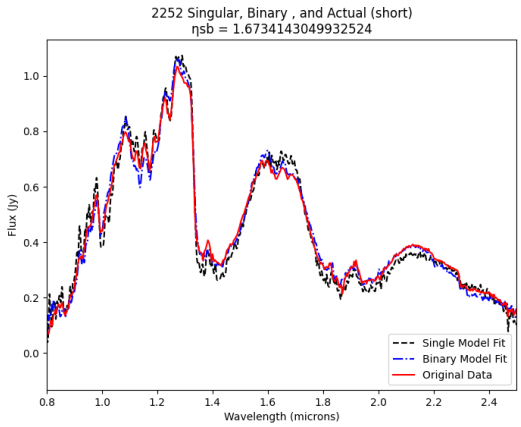
data as the new code requires. The other two objects could be tested in the future if a normalization code was written.

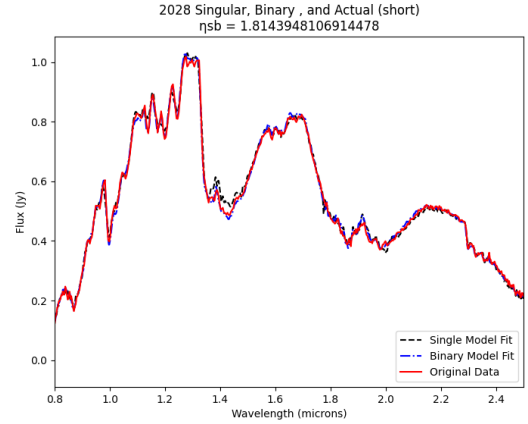
Excitingly, 9 of the 10 objects were best-fit with binary models to above a 90% confidence level. Note that the meaning of the η values obtained using the new code is not the same as that of the η values from Eberhard's work. This is because I used a different grid for the spectral data, and the η calculation includes the number of data points used in fitting. However, the confidence percentages have the same meaning. My results are given in Table 4.2. As the new code has significantly fewer models to use for the fitting process than did the old code, I did not find it unexpected that one of the objects was not confidently fit to binary models. See Fig. 4.1 for the graphs of the best fits.

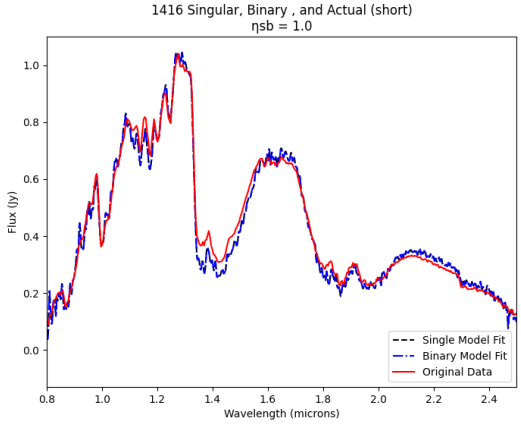

0423 Singular, Binary , and Actual (short)
ηsb = 2.2554194220645223
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data


0320 Singular, Binary , and Actual (short)
ηsb = 1.1683885728762167
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data


1431 Singular, Binary , and Actual (short)
ηsb = 1.198201797702866
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data


2252 Singular, Binary , and Actual (short)
ηsb = 1.6734143049932524
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data


2028 Singular, Binary , and Actual (short)
ηsb = 1.8143948106914478
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data


1416 Singular, Binary , and Actual (short)
ηsb = 1.0
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data

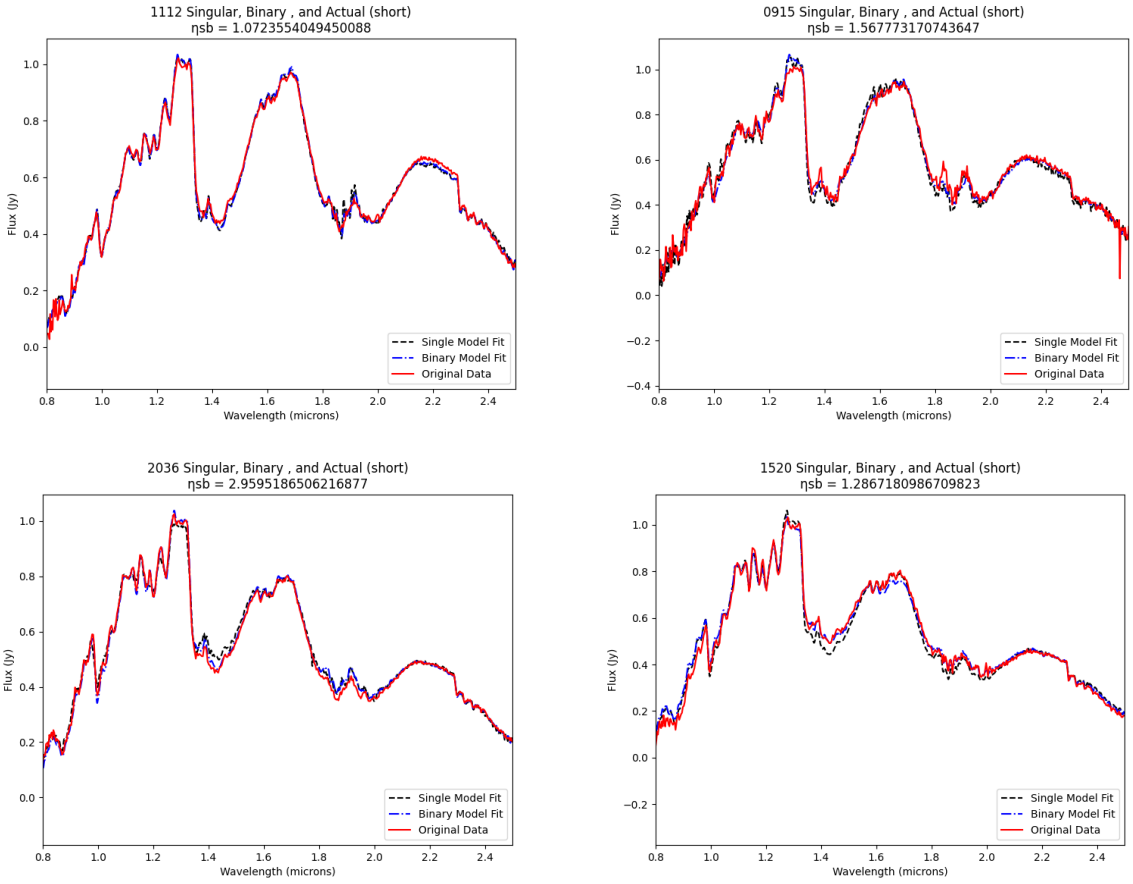


Figure 4.1 Plots of the best fits for each of Eberhard's 10 objects that I tested. Each plot shows the original data, the binary best fit, and the single model best fit.

For my second test, I decided to run the code on the same objects used by Burgasser in his 2010 paper. Burgasser identified 12 strong binary candidates and 8 weak binary candidates. From these 20 candidates he identified 17 spectral binaries. I tested the code on the 13 of these 20 objects for which I had data in the correct form. My results matched Burgasser's for all except one of the objects, SDSS J075840.33+324723.4. I found this object to be a binary with >99% confidence level while Burgasser found a 1% confidence level. However, in his comments for this object, Burgasser notes that this object is a "near-clone to SDSS J1254-0122," which explains the very low 1% confidence level showing that 0758 fit very closely with a single model. I did not include 1254 as one of my model objects as this was one of the objects I tested the code on, therefore it was not available to be chosen as the best single model for 0758. It is interesting to note that, with 1254 removed from the fitting options, the best fit for 0758 is indeed the binary fit. See Table 4.3 for a comparison of my results to Burgasser's results and Fig. 4.2 for the resulting best-fit plots.

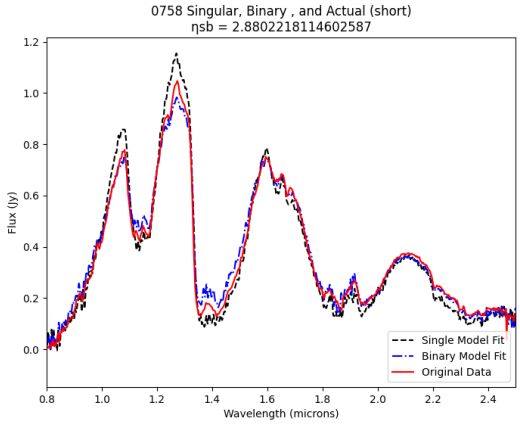
4.1.2 Speed of the Code

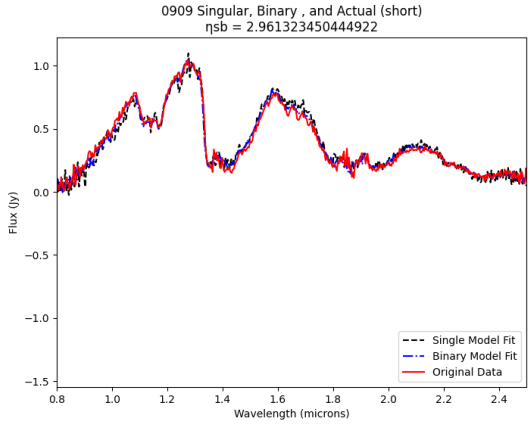
The second goal of this project, speeding up the binary fitting process, was also found to be a success. A big contribution to this increase in speed was the fact that there are fewer data files available to use as fitting templates now that we have switched to using real data rather than the models. Also, the real data has a much lower resolution than the models, reducing the time needed to regrid the models. Where regridding the models could take close to 8 minutes using the old code, the new process required to prepare the models consistently takes less than 10 seconds (Again, I point out that this was due to the change in the number of models, not necessarily because of the regridding code being more efficient). On top of this, the new structure of the code does not require regridding the models before each object is run. Rather, the same set of models can be used for all objects with the standard SpeX wavelength grid of about [0.66 microns, 0.25 microns].

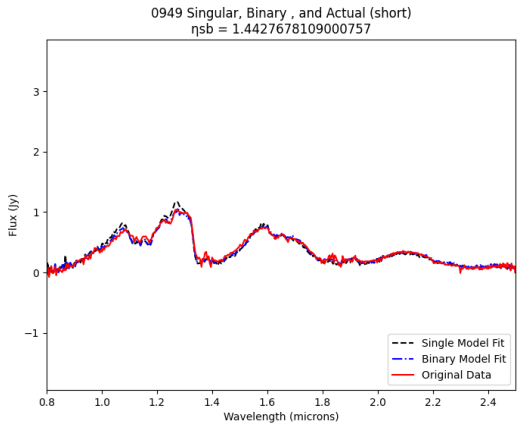
The pipeline function was the main success of this part of the project. Rather than having to run

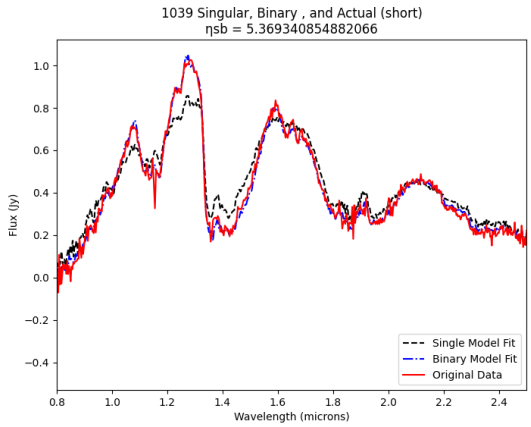
Name	η_{SB}	Confidence Level	Burgasser
SDSS J103931.35+325625.5	5.369341	99%	99%
2MASS J11061197+2754225	3.610466	99%	99%
2MASS J13243559+6358284	4.802163	99%	99%
SDSS J141530.05+572428.7	6.571717	99%	99%
2MASSI J1711457+223204	1.553529	99%	99%
2MASS J21392676+0220226	4.858947	99%	99%
SDSS J151114.66+060742.9	2.117856	99%	99%
SDSS J143945.86+304220.6	1.732264	99%	99%
SDSS J090900.73+652527.2	2.961323	99%	99%
SDSS J120747.17+024424.8	2.374053	99%	99%
SDSS J205235.31160929.8	2.253652	99%	99%
2MASS J094908601545485	1.442768	99%	99%
SDSS J075840.33+324723.4	2.880222	99%	1%

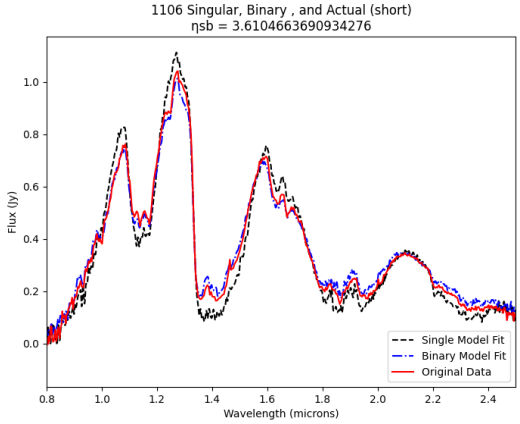
Table 4.3 Results for 13 of Burgasser's 20 objects. Note that my results match Burgasser's for all but one of the objects.

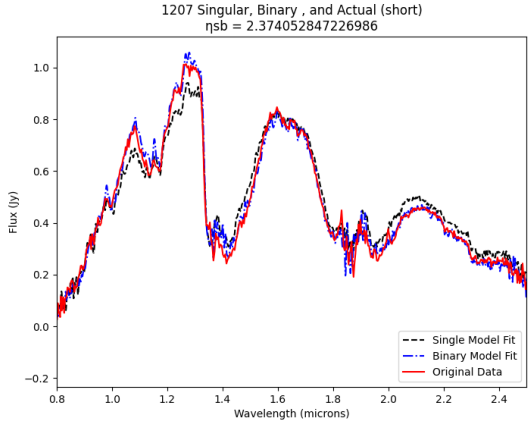

0758 Singular, Binary , and Actual (short)
ηsb = 2.8802218114602587
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data

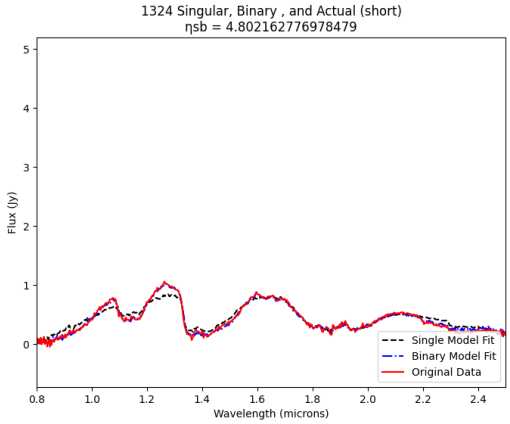

0909 Singular, Binary , and Actual (short)
ηsb = 2.961323450444922
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data

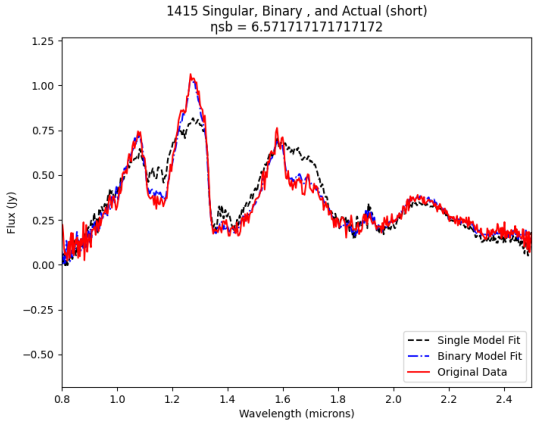

0949 Singular, Binary , and Actual (short)
ηsb = 1.4427678109000757
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data

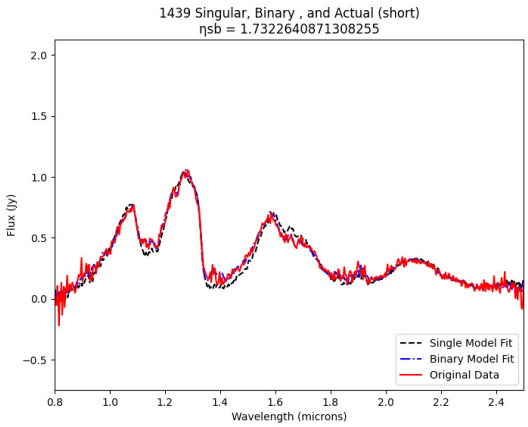

1039 Singular, Binary , and Actual (short)
ηsb = 5.369340854882066
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data

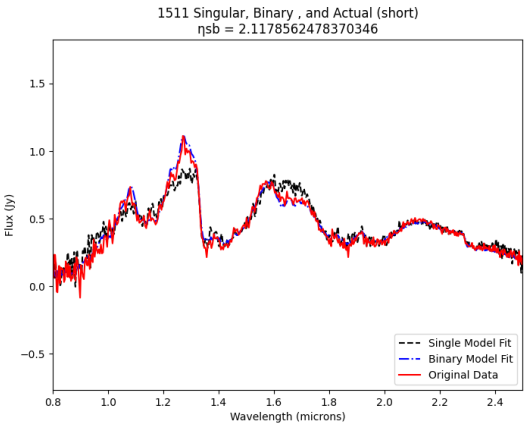

1106 Singular, Binary , and Actual (short)
ηsb = 3.6104663690934276
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data

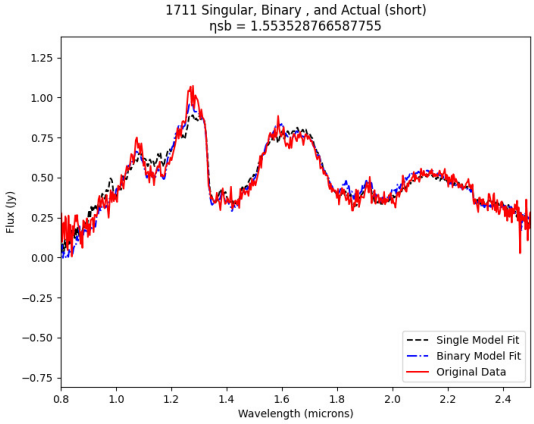

1207 Singular, Binary , and Actual (short)
ηsb = 2.374052847226986
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data

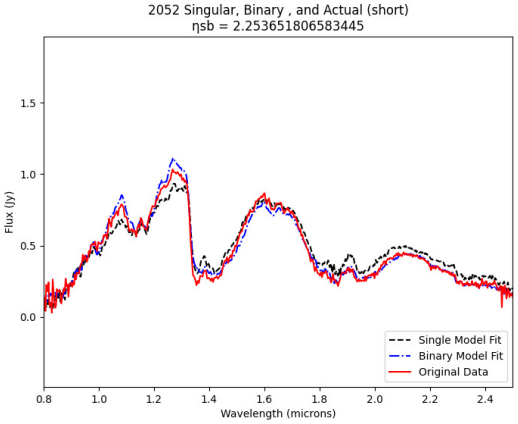

1324 Singular, Binary , and Actual (short)
ηsb = 4.802162776978479
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data


1415 Singular, Binary , and Actual (short)
ηsb = 6.571717171717172
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data


1439 Singular, Binary , and Actual (short)
ηsb = 1.7322640871308255
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data


1511 Singular, Binary , and Actual (short)
ηsb = 2.1178562478370346
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data


1711 Singular, Binary , and Actual (short)
ηsb = 1.553528766587755
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data


2052 Singular, Binary , and Actual (short)
ηsb = 2.253651806583445
Flux (Jy)
Wavelength (microns)
Single Model Fit
Binary Model Fit
Original Data

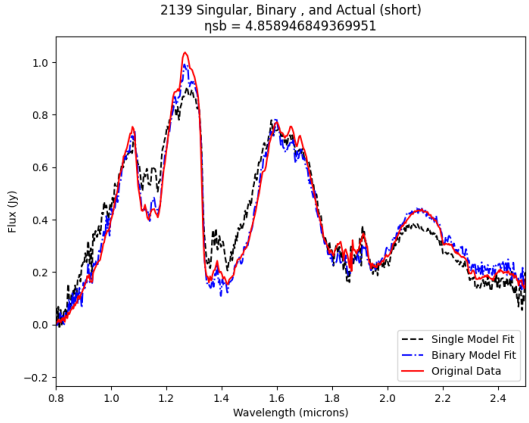


Figure 4.2 Best-fit plots of the 13 of Burgasser's objects I ran the code on. The plots include the original data, the best binary fit, and the best single model fit.

multiple programs in three different coding languages, many of which required the user to change lines in the scripts to match the name of the object each time, the code can now all be run from a single python command in the terminal. Where the process to fit a single object could take upwards of an hour and had many opportunities for user error in the past, the new code takes about two minutes per object. The option to run an entire list of objects at a time, again with only a single command, also greatly increased the efficiency of the code. Users can start a list of objects running in the background and then work on something else while the code takes care of the heavy lifting, where in the past running the code was very user-involved.

4.2 Conclusion and Future Work

In summary, the two objectives of this project were to (1) speed up the code and (2) be able to successfully replicate the spectral fitting results of others in the field.

The new code is significantly faster. Switching to using real data rather than the models is faster and removes multiple steps from the fitting process, on top of providing more meaningful results.

I found it easier to interpret binary fits when the best-fit objects were given as real objects whose magnitudes I could look up rather than the models whose meanings were more obscure. Where it used to take up to an hour to run the old code on a single object, I can now run the code on an entire list of objects using a single command, with each object only taking about two minutes. This will make the code much easier for new students to learn to use. When the James Webb Space Telescope (JWST) launches and begins to obtain new brown dwarf spectral data, this new code can be easily adapted to efficiently search for spectral binaries that would otherwise remain hidden in the new data.

The new code reliably matches results from previous iterations of the code and from work done by others in the field. When tested against 10 objects from Eberhard (Eberhard 2020) and 13 objects from Burgasser (Burgasser et al. 2010), the new code successfully identified 21 out of the 23 objects labeled as spectral binaries by the two authors. In the future, it would be useful to test the code against not only the objects positively identified as binaries by others, but also against known single brown dwarfs to verify that the code reliably fits single objects as well. The code could also be run on the rest of the 117 overluminous objects identified in Eberhard (Eberhard 2020) to search for more spectral binaries now that the code is so much less tedious to run.

Other future improvements to the code could include a step checking to see if the input file includes errors or not, an automatic conversion any data files given in angstroms to microns, and the creation of a second set of models to be used for any data files with a range smaller than the [0.66 microns, 0.25 microns] grid used in this paper.

Appendix A

Workflow: Updated Instructions for Using the Code

Abbreviated Binary Fitter Pipeline Workflow

Requirements:

-Python 3

-The C++ files and Python files need to all be in the same folder

-Be sure that you have the makefile in the same folder as well.

-The folder named "Models" needs to be present as well-
this contains the spectral models the program uses for comparison

-you will need files called ***ErrFinal.txt and ****Params.txt where **** stands for the first four digits of the star's name.

DO ONCE BEFORE RUNNING THE CODE FOR THE FIRST TIME:

-inside the Models folder is a file called Models.txt. This contains the path to each of the model files in that folder. You will want to update the path to reflect your own directory. If you run from a school computer this doesn't super matter, the code will just use the model files from my folder, but if you try to run this on your own computer without updating the path files the ubuntu demon will get angry.

-the params file also has a path to the models folder. You will need to update that path as well. Those should be the only 2 things you have to adjust

How to run:

Create a text file containing the 4-digit names of the objects you wish to run the code on.

Then, call WTII.py in the terminal with the name of the text file as your only input.

EX: >python3 WTII.py objects.txt

What it does:

1)regrids the input spectrum to match the grid of the models and puts into file called ****ReadyToRockAndRoll.txt.

2)runs the binary fitter c++ code on ****ReadyToRockAndRoll.txt

3)runs the singular fitter c++ code on ****ReadyToRockAndRoll.txt

4)compares the fits

5)outputs plots and files showing the best binary and singular fits
6)places all relevant files and plots in a folder made for the object
7)repeats for each object

Checking the output:
If you want something to compare the output to, John Michael's result files are all in his johnme folder under brown dwarf spectra. For example, go to his Spectra0004 folder to see his results for 0004

Note:
The spectral template files in the Models folder are regridded to the standard grid
that is closest to the majority of the data files in the SpeX Prism Library. If you wish to fit an object that has a smaller range of wavelengths, you might need to create a new folder of models using the massConvert2pt0.py code found later in this appendix. If you do this, be sure to create a txt file containing paths to each of the new models and place it in the new Models folder. See the existing Models folder for reference. Else, the existing models should be sufficient and you should not need to mess with them.

Appendix B

massConvert2pt0.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jul  8 18:29:55 2020

@author: sanan
"""

import sys
import time
import os
import MassSmoothConvert2pt0
import shutil

#This is the main code used for converting a list of model
```

```
#spectra to a matching wavelength grid.

tic=time.perf_counter()

if __name__== '__main__':
    print ("\nThis␣file␣is␣being␣executed.")

else:
    print ("\nThis␣file␣is␣being␣imported.")

if len(sys.argv)!=3:
    print("Something␣is␣wrong␣with␣the␣arguments␣list.
␣␣␣␣Evaluation␣aborted.")

else:
    inFolder=sys.argv[1]
    outFolder=sys.argv[2]
    modelNumber=0

    inFileNames=[]
    outFileNames=[]

    move=shutil.move
    smoothConv=MassSmoothConvert2pt0.__main__
    currentDir=os.getcwd()
```

```
with os.scandir(inFolder+'/') as entries:
    for entry in entries:
        name=entry.name
        print(name)
        inFileNames.append(name)
        move(f'{currentDir}/{inFolder}/{name}',
        f'{currentDir}/{outFolder}')

os.chdir(outFolder)
currentDir=os.getcwd()

for n in range(0,len(inFileNames)):
    newName=f'two{inFileNames[n]}'
    f = open(newName, "w")
    f.close()
    outFileNames.append(newName)
    modelNumber=modelNumber+1

for n in range(0,len(inFileNames)):
    smoothConv(inFileNames[n],outFileNames[n])
    move(f'{currentDir}/{inFileNames[n]}',f'../{inFolder}/')


with os.scandir(f'{currentDir}/') as entries:
```

```
        for entry in entries:
            name=entry.name
            print(name)
            inFileNames.append(name)


toc=time.perf_counter()
print(f"This code took {toc-tic:0.8f} seconds")
```

Appendix C

MassSmoothConvert2pt0.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jul  8 18:38:17 2020

@author: sanan
"""

import Spectrum
import Converter
from scipy import interpolate
import numpy as np
import os

#this is the smoothing/regridding function
```

```
#called by massConvert2pt0.py

#Spectrum and Converter are two classes that
#were written by a previous student in C++
#and can be found on the data drive. I translated these
#files into Python for use in this code.
#As I only translated the files, I will
#not include them in this paper but they can
#be found on the data drive

def __main__(inFile,outFile):

    smoothsize=40
    spec=Spectrum.Spectrum(inFile)
    spec.reverse()

    conv=Converter.Converter()
    spec2=conv.smooth(spec,smoothsize,"BOXCAR")
    spec2=spec


    #get rid of negatives
    dataList=spec2.getDataList()
    for n in range(0,len(dataList)):
        if dataList[n]<0:
```

```
            dataList[n]=0
    spec2.setDataList(dataList)

    #check the initial and final lambda values
    wavelengthList=spec2.getWavelengthList()
    startWavelengthValue=wavelengthList[0]
    endWavelengthValue=wavelengthList[-1]

    #create a new lambda grid
    #if the file does not have enough data
    #pts, skip and print out error
    newBegin=0.94
    newEnd=2.41
    dlambda=0.001

    if newBegin>=startWavelengthValue and newEnd<=endWavelengthValue:
        #perform a cubic spline interpolation
        tck = interpolate.splrep(wavelengthList, dataList)
        newWavelengths = np.arange(newBegin,newEnd+dlambda,dlambda)
        newFluxes = interpolate.splev(newWavelengths, tck, der=0)

        #print to outfile
        with open(outFile, "w") as file:
            for n in range(0,len(newWavelengths)):
                wavelength=newWavelengths[n]
```

```
                flux=newFluxes[n]
                file.write(str(wavelength))
                file.write("␣")
                file.write(str(flux))
                file.write("\n")

    else:
        print("This␣file␣doesn't␣have␣enough␣data␣points.␣SKIPPED.")
        if os.path.exists(outFile):
            os.remove(outFile)
            print(f'{outFile}␣has␣been␣removed')
```

Appendix D

pipeline.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 22 14:56:46 2020

@author: sanan
"""

import subprocess
from regridTheSpectrum import regrid
from compareFits import compareFits
import sys

#This function runs all of the fitting
#processes on a single object.
```

```
#This function is called by the
#main function WTII.py so that
#multiple objects can be fit at once.
#I have provided this code as a stand-alone script
#In case the user wishes to only
#run the code on one object at a time

#The function regridTheSpectrum is a simple
#function that regrids a data file to a
#set wavelength. This can be found on the data
#drive.

#The original script compareFits.py was written
#by John-Michael Eberhard. As I only
#changed it to work as a function
#and edited a few parts, while
#Eberhard created the majority, I will not
#include this script in this document.
#It can be found on the data drive

inputFile=f'{sys.argv[1]}ErrFinal.txt'
regriddedSpectrumFile=f'{sys.argv[1]}ReadyToRockAndRoll.txt'
paramsFile=f'{sys.argv[1]}Params.txt'
modelsFolder="Models"
modelTextFile=f'{modelsFolder}/Models.txt'
```

```
def run_the_make_file():
        subprocess.check_call(["make"])

def run_binary_fitter_program():
        stdout = open(f'{sys.argv[1]}BF.txt',"wb")
        subprocess.call(['./BinaryFitter',regriddedSpectrumFile,
        paramsFile],stdout=stdout)

def run_single_fitter_program():

    modelFileNames=[]
    stdout = open(f'{sys.argv[1]}SF.txt',"a")
    with open(modelTextFile,"r") as file:
        for name in file:
            modelFileNames.append(str.rstrip(name))

    for i in range(0,len(modelFileNames)):
        subprocess.call(["./fit",regriddedSpectrumFile,
        modelFileNames[i],paramsFile],stdout=stdout)



print('Checking␣that␣the␣c++␣files␣are␣present␣and␣have␣been␣compiled.\n')
run_the_make_file()
```

```
print('\nSuccess. Feel free to comment out the compiling
command after running this code once\n')

print(f'Your input spectrum {sys.argv[1]} has been received.\n')
print(f'Regridding input spectrum to fit the raw solar models. Placing
new spectrum in a file called {sys.argv[1]}ReadyToRockAndRoll.txt\n')
regrid(inputFile,regriddedSpectrumFile)
print(f'Success\n')

print(f'Now running Binary Fitter and placing output
in a new file called {sys.argv[1]}BF.txt\n')
run_binary_fitter_program()
print(f'Success\n')

print(f'Now running Single Fitter and placing output
in a new file called {sys.argv[1]}SF.txt\n')
run_single_fitter_program()
print(f'Success\n')

print(f'Now comparing binary and single fits
and calculating statistics\n')
compareFits(sys.argv[1])
```

Appendix E

WTII.py

```
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 27 14:44:22 2021

@author: sanan
"""

import sys
import subprocess
import shutil
import os
from regridTheSpectrum import regrid
from compareFits import compareFits
myList=f'{sys.argv[1]}.txt'
```

```
#This is the only code a user should have to
#call when running the updated fitting process
#once the models are ready to go.
#This code takes a list of objects
#in a .txt file as input and
#calls the pipeline.py function on each of them.

#Define sub functions used in the pipeline function
def run_the_make_file():
    subprocess.check_call(["make"])

def run_binary_fitter_program(code):
    stdout = open(f'{code}BF.txt',"wb")
    regriddedSpectrumFile=f'{code}ReadyToRockAndRoll.txt'
    paramsFile=f'{code}Params.txt'
    subprocess.call(['./BinaryFitter',regriddedSpectrumFile,paramsFile]
    ,stdout=stdout)
    stdout.close()

def run_single_fitter_program(code):
    modelFileNames=[]
    regriddedSpectrumFile=f'{code}ReadyToRockAndRoll.txt'
    paramsFile=f'{code}Params.txt'
    modelsFolder="Models"
```

```
    modelTextFile=f'{modelsFolder}/Models.txt'
    stdout = open(f'{code}SF.txt',"a")
    with open(modelTextFile,"r") as file:
        for name in file:
            modelFileNames.append(str.rstrip(name))
    for i in range(0,len(modelFileNames)):
        subprocess.call(["./fit",regriddedSpectrumFile,modelFileNames[i]
        ,paramsFile],stdout=stdout)
    stdout.close()

#Define the pipeline function
def pipeline(code):
    inputFile=f'{code}ErrFinal.txt'
    regriddedSpectrumFile=f'{code}ReadyToRockAndRoll.txt'

    print(f'Your input spectrum {code} has been received.\n')
    print(f'Regridding input spectrum to fit the raw solar models.
    Placing new spectrum in a file called {regriddedSpectrumFile}\n')
    regrid(inputFile,regriddedSpectrumFile)
    print(f'Success\n')

    print(f'Now making a new folder called {code}Results to hold all
    files generated here.\n')
    move=shutil.move
    currentDir=os.getcwd()
```

```
    newFolder = f'{currentDir}/{code}Results'
    if not os.path.exists(newFolder):
        os.makedirs(newFolder)
    print(f'Success.\n')

    print(f'Now␣running␣Binary␣Fitter␣and␣placing␣output
␣␣␣␣in␣a␣new␣file␣called␣{code}BF.txt\n')
    run_binary_fitter_program(code)
    move(f'{currentDir}/{code}BF.txt',newFolder)
    print(f'Success\n')

    print(f'Now␣running␣Single␣Fitter␣and␣placing␣output
␣␣␣␣in␣a␣new␣file␣called␣{code}SF.txt\n')
    run_single_fitter_program(code)
    move(f'{currentDir}/{code}SF.txt',newFolder)
    print(f'Success\n')

    print(f'Now␣comparing␣binary␣and␣single␣fits␣and
␣␣␣␣calculating␣statistics\n')
    move(f'{currentDir}/{inputFile}',newFolder)
    move(f'{currentDir}/{regriddedSpectrumFile}',newFolder)
    os.chdir(newFolder)
    compareFits(code)
    move(f'{newFolder}/{inputFile}',currentDir)
    os.chdir(currentDir)
```

```
    print(f'Success\n')



#run the make file
print('Checking␣that␣the␣c++␣files␣are␣present␣and
have␣been␣compiled.\n')
run_the_make_file()
print('\nSuccess.␣Feel␣free␣to␣comment␣out␣the
compiling␣command␣after␣running␣this␣code␣once\n')

#Read in all of the object names
objectNames=[]
with open(myList,"r") as file:
    for name in file:
        objectNames.append(str.rstrip(name))

#Run the fitting code on each of the objects
for i in range(0,len(objectNames)):
    currentObject=objectNames[i]
    print(f'\n\nNow␣Fitting␣Object␣{currentObject}\n')
    pipeline(currentObject)
```

Bibliography

Beus, L. M., Miles, M., & Stephens, D. 2020, presented at the 235th American Astronomical Society Meeting,

Burgasser, A. J. 2008, Proceedings of the International Astronomical Union, 4(S258), 317

Burgasser, A. J., Cruz, K. L., Cushing, M., Gelino, C. R., Looper, D. L., Faherty, J. K., Kirkpatrick, J. D., & Reid, I. N. 2010, Astrophysical Journal, 710, 1142

Burgasser, A. J., Geballe, T. R., Leggett, S. K., Kirkpatrick, J. D., & Golimowski, D. A. 2006, The Astrophysical Journal, 637, 1067

Burgasser, A. J., et al. 2000, From Giant Planets to Cool Stars ASP Conference Series, 212, 65

Chiu, K., Fan, X., Leggett, S. K., Golimowski, D. A., Zheng, W., Geballe, T. R., Schneider, D. P., & Brinkmann, J. 2006, The Astronomical Journal, 131, 2722

Cushing, M. C., et al. 2011, The Astrophysical Journal, 743(1), 50

Eberhard, J. 2020, (Unpublished Senior Thesis) Brigham Young University

Esplin, T. 2010, (Unpublished Senior Thesis) Brigham Young University

Farnbach, L. 2017, (Unpublished Senior Thesis) Brigham Young University

Gagliuffi, D. C. B., et al. 2014, The Astrophysical Journal, 794, 143

Gardner, D. 2015, (Unpublished Senior Thesis) Brigham Young University

Golimowski, D. A., Burrows, C. J., Kulkarni, S. R., Oppenheimer, B. R., & Brukardt, R. A. 1998, The Astronomical Journal, 115, 2579

Kirkpatrick, J. D., et al. 1999, The Astrophysical Journal, 519(2), 802

Kumar, S. 1963, The Astrophysical Journal, 137, 1121

Marley, & Leggett. 2009, Procedures of Astrophysics in the Next Decade, 101

Matt, K. 2017, (Unpublished senior thesis) Brigham Young University

Morley, C. V., Fortney, J. J., Marley, M. S., Visscher, C., Saumon, D., & Leggett, S. K. 2012, The Astrophysical Journal, 756(2), 172

NASA, ESA, & Stumpf, M. 2009, https://www.spacetelescope.org/images/opo0901a/

Padoan, P., & Norlund, A. 2002, The Astrophysical Journal, 576(2), 870

Rebolo, R., Osario, M., & Martin, E. 1995, Nature, 377, 129

Reipurth, B., & Clarke, C. 2001, The Astronomical Journal, 122(1), 432

Salway, E. 2015, (Unpublished senior thesis) Brigham Young University

Saumon, D., & Marley, M. S. 2008, The Astrophysical Journal, 689(2), 1327

Stephens, D. C., et al. 2009, The Astrophysical Journal, 702(1), 154

Thies, I., Pflamm-Altenburg, J., Kroupa, P., & Marks, M. 2015, The Astrophysical Journal, 800(1), 72

Whitworth, A. P., & Zinnecker, H. 2004, Astronomy Astrophysics, 427(1), 299

Wright, K. 2015, (Unpublished senior thesis) Brigham Young University

Index

Brown Dwarf
    2MASS 0559, 17
    binary fraction, 9
    characteristics, 4
    discovery, 1, 4
    formation, 9
    L Type, 5
    overluminous, 10, 17
    spectra, 2
    Spectral Type, 4
    T Type, 5
    Y Type, 7

Electron Transitions, 3

Fragmentation, 8

Initial Mass Function, 9

Jean's Mass, 7
Jupiter, 2

Planck-Einstein Relation, 4
Point Spread Function, 13

Rayleigh's Criterion, 10

Spectral Fitting, 13
    chi-squared analysis, 15
    confidence level, 15
Spectrum
    absorption and emission lines, 2, 3
    Spectral Type, 4
Stellar Formation, 7

Wein's Law, 3