Generation of Synthetic Color-Magnitude Diagrams for Protoclusters

Winston Zhang

Instructor: Professor Dr. Jonathan C. Tan,
with help from Aayush Gautam and Dr. Juan P. Farias

University of Virginia
Department of Astronomy

This thesis is submitted in partial completion of the requirements of the BA Astronomy Major
May 10, 2024

ABSTRACT

The color-magnitude diagram (CMD) is a type of Hertzsprung-Russell (H-R) diagram, which charts stellar populations by plotting stars' brightness in magnitudes versus their color index, from which we can extract information about the formation and evolution of stars and stellar populations. In this paper, I present Python code that I have implemented which reads data from Nbody6 protocluster simulations and interfaces with a software package called Stellar Population Interface for Stellar Evolution and Atmospheres (SPISEA), to calculate photometric values and generate a synthetic CMD. Using the generated CMDs, I also present a study of the effects of age spread within a simulated star cluster on the spread of magnitude brightness of the stellar population, and how these effects are seen in the CMD.

1. INTRODUCTION

1.1. *Hertzsprung-Russell and Color-Magnitude Diagrams*

Independently invented by Ejnar Hertzsprung in 1911 and Henry Norris Russell in 1914, the *Hertzsprung-Russell (H-R) diagram* is a scatterplot of stars establishing a relation between their luminosity (energy output in watts) and their effective temperature (Hertzsprung 1908, 1911; Russell 1914). The *Color-Magnitude diagram (CMD)* is equivalent to the H-R diagram, with the difference that it replaces luminosity with apparent magnitude and effective temperature with color index.

CMDs prove to be advantageous for observational contexts, as values for color index and apparent magnitude are easier to observe and measure since they are dependent purely on a star's appearance. Luminosity and effective temperature, on the other hand, require additional computation, as luminosity is derived from apparent brightness & distance via the inverse-square law, and temperature is derived from peak wavelength via Wien's law. These computations often entail using other observational techniques, including determining distance from parallax angle or standard candles, as well as correcting for effects such as reddening, extinction, and Doppler shift.

The features in H-R diagrams and CMDs reveal vital information regarding stellar evolution, such as the main sequence, when stars fuse hydrogen in their cores, or the red giant branch, when stars toward the end of their life begin hydrogen shell fusion and helium core fusion (The Editors of Encyclopedia Britannica 2021). These features are seen in the H-R diagram shown in Figure 1. Plotting any given stellar population on an H-R diagram or CMD discloses a wide variety of information, such as its overall age, mass distribution, metallicity, and the rate at which its constituent stars formed.

My project aims to introduce an agile and modular software solution that can interface with simulation datasets to create color-magnitude (as well as Hertzsprung-Russell) diagrams. This would be especially useful in observational applications, as a user could then model the appearance of a stellar population of interest, as well as how its appearance is affected by certain properties regarding its formation.

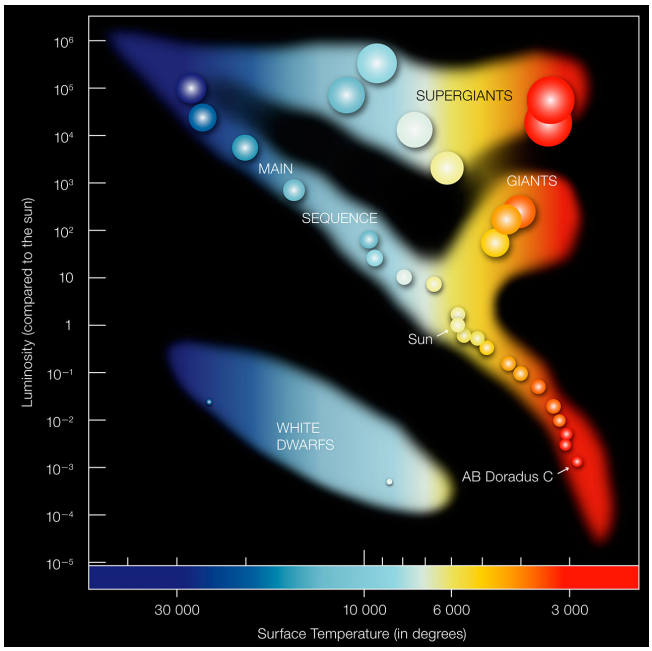

Figure 1. Example H-R diagram plotting stellar luminosity against effective temperature (European Space Agency 2007).

1.2. *Measuring a Star's Brightness via the Magnitude System*

The brightness of a star is directly tied to its energy output, commonly referred to in astronomy as its *luminosity*. A star's luminosity is one of its most important characteristics, as it is closely related to its other properties, such as its size, mass, and effective temperature, all of which dictate its rate of nuclear fusion and chemical composition. While luminosity is a vital quantity in

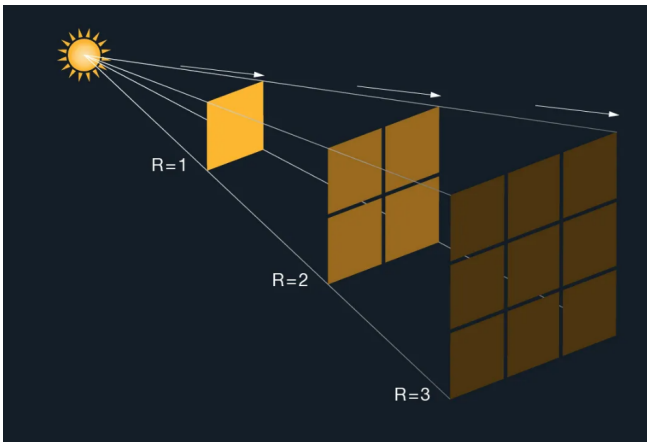

Figure 2. Illustration demonstrating the inverse square law (National Aeronautics and Space Administration n.d.).

astronomy, a star's brightness as it appears in the night sky relative to other stars is not always consistent with their relative luminosities. In the same way a candle appears increasingly dimmer when viewed from greater distances, a star may appear dimmer than a star of lesser luminosity if it is sufficiently far away. Apparent brightness decreases exponentially with an observer's distance from a source due to light dispersing spherically, and spherical surface area scaling exponentially with radial distance. This is known as the *inverse-square law*, described in Equation 1 and visualized in Figure 2.

$$Apparent\ brightness = \frac{Luminosity}{4\pi * distance^2} \tag{1}$$

Therefore, distinguishing between the intrinsic luminosity of a star and its apparent brightness becomes paramount. The magnitude system is a logarithmic scale used to describe the brightness of objects in astronomy, a measure central to this project. Magnitude brightness can be described as either an apparent magnitude or an absolute magnitude.

1.2.1. *Apparent Magnitude*

Originating in Hellenistic Greece, the magnitude system was created based on stars' apparent brightness. Stars were organized into six rankings ranging 1–6, with brighter stars having a lesser magnitude and dimmer stars having a greater magnitude (Miles 2007).

In 1856, Norman Robert Pogson formalized the magnitude system into a logarithmic scale by establishing that a first-magnitude star is 100 times brighter than a sixth-magnitude star. Based on this stipulation, a difference of one magnitude corresponds to a brightness ratio of 2.512. This brightness is measured as *flux*, the energy per unit area of a star's light as it reaches the observer's location. The relation that dictates the Pogson magnitude scale in terms of flux is described in Equation 2:

$$m_1 - m_2 = -2.5 \log_{10}(f_1/f_2) \tag{2}$$

This relation yields the magnitude difference between stars 1 and 2, where star 1 is the star of interest, and star 2 is the reference star to which the flux ratio is relative. This relation can be applied to any two stars in the night sky. Since any star can be chosen as a reference star, defining magnitudes in this manner is referred to as *relative photometry*.

To standardize the magnitude system, astronomers agreed upon the star Vega to define the zero-point flux by convention, such that the magnitude brightness for any given star in reference to Vega is on an "absolute" scale. This is known as both the *Vega system* and *absolute photometry*. The Vega system follows the same mathematical definition as relative photometry, except the flux of reference star 2 is replaced with the flux of Vega, and the corresponding magnitude is set to zero, shown in Equation 3.

$$m = -2.5 \log_{10}(f/f_{Vega}) \tag{3}$$

The Vega-calibrated Pogson magnitude system is also visualized in Figure 3.

1.2.2. *Absolute Magnitude*

A star's absolute magnitude (not to be confused with absolute photometry) is a measure of its brightness on the same descending logarithmic scale while eliminating the brightness-attenuating effects of distance, extinction, and reddening (Hughes 2006). This is done by taking the apparent magnitude of the object as if it were viewed from a distance of 10 parsecs, assuming no intermediate gas or dust. By holding distance constant, the absolute magnitude of stars can be used as a surrogate measure for luminosity. As such, absolute magnitude is sometimes seen as a replacement for luminosity on the vertical axis of H-R diagrams, although the use of absolute magnitude is outside the focus of this paper.

1.3. *Measuring a Star's Color via Color Index*

While color is often treated as a categorical variable in statistical contexts (especially econometric and behavioral), it is a quantitative variable in astrophysics, as an object's color is directly related to the wavelength and intensity of the light that it emits. The color-determining wavelength of peak intensity for the light emitted by an object is dictated by *Wien's law*, shown in Equation 4.

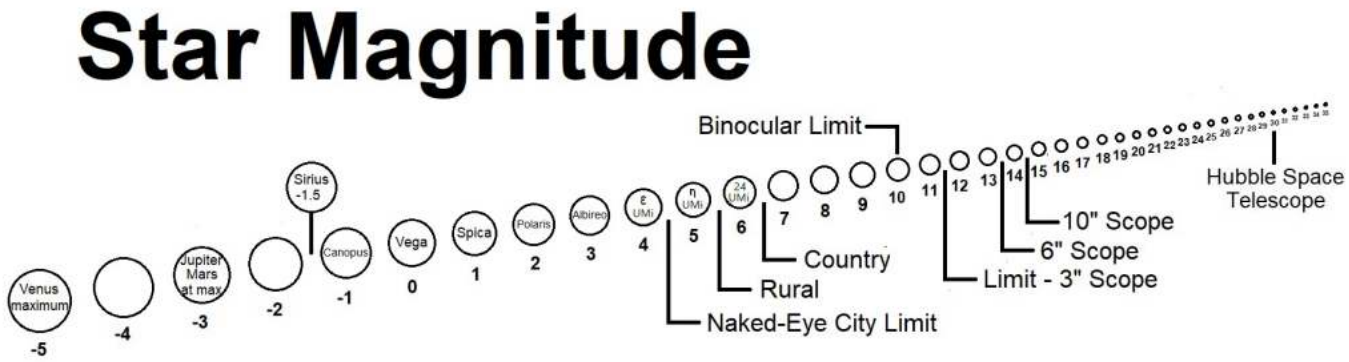

Figure 3. Visualization of Vega-calibrated Pogson Magnitude scale, with real-world examples (Milwaukee Astronomical Society 2024).

$$\lambda_{peak}\ (\mu m) = \frac{2900 \mu m * K}{T\ (K)} \tag{4}$$

In addition to an object's peak wavelength, the intensity of its emitted light is also dependent on its temperature and is dictated by the *Stefan-Boltzmann law*, described in Equation 5.

$$Intensity = \sigma T^4 \tag{5}$$

The inverse relationship that temperature has with peak wavelength and the quadratic relationship between temperature and radiative intensity are together observed in *Planck's law*, which describes the distribution of light intensity as a function of wavelength. Thus, hotter objects are observed to have Planck distributions which are "taller" in shape and their centers closer to the end of the wavelength axis corresponding to shorter wavelengths. Planck's law is the underlying cause of why massive stars, which have higher effective temperatures, appear brighter and bluer: temperature is directly proportional to core gravitational pressure, which increases with mass. The appearance of the Planck distribution as dictated by temperature is shown in Figure 4.

Astronomers take advantage of Planck's law to define a quantitative measure for color, commonly referred to as a *color index*. Color index is derived by measuring the brightness of an object in magnitudes for two separate wavelength bands using photometric filters, and subtracting the two quantities. By convention, the magnitude brightness in the longer wavelength band is subtracted from that of the shorter wavelength band. This metric yields a smaller (or even negative) number for bluer objects, as blue objects have lesser magnitudes in the blue and greater magnitudes in the red. A demonstration of how color index is measured is shown in Figure 5.

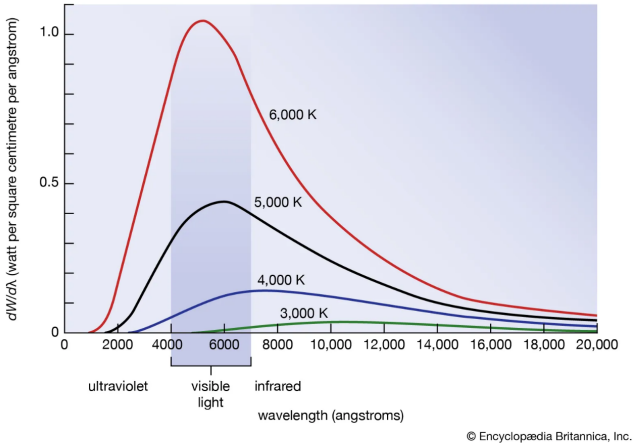

Figure 4. Diagram of Planck's law as a function of temperature (The Editors of Encyclopedia Britannica 2024).

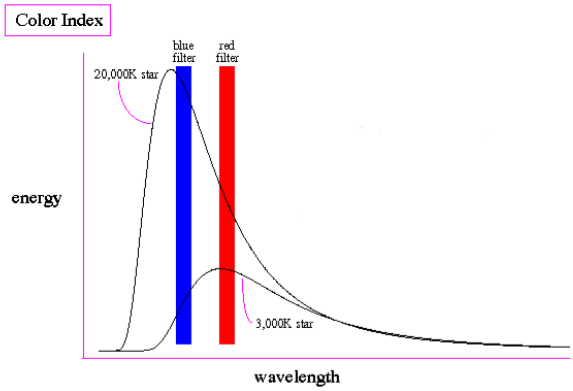

Figure 5. Demonstration of how color index is computed by selectively measuring radiative intensity in two different wavelength bands. Adapted from Schombert (2015).

1.4. *Star Formation*

A star is born as the result of the gravitational collapse of a cloud of gas and dust in interstellar space that achieves a critical density dictated by the Jeans mass and Jeans length (McKee & Ostriker 2007). These clouds are commonly known as molecular clouds because they contain molecular species, the most common of which is molecular hydrogen. Molecular clouds are not uniformly distributed in mass and have internal turbulence, which introduces variations in density throughout. As a molecular cloud collapses under its gravity, variations in density become more pronounced as dense regions (or clumps) have a greater mass concentration and thus collapse faster than lower-density regions. Stars are known to form in clusters within these embedded clumps (Lada & Lada 2003). This process is illustrated in Figure 6. The formation of stars via the collapse of molecular clouds is a part of the stellar life cycle, as dying stars return gas and heavy elements (known in astronomy as metals) to the interstellar medium, from which gas and dust can clump up to form stars anew.

Star formation is a field that is actively studied within astronomy, and is an intricate process that involves many interacting astrophysical processes. It is the hope that modeling star formation will aid in understanding its parameter space and how the process responds to changes in said parameters. Such advancements in understanding star formation will bring astrophysicists closer to answering a wide range of questions, such as how planets form, how the universe looked in its infancy, and how stellar populations change as galaxies evolve.

2. RELATED WORKS

A similar project was undertaken by an undergraduate student in 2021 for the Chalmers Astrophysics & Space Science Summer (CASSUM) Research Fellowship Program, which sought to generate synthetic H-R diagrams using isochrone models and gravitational simulations (Masegian 2021). This project takes a similar approach to generating a synthetic diagram, namely using mass and age data to interpolate between isochrone points, although this project was limited in scope to only generating H-R diagrams that plot luminosity against effective temperature.

My project expands upon this by using the software package SPISEA (described in Section 3.2.1) to generate isochrone curves with which interpolation is executed in order to determine stellar properties and photometric values used in plotting stars on the CMD. SPISEA is instrumental in my project, as I focus on generating CMDs as opposed to H-R diagrams, and SPISEA features synthetic photometry software routines to calculate a star's apparent brightness in magnitudes. SPISEA especially shines in this application because it exhibits a high level of modularity, accounting for myriad variables in the synthetic photometric process (Hosek et al. 2020).

Both my project and Masegian (2021) draw upon Farias et al. (2019), the second paper to a three-paper series that explores and models the formation of star clusters. In it, they present Nbody6 simulation code for the formation of hypothetical protoclusters embedded within in a molecular cloud.

3. METHOD

3.1. *Software Environment Information*

My code was written in Python 3.9.12, primarily run in Jupyter Notebook 6.4.8. Astropy 5.0.4 was used for opening a variety of *.fits* data files, as well as storing data in the form of tables. Numpy 1.21.5 was used for performing mathematical operations on data in the form of arrays. Matplotlib 3.5.1 was used for generating plots. Pdb was used for debugging code.

3.2. *Model Preparation*

Generation of a synthetic CMD is dependent on two software packages: Stellar Population Interface for Stellar Evolution and Atmospheres (SPIAEA), and Nbody6. SPISEA provides the isochrones upon which photometric values are evaluated via interpolation, and gravitational simulations run in Nbody6 supply the values for mass and age for which photometric values are interpolated.

3.2.1. *SPISEA*

SPISEA (pronounced "spicy", as per the creators) is a software package for modeling stellar populations written in Python (Hosek et al. 2020). It has a robust input space, accepting up to thirteen input parameters, including age, distance, stellar evolution model, stellar atmosphere model, extinction, photometric filters, and more.

SPISEA offers support for several commonly used stellar evolution models: MESA Isochrones & Stellar Tracks (MIST) (Choi et al. 2016), Geneva (Ekström et al. 2012), PAdova & TRieste Stellar Evolution Code (PARSEC) (Bressan et al. 2012), Baraffe (Baraffe et al. 2015), and Pisa (Tognelli et al. 2011). Additionally, SPISEA offers support for a wide suite of photometric filters, including those used by Hubble, 2MASS, Gaia, JWST, and more. SPISEA provides users with the ability to define their own evolution/atmosphere models and photometric filters.

Upon specifying the desired input values, SPISEA undergoes a process of calculating synthetic photometry

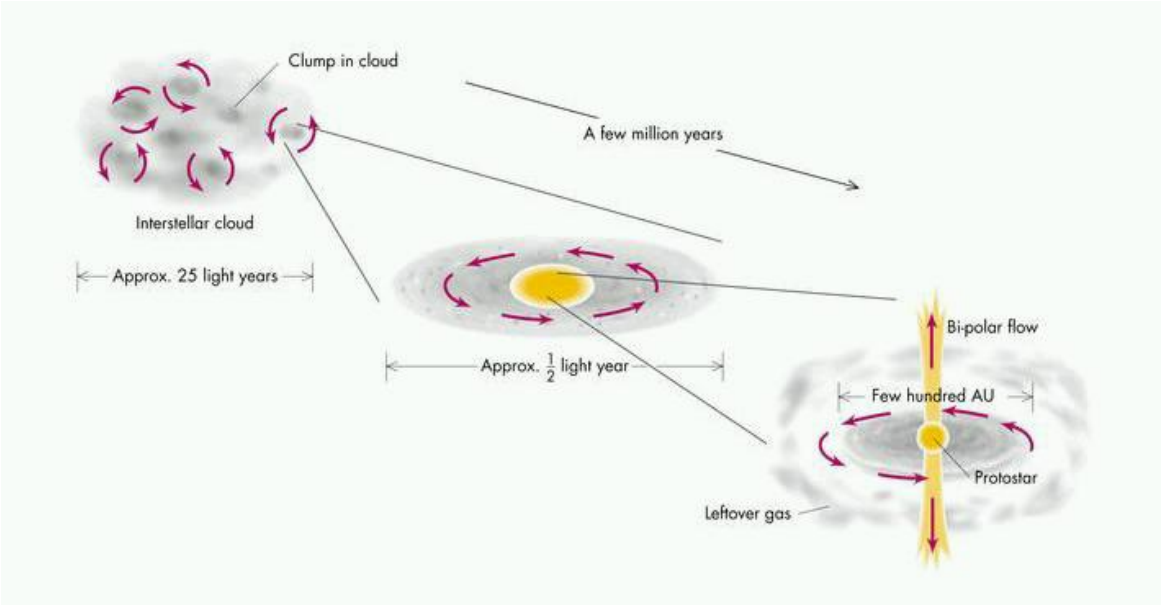

Figure 6. Visualization of star birth via gravitational collapse of a molecular cloud (Min 2008).

for a mass range of stars for a specified age, generating a data table that stores values such as luminosity, effective temperature, and brightness in Vega system magnitudes for the specified filters. These data points can then be used to plot an isochrone curve, which represents how a star's magnitude and color index (or in the case of an H-R diagram, luminosity and temperature) vary with mass, holding age constant. Example isochrone curves generated by SPISEA are shown in Figure 7.

SPISEA can generate multiple isochrones in one command, if an input parameter is supplied as an array of values, as opposed to a single value. This is particularly useful in generating an isochrone that contains photometric data for more than one filter, or for generating a set of isochrones for multiple ages. A combination of input parameters can be specified as arrays, so one can supply both an array of ages and an array of filters to generate a set of isochrones that store photometric values for multiple filters, which also span a range of ages. This feature is used quite extensively in my project, as the generation of a synthetic CMD, especially one with age spread, requires an isochrone grid to be generated for multiple ages to model stellar evolution, and also requires more than one filter to be used in the synthetic photometry calculation such that a color index can be computed for each star and plotted on the horizontal axis.

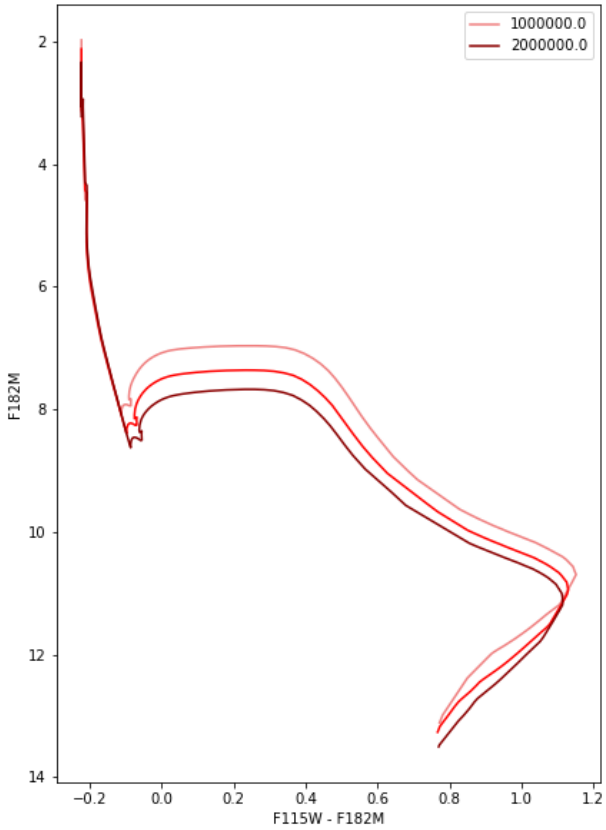

Figure 7. 1, 1.5, and 2 Myr MIST isochrones generated by SPISEA in JWST/NIRCAM filters F115W and F182M, at distance $d = 410pc$, with no extinction, and solar metallicity.

Parameter	Cloud Mass Density Σ_{cloud}, (g/cm^3)	Freefall Timescale ϵ_{ff} (scalar)
Possible values	0.1	0.01
	1.0	0.03
		0.10
		0.30
		1.00

Table 1. Values for Nbody6 protocluster formation simulation parameters.

3.2.2. *Nbody6*

Although SPISEA includes functionality for simulating a cluster and generating a CMD for it, my project seeks to create CMDs using cluster simulations separate from SPISEA. While SPISEA has a wide-spanning parameter space, the synthetic CMD generator lacks two keystone parameters critical to creating a CMD, which are the datasets containing the values for stellar age and mass. Without values for age, SPISEA cannot generate isochrone curves, which are age-dependent, and without values for mass, the CMD generator does not know where along the isochrone curves each star lies.

The Nbody6 simulations from Farias et al. (2019) emulate the gradual formation of a protocluster inside a clump of 3000 solar masses embedded within a molecular cloud. These simulations follow an initial mass function described in Kroupa (2001) and feature primordial binary systems, pairs of gravitationally-bound stars that form together. The simulations are defined with a wide range of parameters describing the physical properties of the clump and parent cloud. The parameters most relevant to this project are the density of the cloud Σ_{cloud} in grams per cubic centimeter and the freefall timescale parameter ϵ_{ff}, a scalar value for which higher values represent a faster freefall. The possible values for these parameters are summarized in Table 1. The interplay of these parameters affects gravitational interactions, and thus the rate of star formation. The mass and age for each star for any given elapsed cluster time t_{cl} can be extracted from the simulations as an Astropy table.

The aim of simulating and analyzing these gradually forming cluster models is to attribute observable effects that accompany the spread in the age of constituent stars to the properties of the cloud from which they form. Understanding the effect of age spread on the appearance of star clusters would then allow astrophysicists to observe a real-life star cluster and estimate in the reverse direction the physical properties of the molecular cloud that formed it.

As noted in Masegian (2021), the clusters simulated in Nbody6 are observed to have greater age spread with longer freefall timescales. The same can be said for clusters with lower mass density, as more time is required for mass to gravitationally collapse to form stars. These effects are expected to manifest themselves in the synthetic CMD in the form of magnitude spread along the vertical axis, as isochrones vary along the vertical axis with age. This is also supported by the effects of age spread on the spread in luminosity seen in Masegian (2021).

3.3. *Diagram Generation*

The process of diagram generation begins with the preparation of the input parameters for generating the SPISEA isochrone grid. First, parameters that can be set in a single assignment statement are specified: distance, extinction, evolution model, atmosphere model, reddening, metallicity, and photometric filters.

A snapshot of an Nbody6 simulation for a particular time t_{cl} since formation is loaded into the program as an Astropy table, from which values for the mass and age of each simulated star are extracted. The values for mass and age are stored in separate Python arrays. The ages for the youngest and oldest stars are then identified, and along with an increment value, are used to define an array of evenly-spaced ages that capture the full age range of the stellar population. If either the youngest or oldest star is out of the age range of the specified evolution model, then the age array is redefined in terms of the bounds of the evolution model. This array of ages is then fed to the SPISEA constructor statement for generating the isochrone grid. This constructor is run, and the resultant generated isochrones are stored in a Python object array. The diagram generator creates grid of isochrone curves for interpolation as opposed to a continuum of curves, as the synthetic photometric calculations done by SPISEA are computationally expensive and thus infeasible to perform for every possible age of the simulated stars. An example isochrone grid is shown in Figure 8.

After the isochrone grid is generated, the diagram generator proceeds to the interpolation process, in which photometric values are calculated for each simulated star. This is done by an iterative loop that reads the mass-age value pair (m, a) for each star, identifies the pair of isochrones whose ages bracket a, performs a linear interpolation along each isochrone curve to locate where a star of mass m appears on the respective curve, then conducts a second linear interpolation between the two points to arrive at the photometric values for a star of mass m and age a. A visual explanation of the interpolation process is included in Figure 9. In addition to photometric values, the interpolation process also com-

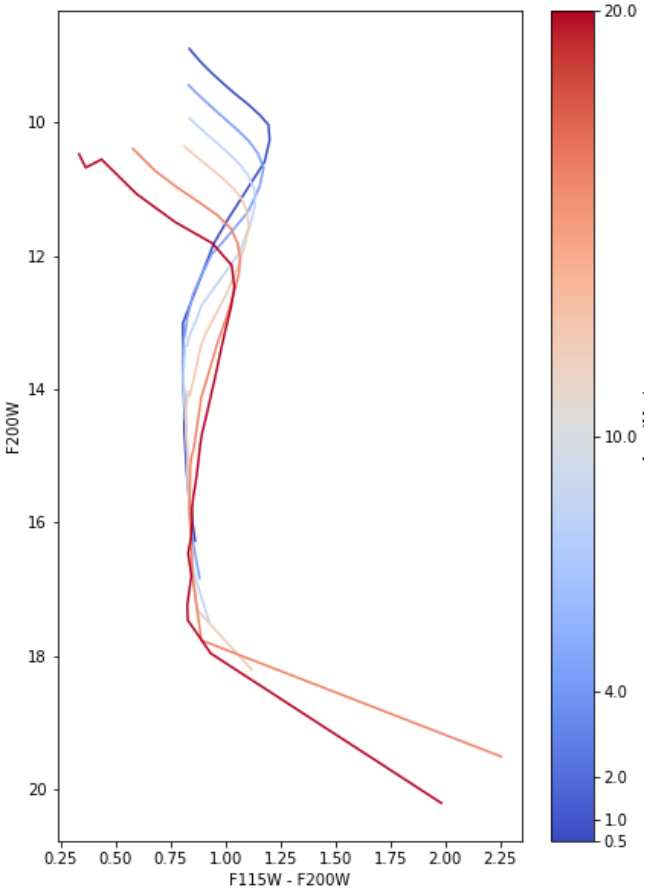

Figure 8. Example grid of isochrones at the ages 500 Kyr, 1 Myr, 2 Myr, 4 Myr, 10 Myr, and 20 Myr. In reality, an isochrone grid generated for the synthetic CMD generator would be regularly spaced; a small selection of ages was plotted to demonstrate a simple example of how isochrone curves vary with age.

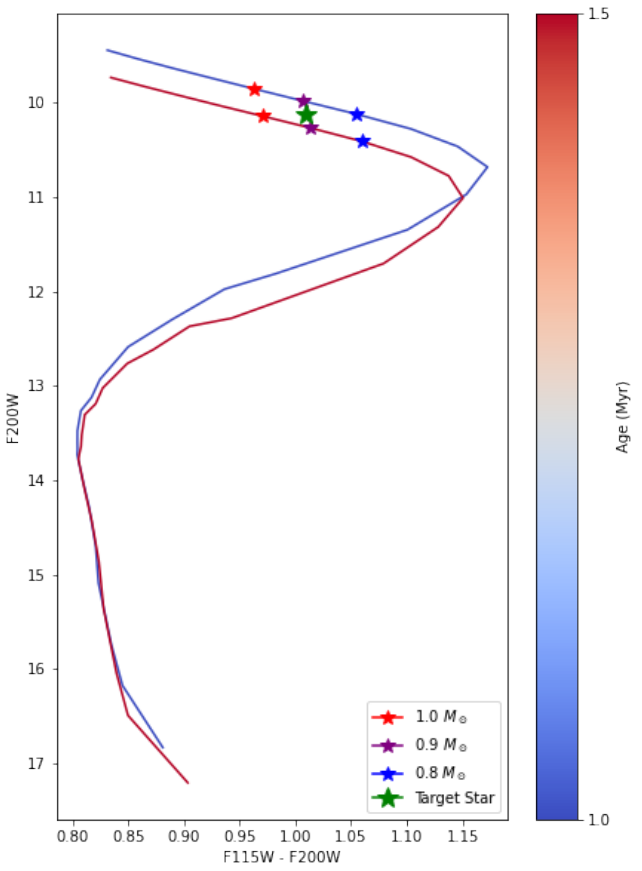

Figure 9. Example interpolation of a 0.9 M_{sun} star with age 1.25 Myr. Example points are more spread out than reality for the sake of the reader's understanding. In reality, photometric interpolation is done using mass values that are much closer together.

putes interpolated values for luminosity, effective temperature, and surface gravity. All of these values are stored in an array that is used for the subsequent plotting of stars as points on the CMD.

Once all simulated stars have been run through the interpolation process, the diagram generator is ready to create the diagram. It begins by plotting the isochrone curves generated by SPISEA, followed by the plotting of stars' output values as computed by the interpolator. The user can redefine the code to dictate whether the program plots apparent magnitude versus color index for generating a CMD, or luminosity versus effective temperature for generating an H-R diagram.

3.4. *Accounting for Binary Systems*

As previously mentioned, the Nbody6 simulations in Farias et al. (2019) feature binary systems. Accounting for binaries in the CMD generation process is quite straightforward and will look different depending on whether the user wishes to treat them as resolved or unresolved. A resolved binary is a system of two gravitationally-bound stars that subtends a large enough angular distance to appear to the observer as two separate stars, and an unresolved binary is a system close enough that it appears as a single "star".

The Nbody data table discerns binary systems from singles stars in its method of data representation. It does so by storing the mass of the primary star in the "mass" column and maintaining separate columns that store the companion's mass and the combined mass of the system, as well as a separate column that stores a boolean flag indicating whether the row represents a binary system. Thus, if a row in the table is a single star, this boolean variable will be false, and the companion mass column will be empty. The Nbody data table is ordered in such a way that the table begins with rows representing single stars at the top, with all binary systems occupying the bottom of the table. A simplified visualization of this data representation is shown in Table 2.

Mass	Is Multiple	Companion Mass	System Mass	[...]	Age
[...]					
1 M_{sun}	False	(Empty)	(Empty)		1.5 Myr
1 M_{sun}	True	0.25 M_{sun}	1.25 M_{sun}		1.4 Myr
[...]					

Table 2. Example visualization of Nbody6 table representation for single stars and binary systems.

3.4.1. *Resolved Binaries*

Modeling resolved binary systems is trivial and follows the same steps for plotting single stars. The masses of the companion stars are extracted from the Nbody table, and using the same age as the primary star, each companion star is run through the same process of interpolation and plotting, being treated the same as a single star. Plotting binary systems in this manner results in the appearance of binary stars as a point representing the primary star and another point representing the companion star in a region of lower mass on the same age track, as Nbody6 models all binary systems as having the same age, and always chooses the higher mass star to be the primary star when pairing stars into binary systems.

3.4.2. *Unresolved Binaries*

For unresolved binary systems, a more involved process is undertaken. The same process of extracting the mass of companion stars and running them through the interpolator is taken. However, after interpolation, unresolved binaries are not plotted as a pair of points on the CMD, as an unresolved binary would appear as a single point source in the night sky to an observer. Instead, a combined brightness for the binary system is computed by converting the primary and companion stars' magnitude brightness into flux as per Equation 3 using the zero-point flux of Vega (which is filter-dependent), summing the flux, and converting the combined flux back into a magnitude brightness:

$$f_{primary} = 10^{-m_{primary}/2.5} * f_{Vega}$$
$$f_{companion} = 10^{-m_{companion}/2.5} * f_{Vega}$$
$$m_{binary} = -2.5 \log_{10}(\frac{f_{primary} + f_{companion}}{f_{Vega}})$$

The binary system is then plotted with this new combined apparent magnitude against the color index of the primary star. Plotting a binary system in this manner causes it to be plotted as a point that sits higher on the vertical axis than if the primary star were to be plotted on its own. This has the net effect of creating a "binary sequence", a band of points on the synthetic CMD that appears to hover above the track of single stars due to their higher combined brightness. For this reason, a CMD that treats binary systems as unresolved is expected to have a greater spread in magnitude as opposed to one that plots the same simulated cluster while treating binaries as resolved.

4. RESULTS & ANALYSIS

For a majority of the synthetic CMDs generated, the isochrone grid used for interpolation was created using a proprietary Baraffe evolution model that extends its lower mass range down to 0.01 M_{sun}. The isochrone parameters were specified as follows: the distance was 410 parsecs, extinction was neglected, and the metallicity was set to that of the Sun. A merged atmosphere model was used that is based on the ATLAS9 (Castelli & Kurucz 2004), PHOENIXv16 (Husser, T.-O. et al. 2013), BTSettl (Baraffe et al. 2015), and Koester10 (Koester 2010) atmosphere models. A more detailed description of this merged atmosphere model is found in Lam et al. (2020). The age range was defined to cover a cluster time duration spanning 500 Kyr to 20 Myr in 500 Kyr increments. This age array was fed to the SPISEA isochrone generator to create the isochrone grid that would be used for interpolation. The isochrone grid was plotted without any stars and is shown in Figure 10.

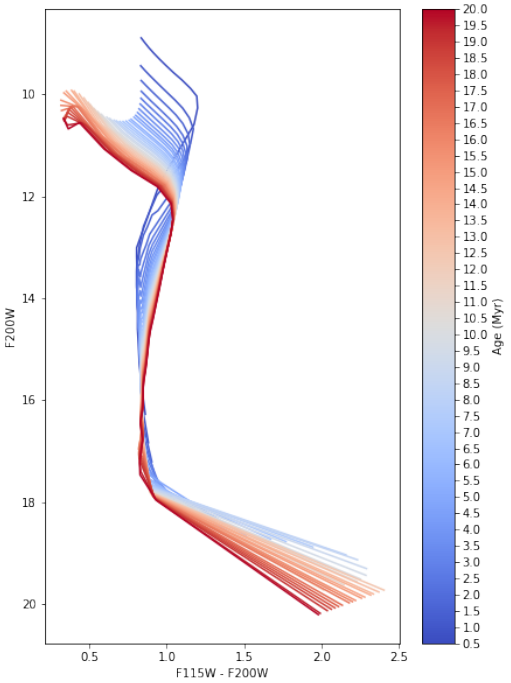

Figure 10. Grid of 0.01–1.4 M_{sun} Baraffe isochrone curves ranging from 500 Kyr to 20 Myr, with a distance of 410 parsecs and no extinction.

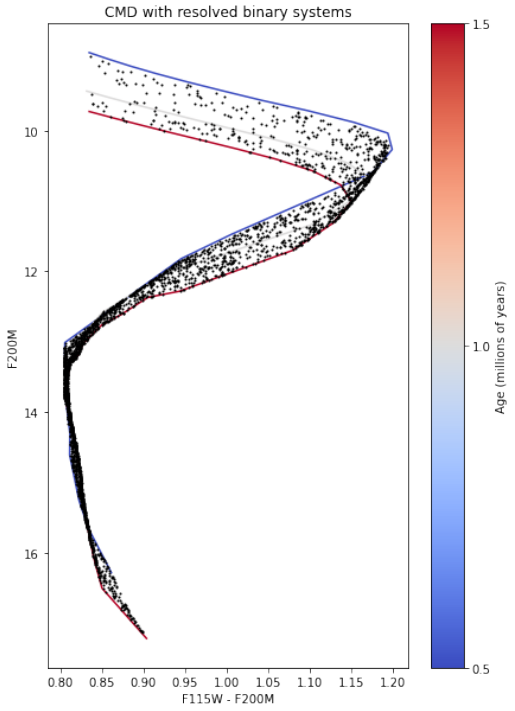

Figure 11. Sample CMD generated using extended-range Baraffe isochrones, for a simulated protocluster with age of 1.5 Myr, cloud density of 1.0 g/cm^3, freefall parameter $\epsilon_{ff} = 0.03$, a distance of 410 parsecs, and no extinction.

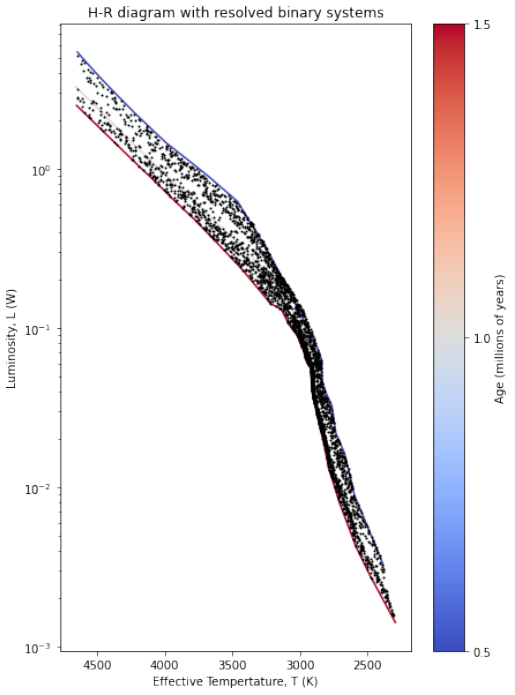

Figure 12. Sample H-R diagram generated using the same parameters as Figure 11.

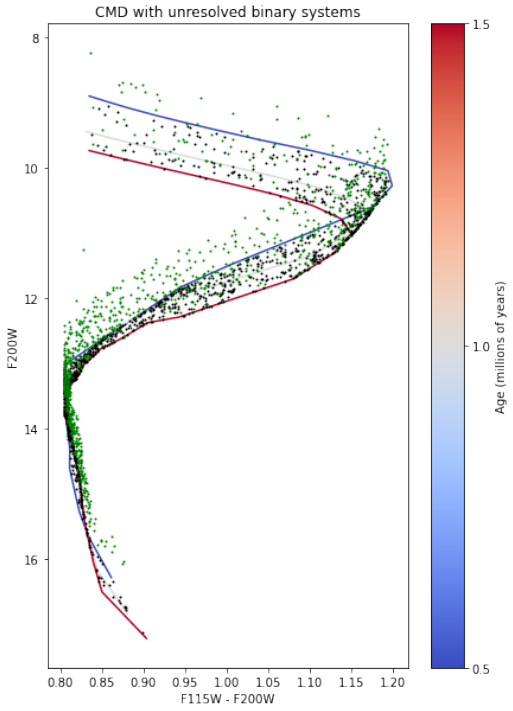

Figure 13. Sample CMD treating binaries as unresolved, generated using the same parameters as Figure 11. Green points represent unresolved binaries and black points represent single stars.

To demonstrate and verify the functionality of the synthetic CMD generator, a CMD and an H-R diagram were generated for a simulated protocluster. The protocluster simulation was loaded with the cluster time set to 1.5 Myr, its cloud density equal to 1.0 g/cm^3, and freefall parameter ϵ_{ff} equal to 0.03. The distance was set to 410 parsecs, and extinction was neglected. The generated color-magnitude diagram is featured in Figure 11 and the H-R diagram generated using the same simulation data and parameters is included in Figure 12. Another CMD was then generated for the same simulated cluster, this time treating binary systems as unresolved, and is shown in Figure 13.

Several CMDs were then generated for a range of ages to show how the appearance of the CMD changes as a cluster ages, shown in Figure 14. An important trend to note is that magnitude brightness on average decreases as the cluster ages. This is supported by the general downward migration of isochrones with age in Figure 10. The same trend is noted with luminosity and age in Masegian (2021).

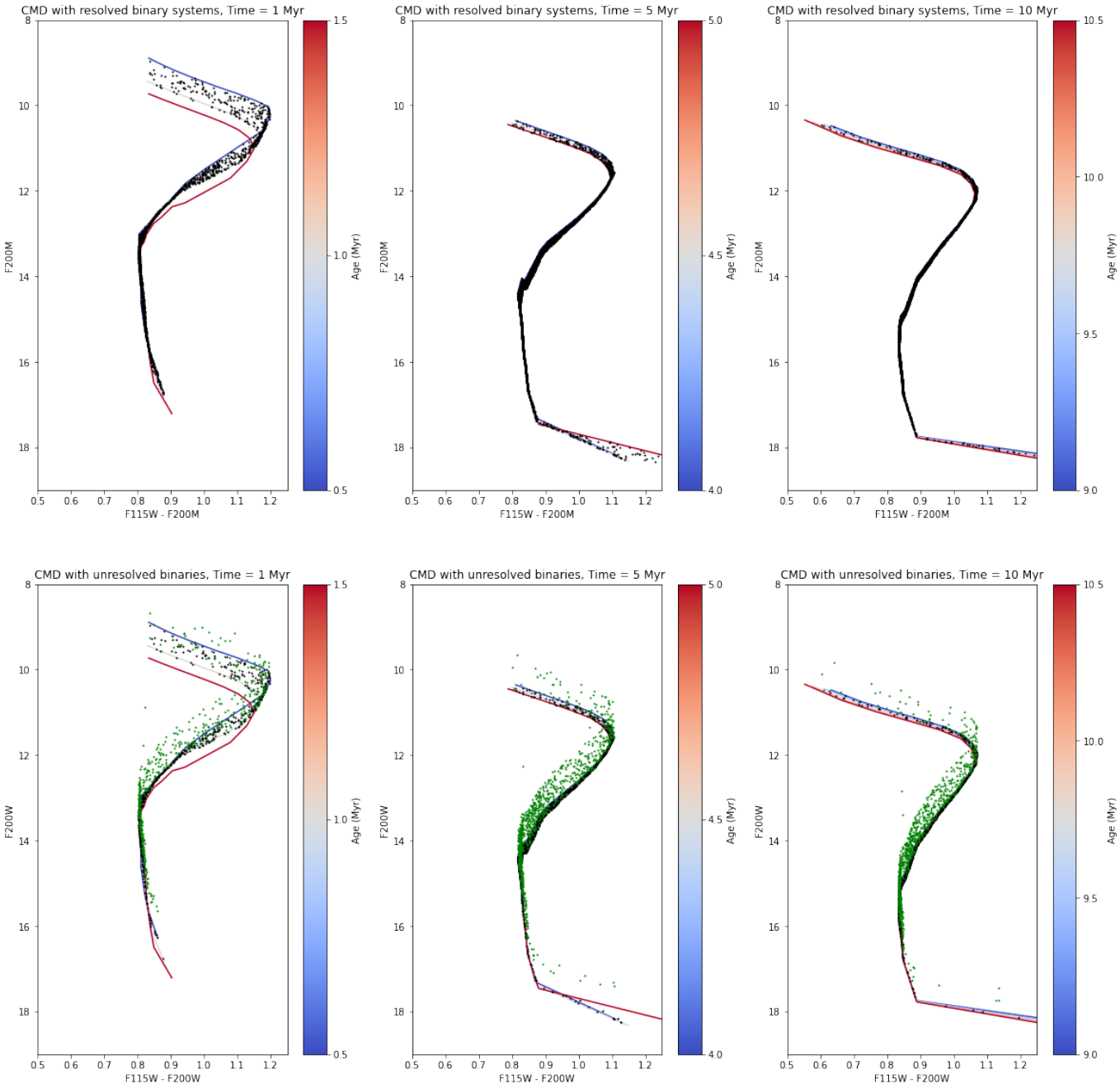

Figure 14. Grid of CMDs generated at various cluster ages for the same simulation parameters as Figure 11. The diagrams show that as a cluster ages, the mean magnitude brightness of stars decreases.

4.1. *Measuring the Effects of Age Spread*

To demonstrate the effects of magnitude spread as a function of freefall timescale, the CMD generator was run on multiple simulation snapshots using the same parameters, with the exception that snapshots were loaded from simulations of varying freefall timescale. Holding age constant, one can see in the generated CMDs that protoclusters exhibit a greater spread in magnitude brightness for longer freefall timescales due to the increased age spread. The CMDs generated that demonstrate the increase in magnitude spread with freefall timescale are shown in Figure 15.

Then, the effects of magnitude spread as a function of freefall timescale were then measured on each CMD by defining a bounding box, splitting it up into color index bins along the horizontal axis, computing the standard deviation for the stars enclosed within each bin, and averaging all of the standard deviations. The bounding box was defined in the intermediate-mass region of the isochrones so that it would measure spread in a region of the diagram where the slope of the curves is approximately constant, avoiding areas where the isochrone curves bend back on themselves. The spread metric was computed as an average standard deviation across several color bins—as opposed to a single standard deviation calculation—to mitigate any contributions that the slope of the isochrone curves may have on the magnitude spread. This metric was measured for all values of freefall timescale ϵ_{ff} as described in Table 1, and plotted against ϵ_{ff} on a logarithmic scale. The magnitude spread as a function of freefall timescale for a cluster time of 2 Myr is shown in Figure 16.

In examining the generated CMDs and measuring the magnitude spread as a function of freefall timescale, holding elapsed cluster time constant, three important trends can be observed:

1. For the same freefall timescale, the magnitude spread is greater for parent clouds of low mass density.

2. CMDs that plot binaries as unresolved have more magnitude spread than those that plot binaries as resolved.

3. For all parent cloud densities, magnitude spread generally increases with a longer freefall timescale (smaller ϵ_{ff}).

These observations are consistent with expectations and agree with the findings in Masegian (2021).

The same metric for magnitude spread was then performed on the same simulations while varying the cluster time. The spread of points along the vertical axis of the CMD is observed to increase with the age of the cluster up to a cluster time of 10 Myr. While this metric is expected to monotonically increase with cluster age, the spread in magnitudes appears to decrease past a cluster age of 10 Myr. This is an artifact and is due to the isochrones shifting in position on the CMD, with regions of higher mass —which have a shallower slope— drifting into the bounding box used to calculate spread. A plot demonstrating the increase in magnitude spread as a function of cluster time is shown in Figure 17.

The increase in magnitude spread over time is determine to be caused by the age spread of the simulated stars, which also increases with time. Age spread increases with cluster time because existing stars age as new stars continue to form, increasing the spread in stellar age until star formation stops. Using the same metric defined in Masegian (2021), I measure age spread for a particular cluster time by computing the 95th percentile of age minus the 5th percentile. This percentile range metric for age spread increases with the cluster time in the same way that magnitude spread does and is shown in Figure 18. Finally, we can see in Figure 19 that magnitude spread is directly correlated to age spread for the age range where the magnitude spread is more reliable. This relationship is also consistent with the positive relationship between luminosity spread and age spread found in Masegian (2021).

4.2. *Time Complexity Analysis of Software Execution*

In theoretical computer science, time complexity is a method of analyzing how efficiently a software program can execute, based on the size of its inputs (Papadimitriou 2014). Time complexity analysis examines how computation time is expected to trend as the input size increases, described as a mathematical relationship. Because time complexity analysis examines the asymptotic behavior of software as a function of its input size, time complexity is interchangeably referred to as asymptotic analysis. Time complexity analysis can model a program's best-case (*"Big-Omega"*), worst-case (*"Big-O"*), and expected performance (*"Big-Theta"*). To evaluate the software performance of my code, I choose to analyze its worst-case complexity using Big-O notation (Mala & Ali 2022). The decision to use Big-O notation is driven by the large parameter space for the synthetic CMD generator, for which most of the use cases in this paper held many input parameters constant, including distance, metallicity, extinction, and stellar atmosphere model.

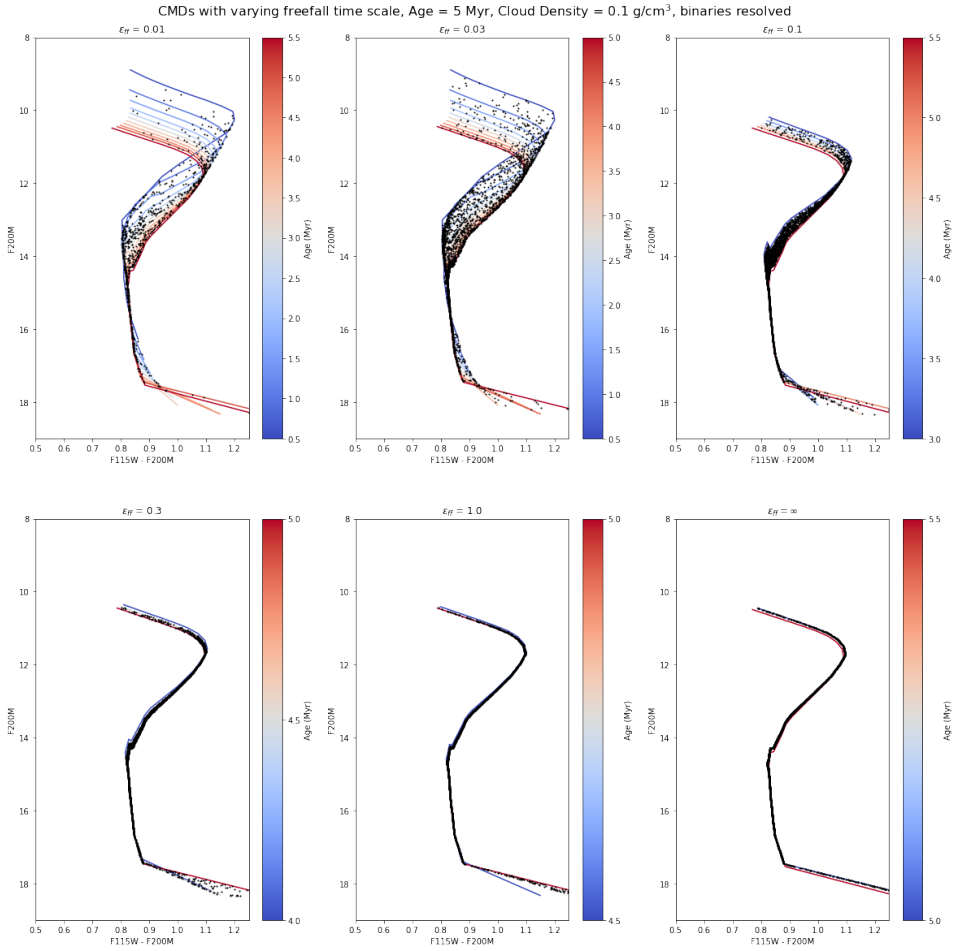

Figure 15. CMD grid showcasing magnitude spread in response to age spread introduced by the freefall timescale. Something to note is that there appear to be fewer stars in the CMD for $\epsilon_{ff} = 0.01$ due to the low rate of star formation.

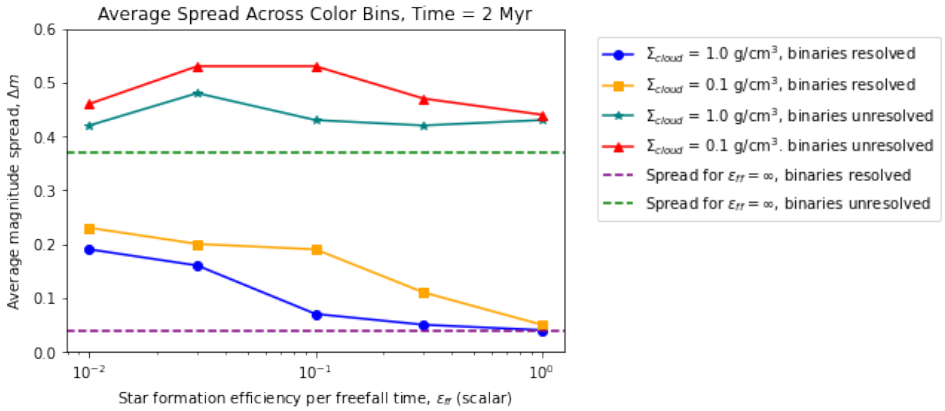

Figure 16. Average magnitude spread across color bins as a function of freefall timescale, for a cluster time of 2 Myr. Magnitude spread appears to be larger for longer freefall timescale, lower cloud density, and cases in which binaries are treated as unresolved.

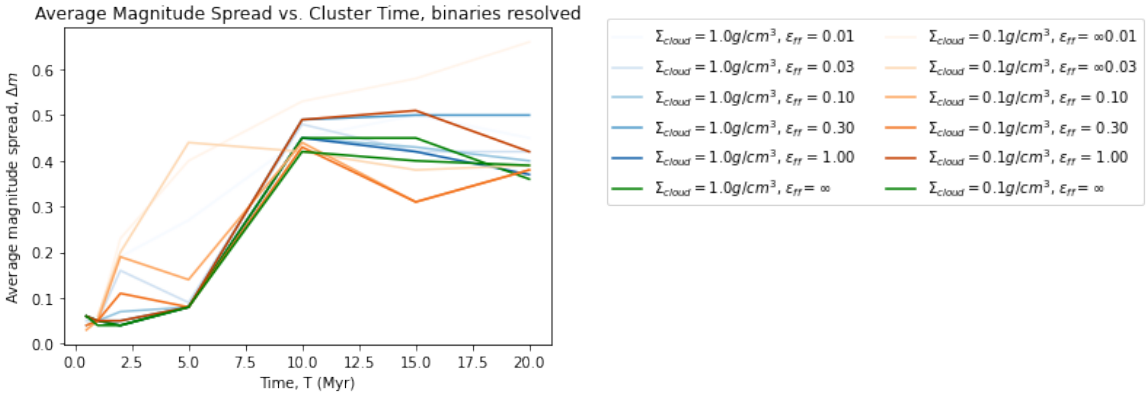

Figure 17. Plot of magnitude spread as a function of cluster time for all possible values for cloud density and freefall timescale. The curves diverge from the expectation of being monotonically increasing due to the spread for metric losing its efficacy as isochrone curves migrate and change shape with cluster age.

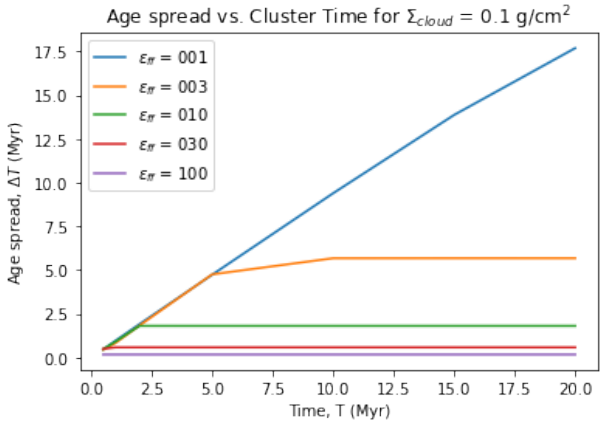

Figure 18. Plot of age spread as a function of cluster time for each freefall timescale for a cloud of density 0.1 g/cm^3. This plot emphasizes the positive relationship between age spread and cluster time, which plateaus as star formation halts.

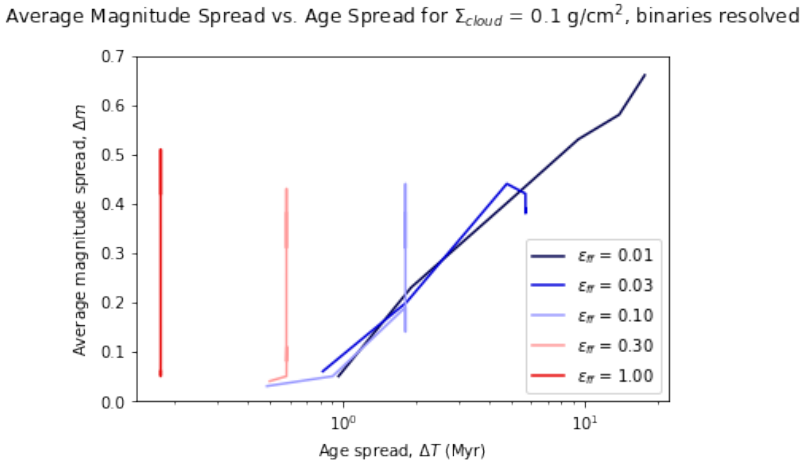

Figure 19. Plot of magnitude spread as a positive function of age spread. The curves also demonstrate the relationship between age spread and the rate of star birth as they elongate with longer freefall times.

Various orders of Big-O complexity and accompanying example algorithms are given:

- $O(1)$ **Constant time:** the time to execute does not depend on the size of the input. Example: accessing a specific index of an array.
- $O(n)$ **Linear time:** the time to execute scales directly with the size of the input. Example: performing a linear search for a particular value in an array using a single for-loop.
- $O(n^2)$ **Quadratic time:** the time to execute scales with the size of the input to a power. Considered to be inefficient, and faster solutions are more desirable. Example: searching for a pair of matching values in an array using nested for-loops.
- $O(2^n)$ **Exponential time:** the time to execute doubles when the size of the input increases by one element. This is considered the worst time complexity and is often the time complexity for brute-force algorithms. Example: a password-cracking algorithm that successively picks combinations of alphanumeric characters until successful.

When performing Big-O analysis, lower-order terms are dropped, leaving only the highest-order term. Consider a program that searches for a pair of matching values as described in our $O(n^2)$ example, but it then iterates through the array again to add that value to all other array elements, shown in Listing 1. This particular program has a nested for-loop that iterates through an array of size n for each element in that array. It is then followed by a single for-loop which iterates through the same array. Thus, the full expression for time complexity would be $O(n^2 + n)$, which then simplifies down to $O(n^2)$. Additionally, Big-O notation discards any coefficients not related to the input size. For example, a program that does three linear searches on the same array would be $O(3n)$, which reduces to $O(n)$.

The time complexity of the entire process of generating a synthetic CMD from start to finish can be broken down into three major components, in ascending order of expected time to complete:

1. The code that I wrote to extract Nbody data, interpolate points, and generate the diagram.
2. The generation of the isochrone grid using SPISEA.
3. The protocluster formation simulation in Nbody6.

A code review of the synthetic CMD generator reveals that its time complexity is dependent on two inputs: the size of the Nbody cluster table (the number of stars in the simulation) n and the number of points in the isochrone curves p (which is model-dependent). For both input vectors, the generator performs several iterative for-loops as it reads simulation data, interpolates stars, and plots both the isochrone curves and the stellar population. For the worst case, an isochrone is generated for every possible value of age in the Nbody data table, which is upper-bounded by n. If the isochrone files already exist, the program only needs to read the files, which is linear in p. Thus, the worst-case for generating a diagram is the maximum possible number of age values times the number of points in the isochrones, or $O(np)$. Interpolating and plotting the stars, regardless of whether binaries are resolved or unresolved, is linear in n. In total, the simplified Big-O notation for reading the data files and generating a diagram is $O(np + n)$.

```java
/**
 * Searches array for a pair of matching values then adds that value to every other array element
 *
 * Precondition - array has all positive values
 *
 * @param array - Array of integers
 * @return array - Array of integers at the end of execution
 */
public int[] addDuplicateElement(int[] array) {

    // Stores the value of matching array elements
    int match = 0;

    // Scan array to find a pair of matching values
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (array[i] == array[j])
                // Two matching array elements are found, store the value in match
                match = array[i];
        }
    }

    // Add value to all other array elements, if none found, match is still 0 and the arrray is unaffected
    for (int i = 0; i < n; i++)
    {
        array[i] += match;
    }

    return array;
}
```

Listing 1. Java example of $O(n^2)$ complexity code. The nested for-loop is $O(n^2)$ and the single for-loop is $O(n)$, leaving us with a full expression $O(n^2 + n)$ which simplifies down to $O(n^2)$.

Assuming the isochrone curves have already been generated, the diagram generator executes quite quickly. If the data files containing the values for the points that make up the isochrone curves are not present in the same directory as the CMD generator, then it will generate an isochrone grid from scratch using the SPISEA constructor. A review of the SPISEA code[1] finds that the most expensive part of generating isochrone curves is the routine for calculating the synthetic photometry, which features a series of nested for-loops dependent on the number of filters and the number of points in the isochrone curves. We assume the number of filters to be constant, so the time complexity of SPISEA is $O(n)$. When we include isochrone generation in the time complexity analysis, with the assumption that the Nbody data table has already been generated, the Big-O notation does not change and is still $O(np + n)$.

If one were to consider running the Nbody6 simulations as a part of the synthetic CMD generation process in addition to isochrone generation, then the time complexity of Nbody6 is added to the Big-O notation. According to Wang et al. (2015), Nbody6 has a time complexity of $O(n^2)$, so the full expression for the synthetic CMD generator including SPISEA and Nbody6 is $O(n^2 + np + n)$. This ultimately simplifies down to $O(n^2)$.

In terms of measurable time, the synthetic CMD generator can create a diagram in less than a minute if the Nbody6 simulation data is already generated and so are the required isochrones. If no isochrones have been generated, then the synthetic CMD generator is expected to take a few minutes to an hour. Although SPISEA has a linear time complexity, it takes several minutes to generate an isochrone grid due to how large the evolution model files are. Additionally, the time it takes for the isochrone grid to generate depends on how finely spaced the isochrone grid is. Holding granularity constant, the measurable time to generate the isochrone also depends on the age spread, as a greater age spread will require more isochrones to be generated, which has been established to depend on the cloud density and freefall timescale of the Nbody simulation. Finally, if the Nbody simulation data is not present and needs to be generated, then the user would need to run an Nbody simulation and wait for it to complete, a latency period on the magnitude of days. It is assumed that for most cases, a user would already have several simulation datasets ready to go.

[1] https://github.com/astropy/SPISEA

5. FUTURE WORKS

Future works may consider various improvements to the software, both in improving its scientific accuracy, as well as its general performance and usability.

5.1. *Suggested Scientific Improvements*

Astronomically literate readers would be justified to take issue with the method of interpolation with which photometry was calculated. Values were computed using a series of linear interpolations along magnitudes, which is a logarithmic scale, and thus not entirely in line with the mathematics of the Pogson magnitude system. The main reason for this discrepancy is that the interpolation scheme was written before I learned about the magnitude system in the Intro Astrophysics course that I was taking while working on this project. Despite this, data points that make up the isochrone curves generated by SPISEA are in close enough proximity that interpolating between them does not produce any gross deviations from the expected values. Figure 20 shows the margin of error realized by the current interpolation subroutine. While this may seem like a simple fix, implementing a more photometrically sound method of interpolating along magnitudes would also require implementing a way to identify the filters being used and using the proper zero-point flux associated with said filters. This has not yet been addressed at the time of writing this paper, as the marginal benefit of implementing such a solution did not outweigh the expected cost of time spent programming such a feature.

Several approaches were taken in trying to measure magnitude spread as a function of cluster age, freefall timescale, and parent cloud density. In Masegian (2021), a similar approach was taken by defining temperature bins along the horizontal axis of the H-R diagram and computing the average standard deviation across the color bins. The same geometric approach proved to be more challenging to use in the context of this project for various reasons. Isochrone curves plotting magnitude in the NIRCAM filter F200W against the F115W-F200W color index have regions in which they bend back on themselves, making it difficult to define bins along the horizontal axis, as higher mass stars can fall into the same color bins as intermediate-mass stars. This was not as big of an issue in Masegian (2021), as isochrones on the H-R diagram are relatively straight. The workaround to this issue that was used in this project was to define upper and lower bounds for magnitude, but this introduces a new problem, as isochrones also decrease in mean magnitude with increasing age. Thus, the same bounding box will not always measure spread in the same mass range as a protocluster ages.

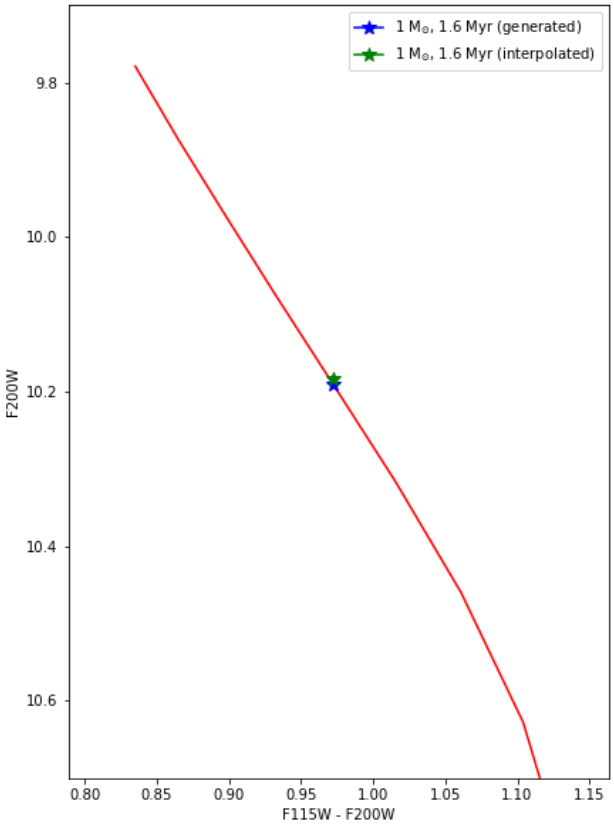

Figure 20. Comparison of interpolated output against generated output. 1.5 Myr and 2.0 Myr Baraffe isochrones were used for synthetic interpolation for a 1 M_{sun}, 1.6 Myr-old star. Then, a 1.6 Myr isochrone was generated to find the expected photometric values of a 1 M_{sun} star. Both points were then plotted on top of the generated isochrone.

Additionally, this approach loses efficacy for higher age models, as the isochrone curves become steeper in slope, causing measurements of spread for older clusters to be increasingly contaminated by the shape of the isochrone curves. Future works might consider probing the effects of age spread on the CMD using an analytical approach as opposed to a geometric approach.

As mentioned, SPISEA is packaged with several evolution models. In generating the synthetic CMDs featured in this paper, a trade-off was made by choosing to use the Baraffe model, which covers quite a small mass range compared to the other models. While the Baraffe model is particularly useful for modeling low-mass stars, as well as binary systems with a low secondary-to-primary mass ratio, its mass range only goes up to 1.4 M_{sun}. Similar to the merged stellar atmosphere model that I used in this project, Hosek et al. (2020) offers an evolution model that merges the Baraffe, Pisa, Geneva, and PARSEC models via a process of linear interpolation in the regions where the mass ranges overlap. A similar merged model using the proprietary extended mass range Baraffe model would prove useful, as it would cover a wider mass range while preserving the Baraffe model's primary strength in modeling low mass ratio binary systems. This would provide a more complete picture of the simulated protoclusters by including the higher mass stars previously left excluded, especially for simulations with higher mass density.

5.2. *Suggested Software Improvements*

Currently, the synthetic CMD generator exists only as proof-of-concept code in Jupyter Notebooks and requires some values to be manually set by the user, namely filter code names and zero-point fluxes. A major driving factor behind the decision to use SPISEA, upon which this project is heavily dependent, was the agility and modularity that it offers as a software package. The benefits that SPISEA offers cannot be fully realized until most, if not all, of the functionality in my code can run programmatically. Consolidating all of the code into a callable Python script would also make for a more elegant user experience, which would only require users to write a few lines of code to use, instead of editing and running multiple Jupyter Notebook cells.

Other additions to the codebase that would maximize usability include creating user documentation. This could be either in the form of a markdown document on the GitHub repository where this code is hosted or as a fully-fledged website, as GitHub provides webpage hosting. Creating a step-by-step example of a CMD generation in Jupyter Notebook would also be helpful in educating users. Another possible addition that would increase usability would be a graphical interface that would allow a user to run the generator without having to write a single line of code.

6. CONCLUSION

I introduce a software solution that generates both Hertzsprung-Russell and Color-Magnitude Diagrams, capable of modeling binary systems as both resolved and unresolved. The generator does this by interfacing with Nbody6 cluster simulation data to obtain values for the age and mass of each star, which is fed into an interpolation scheme that uses a grid of isochrone curves generated by synthetic photometry software SPISEA, before creating the diagram by plotting the isochrone grid and superimposing the stars on top of the isochrone curves.

I discuss my methods for performing a statistical study on the generated CMDs. I find that in general, magnitude spread is higher for simulations with lower mass density, and for CMDs that plot binary systems as unresolved. Additionally, I find that the spread in

magnitude of a CMD increases with freefall timescale, holding age constant. Finally, I find that the spread also increases with the cluster's age as a direct effect of the stellar population's age spread as older stars dim as they age while new stars are being actively formed. All of these findings are consistent with those found in Masegian (2021)

While far from a finished product, the software in the form of proof-of-concept code is publicly available on GitHub[2].

7. ACKNOWLEDGEMENTS

I thank my parents for the support they provided me in life which has led up to me writing this thesis. Although I have already included them in the acknowledgments section of my other senior thesis for the School of Engineering, all the acknowledgments sections in the world could not begin to thank them enough. I thank my cat, Segmentation Fault, for being a general nuisance while I worked on this project, as she would spend a lot of time standing in front of my monitor and sitting on my keyboard while I wrote code. I think discovered at least five or six new hotkeys this year because of her.

I thank my high school chemistry teacher Anthony Petras and my high school computer science teacher Steven Barber, both of whom played a vital role in shaping my worldview and my relationship with science. I thank the new friends I have made as an astronomy student, whose company made two semesters somehow feel so long yet fly by at the same time. I thank my astronomy professors Dr. Matthew Pryal, Dr. Ed Murphy, Dr. Mark Whittle, and Dr. Steve Majewski for their support in my short time as an astronomy student. My decision to switch majors in my final year as a college student was not an easy one, but one that I do not regret in the slightest, as the sense of connection I have made with them is unlike any I have experienced in either the Computer Science or Economics departments.

Finally, I thank Dr. Jonathan Tan, Aayush Gautam, Dr. Juan Farias, and Dr. Matt Hosek Jr. for their assistance along every step of this project. Without them, none of what was described in this paper would have been possible.

[2] https://github.com/zangston/synthetic-hr

REFERENCES

Baraffe, I., Homeier, D., Allard, F., & Chabrier, G. 2015, Astronomy and Astrophysics, 577. https://doi.org/10.1051/0004-6361/201425481

Bressan, A., Marigo, P., Girardi, L., et al. 2012, Monthly Notices of the Royal Astronomical Society, 427, 127. https://doi.org/10.1111/j.1365-2966.2012.21948.x

Castelli, F., & Kurucz, R. L. 2004, Proceedings of the IAU Symposium, doi: https://doi.org/10.48550/arXiv.astro-ph/0405087

Choi, J., Dotter, A., Conroy, C., et al. 2016, The Astrophysical Journal, 823, doi: 10.3847/0004-637X/823/2/102

Ekström, S., Georgy, C., Eggenberger, P., et al. 2012, Astronomy and Astrophysics, 537. https://ui.adsabs.harvard.edu/link_gateway/2012A&A...537A.146E/doi:10.1051/0004-6361/201117751

European Space Agency. 2007, Hertzsprung-Russell Diagram. https://www.eso.org/public/images/eso0728c/

Farias, J. P., Tan, J., & Chatterjee, S. 2019, MNRAS, 483, 4999, doi: 10.1093/mnras/sty3470

Hertzsprung, E. 1908, Astronomische Nachrichten, 179, 71, doi: 10.1002/asna.19081792402

Hertzsprung, E. 1911, Publikationen des Astrophysikalischen Observatoriums zu Potsdam, 63

Hosek, Jr., M. W., Lu, J. R., Lam, C. Y., et al. 2020, The Astronomical Journal, 160, 143, doi: 10.3847/1538-3881/aba533

Hughes, D. 2006, Journal of Astronomical History and Heritage, 9, 173. https://ui.adsabs.harvard.edu/abs/2006JAHH....9..173H/abstract

Husser, T.-O., Wende-von Berg, S., Dreizler, S., et al. 2013, AA, 553, A6, doi: 10.1051/0004-6361/201219058

Koester, D. 2010, Mem. Soc. Astron. Italiana, 81, 921

Kroupa, P. 2001, Monthly Notices of the Royal Astronomical Society, 322, 231. https://doi.org/10.1046/j.1365-8711.2001.04022.x

Lada, C., & Lada, E. 2003, Annual Review of Astronomy and Astrophysics, 41. https://doi.org/10.1146/annurev.astro.41.011802.094844

Lam, C., Lu, J., Hosek, Jr., M. W., Dawson, W., & Golovich, N. 2020, The Astrophysical Journal, 889, doi: 10.3847/1538-4357/ab5fd3

Mala, F., & Ali, R. 2022, Journal of Applied Mathematics and Computation, 6, 1, doi: 10.26855/jamc.2022.03.001

Masegian, A. 2021. https://www.amasegian.com/wp-content/uploads/2022/09/A-Search-for-Age-Spreads-in-Synthetic-Hertzsprung-Russell-Diagrams-of-Young-and-Forming-Clusters.pdf

McKee, C., & Ostriker, E. 2007, Annual Review of Astronomy and Astrophysics, 45. https://doi.org/10.1146/annurev.astro.45.051806.110602

Miles, R. 2007, Journal of the British Astronomical Association, 117, 172. https://adsabs.harvard.edu/full/2007JBAA..117..172M

Milwaukee Astronomical Society. 2024, Beginner's Guide: The Stars. https://milwaukeeastro.org/beginners/stars.asp

Min, Y. 2008, Stellar Birth. https://people.astro.umass.edu/~myun/teaching/a100_old/longlecture14.html

National Aeronautics and Space Administration. n.d., Chapter 6: Electromagnetics. https://science.nasa.gov/learn/basics-of-space-flight/chapter6-1/

Papadimitriou, C. 2014, PNAS. https://doi.org/10.1073/pnas.1416954111

Russell, H. N. 1914, Popular Astronomy, 22, 275

Schombert, J. 2015, Color Index. https://pages.uoregon.edu/jschombe/glossary/color_index.html

The Editors of Encyclopedia Britannica. 2021, Encyclopedia Britannica. https://www.britannica.com/science/Hertzsprung-Russell-diagram

—. 2024, Encyclopedia Britannica. https://www.britannica.com/science/Plancks-radiation-law

Tognelli, E., Prada Moroni, P. G., & Degl'Innocenti, S. 2011, Astronomy and Astrophysics, 533, doi: https://doi.org/10.1051/0004-6361/200913913

Wang, L., Spurzem, R., Aarseth, S., et al. 2015, Monthly Notices of the Royal Astronomical Society, 450, 4070, doi: 10.1093/mnras/stv817