Stars Final Report

Alexander Fogal, 20721508

April 26, 2021

1 Introduction and Background

The goal of this project was to numerically simulate a star, and thereby produce a main sequence Hertzsprung-Russel (HR) diagram, as well as Luminosity-Mass and Mass-Radius relationships over the main sequence. A description of the numerical methods will be given in section 2, and the figures can be found in section 3. Furthermore, two specific stars with mass $M < 0.75M_\odot$ and $M > 2M_\odot$ were simulated, and various quantities related were graphed. These can be found in section 4.

1.1 Stellar Structure Equations

The first equation of concern is Hydrostatic Equillibrium (HSE), which was manipulated to yield the radial gradient of density in terms of the partials of pressure:

$$\frac{dP}{dr} = -\frac{GM\rho}{r^2} \implies \frac{d\rho}{dr} - \left[\frac{GM\rho}{r^2} + \frac{\partial P}{\partial T}\frac{dT}{dr}\right] \Big/ \frac{\partial P}{\partial \rho} \tag{1}$$

This was the equation used to determine the density of the simulated stars, where ρ is the density, r is the radial distance from the centre, G is Newton's gravitational constant, M is mass, T is temperature, and P is pressure. Note that all equations listed herein are borrowed from Broderick[1] unless otherwise mentioned. This was supplemented with the energy transport equation (2), the definition of enclosed mass (3), the luminosity equation (4), the opacity equation (5), and the energy generation equation (6):

$$\frac{dT}{dr} = -\min\left[\frac{3\kappa\rho L}{16\pi acT^3r^2}, (1-1/\gamma)\frac{T}{P}\cdot\frac{GM}{r^2}\rho\right] \tag{2}$$

$$\frac{dM}{dr} = 4\pi r^2\rho \tag{3}$$

$$\frac{dL}{dr} = 4\pi r^2\rho\epsilon \tag{4}$$

$$\frac{d\tau}{dr} = \kappa\rho \tag{5}$$

$$\epsilon = \epsilon_{PP} + \epsilon_{CNO} \tag{6}$$

Equations (1)-(5) form the system that was integrated, which were supplemented by equations (6)-(16) which will be described in a moment. As well as the variables previously defined, we also have that κ is the opacity, L is luminosity, γ is the adiabatic constant, ϵ is the energy production, and τ is the optical depth. The studious reader might note that several quantities remain undefined thus far. First, we must settle on a pressure regime or equation of state. There are three cases that must be considered here: the non-relativistic degenerate pressure, the ideal gas pressure, and the photon gas pressure.

We approximate the total as the sum of the three, as one will typically dominate in the related regime as mentioned in Broderick[1]:

$$P = \frac{(3\pi^2)^{2/3}}{5} \cdot \frac{\hbar^2}{m_e} \left(\frac{\rho}{m_p}\right)^{5/3} + \rho \frac{kT}{\mu m_p} + \frac{aT^4}{3} \tag{7}$$

$$\mu = (2X + 0.75Y + 0.5Z)^{-1} \tag{8}$$

We also defined the mean molecular weight for a fully ionized gas, μ, which we will consider approximately constant. The other variables were m_e, the mass of the electron, $\hbar$, the reduced Planck constant, m_p, the mass of the proton, and k, Boltzmann's constant. In addition to this, we had the Hydrogen fraction $X = 0.734$, the Helium fraction $Y = 0.25$, and the metallicity fraction $Z = 0.016$, where the values were borrowed from Broderick[2]. We can now also define the partials of pressure which appear in the HSE equation (1):

$$\frac{\partial P}{\partial \rho} = \frac{(3\pi^2)^{2/3}}{3} \frac{\hbar^2}{m_e m_p} \left(\frac{\rho}{m_p}\right)^{2/3} + \frac{kT}{\mu m_p} \tag{9}$$

$$\frac{\partial P}{\partial T} = \rho \frac{k}{\mu m_p} + \frac{4}{3} a T^3 \tag{10}$$

Now, for the energy generation, we consider only the Proton-Proton (PP) chain and Carbon-Nitrogen-Oxygen (CNO) cycle, which we approximate using power laws:

$$\epsilon_{PP} = (1.07 \times 10^{-7}) \rho_5 X^2 T_6^4 \tag{11}$$

$$\epsilon_{CNO} = (8.24 \times 10^{-26}) \rho_5 X X_{CNO} T_6^{19.9} \tag{12}$$

Where we have defined reduced density $\rho_5 \equiv \rho/10^5$ and reduced temperature $T_6 \equiv T/10^6$, and used the solar CNO abundance $X_{CNO} = 0.03X$. As for opacity, we consider and combine three main sources of opacity: electron scattering, free-free interactions, and H^- interactions. The approximate Rosseland mean opacities are:

$$\kappa_{es} = 0.02(1 + X) \tag{13}$$

$$\kappa_{ff} = (1.0 \times 10^{24})(Z + 0.0001) \rho_3^{0.7} T^{-3.5} \tag{14}$$

$$\kappa_{H^-} = (2.5 \times 10^{-32})(Z/0.02) \rho_3^{0.5} T^9 \tag{15}$$

Where we defined another reduced density $\rho_3 \equiv \rho/10^3$, similar to before. An approximate combination of these is given by:

$$\kappa = \left[\kappa_{H^-}^{-1} + \max(\kappa_{es}, \kappa_{ff})^{-1}\right]^{-1} \tag{16}$$

As per the discussion in Broderick[1], this works because at high temperatures, the main driver of opacity is the free electron sea, which means the opacity is determined by the maximum of electron scattering and free-free interactions. As we go further from the centre, hydrogen that is cool enough recombines, and then the opacity becomes dominated by the presence of H^- ions. We neglect conduction here.

This concludes the discussion of equations used to simulate a main sequence star, we now have a total of 16 interconnected equations that we must somehow integrate. The equations (1)-(5) form a system of coupled differential equations (DEs), and so this is the focus of the next section on boundary conditions for these DEs.

1.2 Boundary Conditions for the Stellar Structure Equations

We will begin by defining the boundary conditions in the most natural way, with some defined at the centre of the star, and some defined at the surface of the star. First, we have the rather obvious boundary conditions for mass and luminosity:

$$M(0) = 0 \quad \text{and} \quad L(0) = 0 \tag{17}$$

And the surface boundary conditions for the radius and surface luminosity:

$$\tau(\infty) - \tau(R_*) = \frac{2}{3} \quad \text{and} \quad L(R_*) = 4\pi\sigma R_*^2 T_*^4 \tag{18}$$

Where we have let the subscript q_* represent each quantity at the surface of the star, which is defined by the first boundary condition in (18). These boundary conditions allow us to specify all but one parameter, which is used to parameterize the main sequence. Here we have chosen that this should be:

$$T(0) = T_c \tag{19}$$

Which allows us to determine rho_c as well. This concludes the discussion of boundary conditions. The following section will detail how the actual simulation was accomplished from the amalgam of the above equations and boundary conditions.

2 Numerically Simulating a Star

This section will outline the steps taken to simulate a star, which begins with a discussion of how the numerical integration of the system (1)-(5) was performed in subsection 1, continues with a discussion of how trial solutions were assessed in subsection 2, and finishes with a discussion of generating trial solutions in subsection 3.

2.1 On Numerical Integration: RK45

In order to actually integrate the system defined by equations (1)-(5) and the boundary conditions (17)-(19), we had to define an integration scheme. Originally, the `solve_ivp()` function provided by `scipy.integrate` was used for integration, which provided a prepackaged Runge-Kutta 4(5) (RK45) method with adaptive stepsizing, but we found this method to be quite slow, and it did not allow for the implementation of ceasing integration when a function evaluates to a certain value, which will come up later. Using that function, we had to brute force to make sure integration went far enough. So instead, a customized RK45 with adaptive stepsizing was implemented, according to Felhberg[3]. A function called `rk45_step()` was implemented in a loop, such that the function was provided the current stepsize, the system, the current radius, and a tolerance, and returned the next stepsize and the value at the current point. This method was chosen because it allowed us to integrate the whole system at once, was fast, and had minimal error, the latter of which were due to the aforementioned adaptive stepsizing. We added two tweaks to this method in order to produce more accurate solutions.

The first tweak was that when the temperature dropped below 10% of the core temperature, we enforced the use of the minimum stepsize, which allowed for much higher resolution at the surface, where it mattered the most. The second tweak added was the use of an *opacity proxy*, as suggested in the outline document. The opacity proxy is given by:

$$\tau(\infty) - \tau \approx \delta\tau \equiv \frac{\kappa\rho^2}{|d\rho/dr|} \tag{20}$$

The integration loop continued until the opacity proxy took a value less than 0.1, or a mass limit of $10^3 M_\odot$, or a surface limit of $20R_\odot$, which helped limit the integration time as well, while making sure that $\tau(\infty)$ was indeed far enough. We used a maximum (and initial) stepsize of 10,000m, and a minimum stepsize of 5,000m.

2.2 On Evaluating a Trial Solution

We will now briefly discuss how we evaluate the quality of a trial solution. In order to connect the boundary conditions defined at the surface with those defined at the core, we need to actually perform the integration. The only issue is, we need some way to actually assess how good our choice of boundary conditions was in the first place! This was accomplished by a condition on luminosity: an error term that compares the luminosity at our defined surface with the expected luminosity given the location of the surface and surface temperature. The fractional error is given by:

$$f(\rho_c) = \frac{L_* - 4\pi\sigma R_*^2 T_*^4}{\sqrt{4\pi\sigma R_*^2 T_*^4 L_*}} \tag{21}$$

Note that this should converge on $f = 0$ for a perfect match. In practise, this does not always occur, and so we set the boundary condition to zero by hand by choosing the surface temperature to match after minimizing this error term.

2.3 On Producing Trial Solutions

The general outline of producing a trial solution is as follows:

1. Set T_c and perform bisection to find ρ_c
2. Integrate the system to get R_*, L_*, T_*
3. Compare the error to a condition on luminosity
4. Repeat until the error converges.

The structure of the code corresponding to the above outline is as follows: the `mainsequence()` function has a loop over a range of central temperatures roughly corresponding to a main sequence. This loop calls a function `find_ics()`, which takes an upper and lower bound for the range of central density, the current core temperature, and a guess for the limit of integration, as well as the system and a boolean for the modified system, which will be expanded upon later. The function `find_ics()` performs a naive bisection according to Newman[4] on the mentioned error condition.

Bisection is a generalization of binary search for doing root-finding, wherein there is an upper bound and a lower bound, one of which has a negative evaluation and one of which has a positive evaluation. According to the continuity theorem, if we have an interval bounded by a negative and positive point, we are guaranteed that there is a root contained in that interval. Bisection checks that one endpoint is positive and one is negative, and then calculates the error at the midpoint, and then takes the midpoint and the endpoint with opposite sign and enters a recursive loop. In this way, the endpoints are moved gradually closer to the root of the error.

The function `find_ics()` obviously requires that one endpoint has positive error, and the other has negative error. The `mainsequence()` function was written with checks to ensure that if one bound is not the opposite sign of the other, then it tries different bounds. The `find_ics()` function has a multitude of exit conditions, as this is where the bulk of the slowdown occurs. The exit conditions were: if the error is less than 3×10^{-5}, if the error is less than 1×10^{-4} and there has already been 20 iterations, if the error is less than 1×10^{-2} and there has been 50 iterations, the error is less than 1 and the difference of recent errors is less than 1×10^{-4}, if the difference in recent densities is less than 1×10^{-10} and the difference of errors is as previously mentioned and there is over 20 iterations, and finally, if there has been 60 iterations. In this way, we minimized the amount of time spent on bisection.

In order to assess the error in a solution, `find_ics()` calls a function called `trial_soln()`. This function takes the current value of central density, the current core temperature, the guessed limit of integration, the system, and a `modified` boolean and returns an array with the full range of each of the relevant quantities, and importantly, the error we are attempting to minimize. These values are stored for the current error closest to zero and are returned by `find_ics()` when one of the exit conditions is tripped.

As for the function `trial_soln()` itself, this function sets variables for the boundary conditions, loops over the `rk45_step()` as mentioned, interpolates all the relevant quantities, determines the surface from an interpolated $\tau(\infty) - \tau$ evaluated at 2/3, and then determines the error. All interpolation was done using `scipy.interpolate.interp1d()`. We chose $\tau(\infty)$ by simply taking the last non-`NaN` value in the array. This function uses slightly different boundary conditions than previously noted, as some of the DEs diverge at zero. For this reason, we chose the integration to actually start at $R_c = 10m$, which was much smaller than any relevant length scales. So the boundary conditions defined at zero before were redefined as follows:

$$T(R_c) = T_c \quad M(R_c) = \frac{4}{3}\pi R_c^3 \rho_c \quad L(R_c) = \frac{4}{3}\pi R_c^3 \rho_c \epsilon(\rho_c, T_c) \quad \tau_c = \kappa(\rho_c, T_c)\rho_c \tag{22}$$

After all of this is complete, the function returns an array of values containing radii, densities, temperatures, masses, luminosities, optical depths, the surface radius in metres, and the error.

The mentioned `modified` boolean will now be discussed. As the group project involved a modification to the stellar structure equations, we implemented a boolean to keep track of this, and to make sure the right functions and values are used at the right time. The modification here was a modified opacity, which included an extra term. The modified opacity equations are given by:

$$\kappa_i = \begin{cases} \kappa_0 (Z + 0.0001)\rho_3^{0.5} T^{-3.5} & T < T_i \\ 0 & T \geq T_i \end{cases} \tag{23}$$

$$\kappa = \left[\kappa_{H^-}^{-1} + \max(\kappa_{es}, \kappa_{ff}, \kappa_i)^{-1}\right]^{-1} \tag{24}$$

Where a range of values for κ_0 and T_i were covered, which was done simply by setting the corresponding variables globally and then calling `mainsequence()` multiple times. The motivation for this was that we were experimenting with a new source of opacity, some species that ionizes at temperature T_i and so only contributes to the opacity below that temperature. In order to accomplish this, the `modified` boolean was passed around to make sure that `kappa_modified` was used in all the relevant locations.

The final algorithmic choice to discuss was the plotting section. We created plots using the `matplotlib.pyplot` package simply by calling `pyplot.plot` function and various other modifications like labels and titles. Nothing complicated there, we had a function `plot_all()` for encapsulation, but all it was was a sequence of plotting calls. Technically, the HR diagram, L-M, and M-R plots were done *in* the `mainsequence()` function, which was done in order to have it replot and save after every star, so that progress could be assessed.

3 Results

This section focuses on the results of the code detailed above. The first subsection focuses on the results of the main sequence simulation, and the second subsection focuses on two specific stars.

3.1 Main Sequence Results

Simulating the entire main sequence took us about 2 hours in totality. The first result is the HR diagram, which shows surface temperature and surface luminosity on a log-log plot:

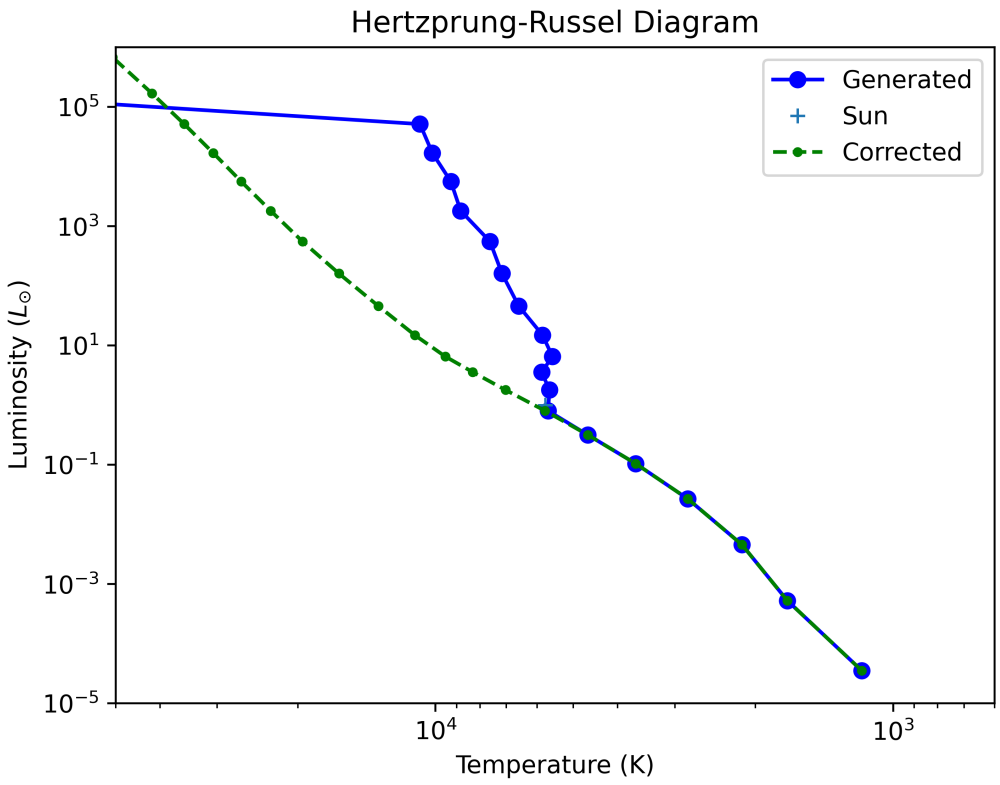

Figure 1: The simulated HR diagram. The blue line is the result of the code, the green line is using the hand corrected surface temperature as mentioned in section 2.2. Note the location of the sun on the diagram falls exactly on the line as would be expected.

The HR diagram produced resembles both the diagram given in Broderick[1][2] and diagrams that can be found online, which is a testament to the correctness of our results! Next we present our graph for $L/L_\odot$ as a function of $M/M_\odot$:

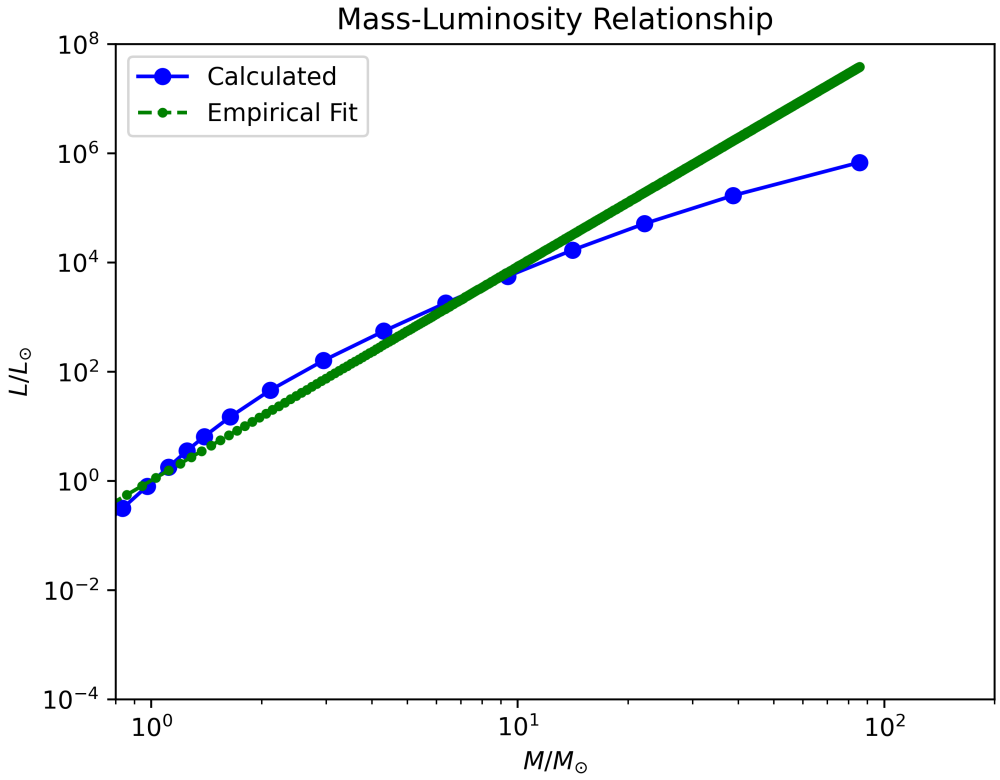

Figure 2: The simulated L-M diagram. The blue line is the result of the code, the green line is the expression given in Broderick[2] taken from Kippenhahn & Weigert[5].

In the Mass-Luminosity graph, the code can be seen to produce a decent match to the empirical relation. It would be unreasonable to expect that these two would perfectly align, as the empirical fit is a power law, and so unable to capture the subtleties of reality. Therefore we conclude that our produced trend is a good match for the power law, which is given by:

$$L/L_\odot = \begin{cases} 0.35\left(\frac{M}{M_\odot}\right)^{2.62} & M < 0.7M_\odot \\ 1.02\left(\frac{M}{M_\odot}\right)^{3.92} & M \geq 0.7M_\odot \end{cases} \tag{25}$$

We finally present our graph for $R/R_\odot$ as a function of $M/M_\odot$:

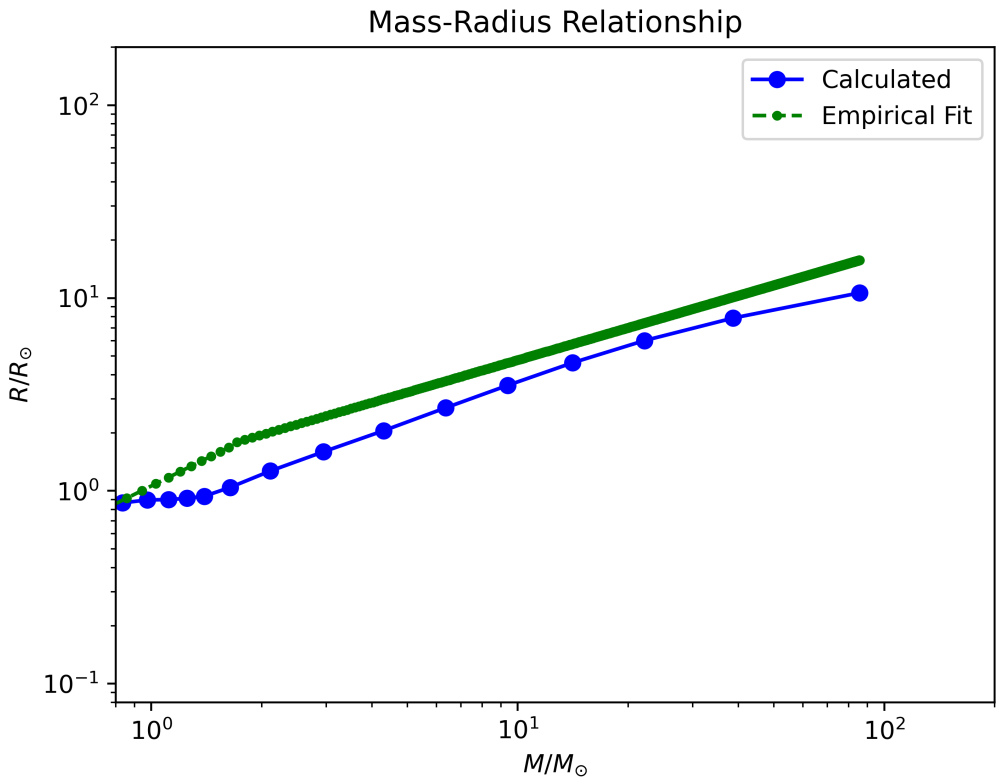

Figure 3: The simulated M-R diagram. The blue line is the result of the code, the green line is the expression given in Broderick[2] taken from Kippenhahn & Weigert[5].

In the Mass-Radius graph, we can once more see that the code produces a decent match to the empirical relation. There is a slight offset here, which we hypothesized to be related to the correction being required and not being applied to anything except the surface temperature. Note that a flat part can be seen on the left, which is due to the change from radiative to convective cores! The empirical relation is given by:

$$R/R_\odot = \begin{cases} 1.06\left(\frac{M}{M_\odot}\right)^{0.945} & M \leq 1.66 M_\odot \\ 1.33\left(\frac{M}{M_\odot}\right)^{0.555} & M > 1.66 M_\odot \end{cases} \tag{26}$$

So in all three cases, the code generated plots that are generally in agreement with the accepted body of work, and the latter two match their respective empirical formulations well.

3.2 Specific Star Results

3.2.1 Star 1

The first star I simulated had mass $0.0824 M_\odot < 0.75 M_\odot$ as required. Its other parameters were: $T_* = 994K$, $\rho_c = 384909 kg/m^3$, $L = 1.2 \times 10^{-5} L_\odot$, and $R_* = 0.1166 R_\odot$. It has the following plots:

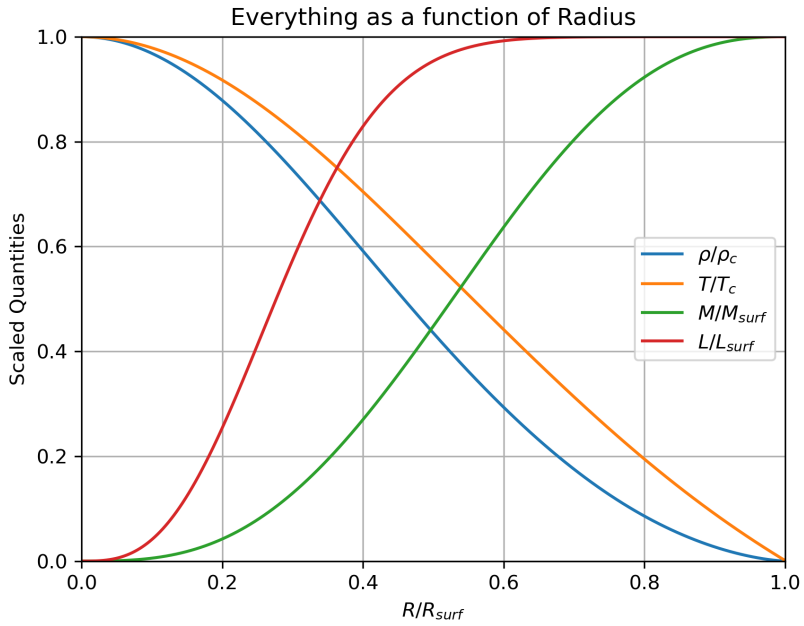

Figure 4: A Plot of density, temperature, mass, and luminosity for the first star.

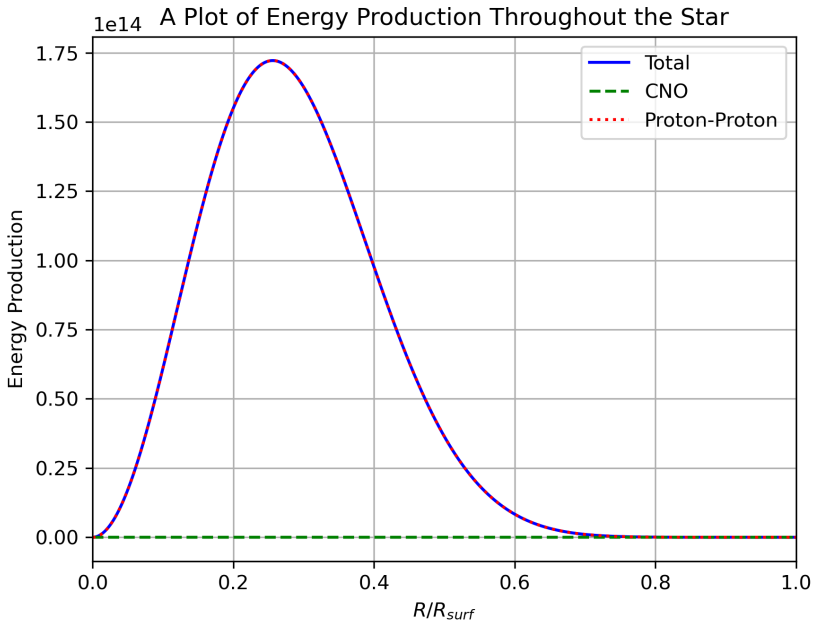

Figure 5: A Plot of the luminosity gradient for the first star.

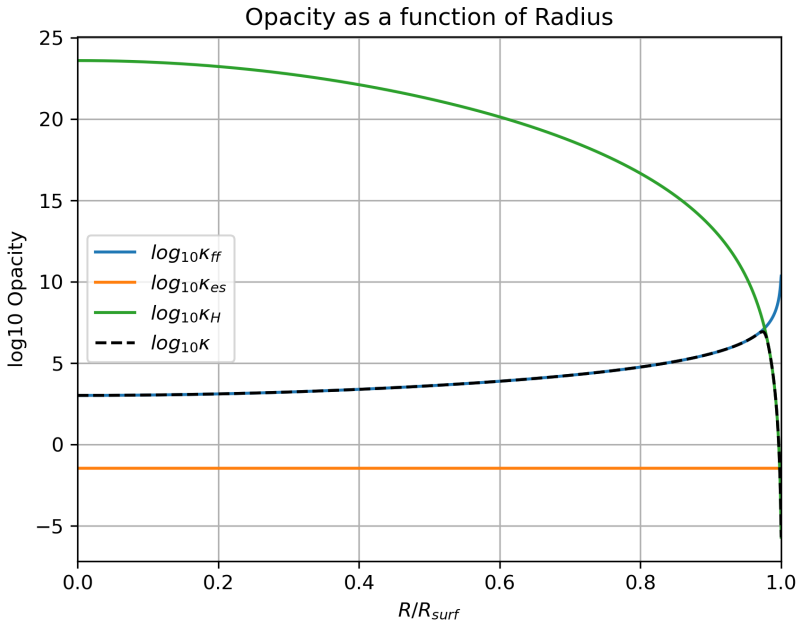

Figure 6: A Plot of opacity for the first star.

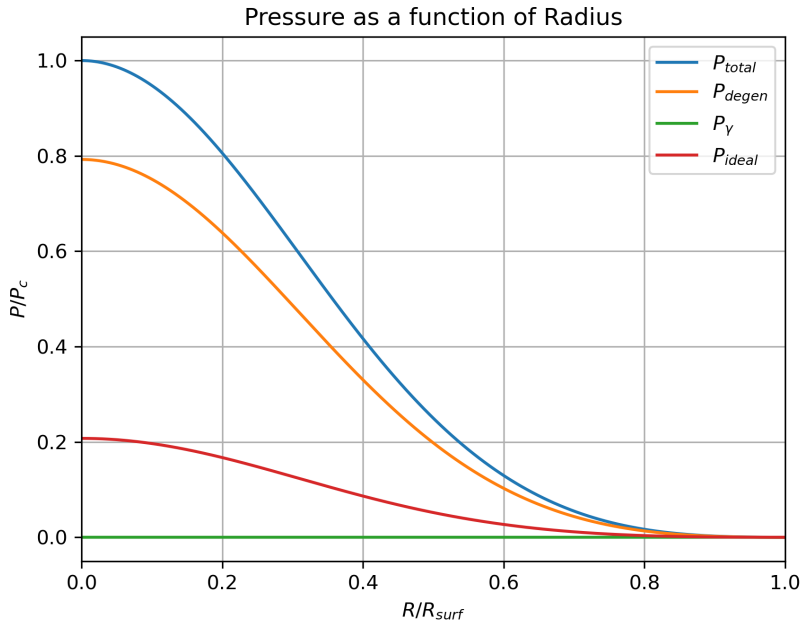

Figure 7: A Plot of pressure for the first star.

3.2.2 Star 2

The second star I simulated had mass $4.89M_\odot > AM_\odot$ as required. Its other parameters were: $T_* = 20643K$, $\rho_c = 16678kg/m^3$, $L = 822L_\odot$, and $R_* = 2.24R_\odot$. It has the following plots:

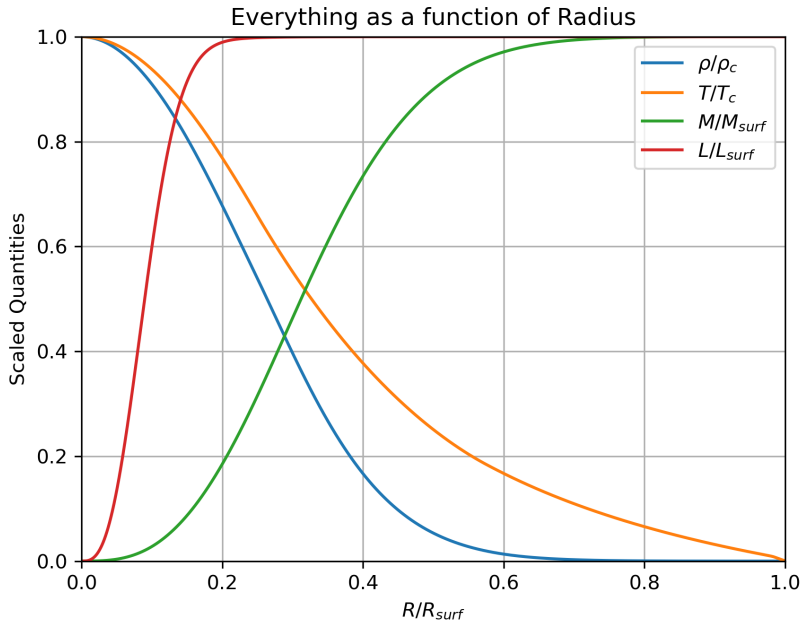

Figure 8: A Plot of density, temperature, mass, and luminosity for the second star.

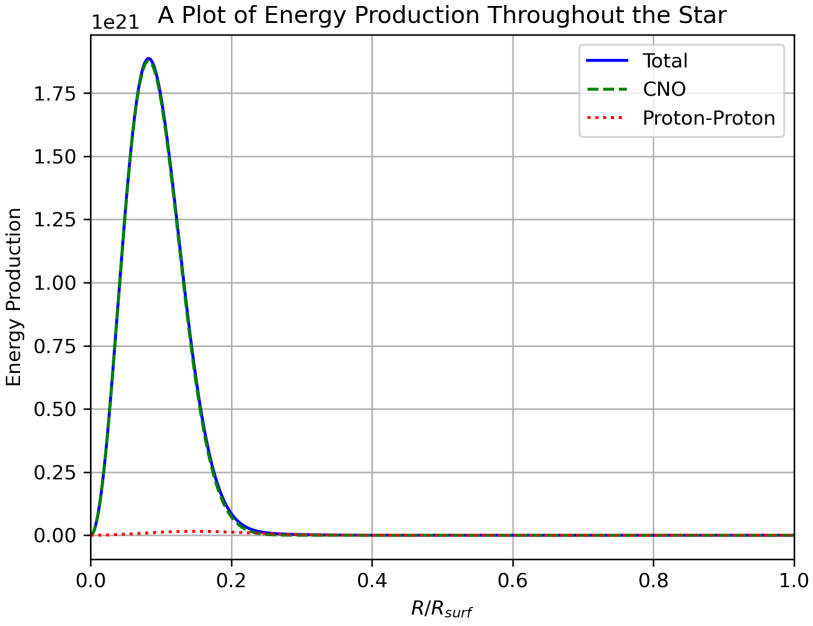

Figure 9: A Plot of the luminosity gradient for the second star.

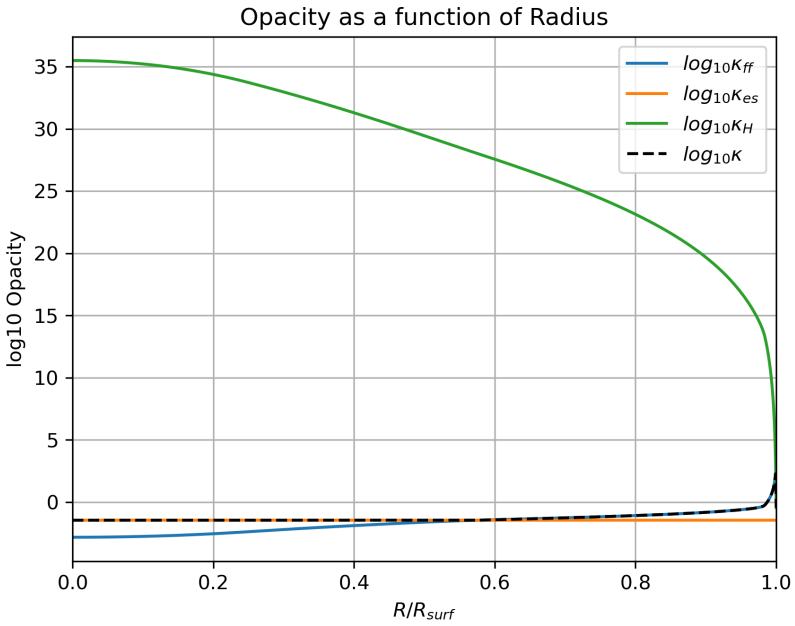

Figure 10: A Plot of opacity for the second star.

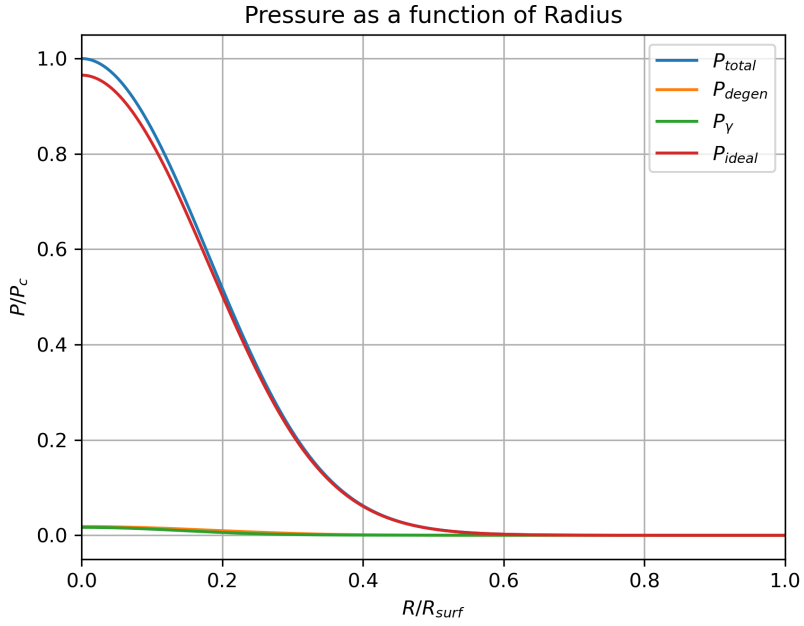

Figure 11: A Plot of pressure for the second star.

3.2.3 Comparison: Convective Zones

In order to find where each star is convective, we must search for where:

$$\frac{d\log P}{d\log T} = (1 - 1/\gamma)^{-1} = 2.5 \tag{27}$$

This is known as the Schwarzchild criterion, which encapsulates the constant fight between convective stability and instability. When the above expression is greater than 2.5, we have convective stability, otherwise, convection occurs. This search was performed numerically in python after simulating each star in totality. For the low mass star, we found that the convective zone was $R/R_* \in [0, 1]$, which is to say the star was fully convective. This makes sense, as the density is quite high compared to the temperature, which makes radiation more difficult. To contrast this, the second star had the convective zone as $R/R_* \in [0, 0.25] \cup [0.97, 1]$. This shows a distinct convective core with a radiative envelope, as well as another convective layer at the surface. The convective core occurs for the reasons mentioned before: high density makes radiation quite difficult to achieve. As the density falls off there is a transition to a radiative envelope, and finally there is a transition back to convection at the outer layer. The outermost convective layer is due to a large temperature decrease near the surface, and a large opacity as well. The two stars are similar that they have convective cores, but the more massive star also has a radiative part surrounding its convective core, where the smaller star is just the convective core itself.

3.2.4 Comparison: Dominant Energy Generation Source

The dominant energy generation source can be determined from figures 5 and 9, which show energy production in the form of dL/dr for each process simulated, as well as the total. We can see that for the low mass star, the majority of energy is generated by the PP chain, which makes sense, as the core temperatures are not high enough for the CNO cycle to be relevant. For the high mass star, we see the exact opposite occurring: the CNO cycle fully dominates the energy production. This makes sense because the core temperature is significantly higher, and so the CNO cycle quickly proceeds from relevant to dominant due to the very large exponent on the temperature. In this way, these two stars are not similar at all, but exactly opposite of each other: one is dominated fully by PP, and the other is dominated fully by the CNO cycle.

3.2.5 Comparison: Dominant Opacities

The dominant opacities in each region can be determined from figures 6 and 10, which show log plots of each opacity as well as the over all opacity over the range of the star. For the low mass star, the dominant opacity through $R/R_* \in [0, 0.96]$ is the free-free interaction opacity. The remaining part of the star is dominated by H^- opacity as would be expected in the region near the surface, where hydrogen is too cool to ionize. For the massive star, the region $R/R_* \in [0, 0.5]$ is dominated by the electron scattering opacity, the region $R/R_* \in [0.5, 0.99]$ is dominated by the free-free opacity, and the final region of the star is dominated by H^- opacity for the reason previously stated. Both stars are largely dominated by the free-free opacity, and both also are dominated by H^- opacity near their respective surfaces. However, the more massive star is dominated by the electron scattering opacity for a large swathe of the range, whereas that opacity never dominates in the smaller star.

4 Conclusion

In conclusion, we were able to fully simulate a star given a surprisingly restricted number of equations. We were then able to analyze a set of specific modifications and present that to the class, and then write up this report. Given the relative simplicity of this model, it is somewhat surprising to see the recreation of observed phenomena like convective cores, convective surface layers, and others. Even more surprising, the features of the main sequence are also accurately simulated here, which was certainly more than could be expected of a handful of equations.

It is now time for the author to stand on his soapbox, so skip this section if it is not relevant. First and foremost, I'd like to say that I had a ton of fun learning about stars for this class. I am normally not very much interested by astronomy topics, but stars are certainly far more fascinating than I could have imagined! So thank you, professor Broderick, for the lectures and opportunity to learn. Furthermore, I found the content of this project quite enjoyable, but the overall experience was extremely stressful. I urge the professor to allow the constituents to select their own groups, and to maybe restrict the group size a little bit. The only reason I bring this up is that I wrote the entirety of the code for my group (about 1k lines in the final state, about 1-2.5k additional lines written/rewritten over time), as well as the bulk of the presentation. I understand that everyone is busy at this time, but I was quite busy as well (I'm never taking 6 courses again), and I regularly found myself staying up to 3-4 a.m. writing and testing code for this project because none of my 6 other group members were not contributing. The current model unfairly encourages a singleton to do the majority of the work, I believe. There is only so much that I could do online to attempt to coerce my group to contribute, maybe this worked better in person for that reason. I should also add that I intended this to be a commentary on the structure of the project, not to throw my own group under the bus (as I mentioned, I understand that everyone is busy, and most people are struggling with the pandemic), and I hope this does not impact their grade at all, but I still felt it necessary to discuss.

Not to end on a sour note, I would like to once again thank professor Broderick and the course staff for what was otherwise a great semester. Thank you, and thank you for reading!

5 References

[1] Broderick Avery, *PHYS 375 Final Project Description*, Posted to Learn, University of Waterloo, March 2021.

[2] Broderick Avery, *PHYS 375 Stars Lectures*, Posted to Learn, University of Waterloo, 2021

[3] Fehlberg Erwin, *Low-order classical Runge-Kutta formulas with step size control and their application to some heat transfer problems*, NASA Technical Report 315, 1969, https://ntrs.nasa.gov/api/citations/19690021375/downloads/19690021375.pdf

[4] Newman Mark, *Computational Physics*, CreateSpace Independent Publishing Platform, 2012. ISBN-13: 978-1480145511.

[5] Kippenhahn Rudolf, Weigert Alfred, Weiss Achim, *Stellar Structure and Evolution*, Springer-Verlag Berlin Heidelberg, 2012. ISBN-13: 978-3-642-30255-8.