Modeling Stars:
Two Method Comparison and Analysis

Najla Alahmadi, Michael Cock, Rebecca Halloran, Brady Metherall, Hector Robinson

March 15, 2017

In this project, we modelled a spherically symmetric, static star by numerically integrating the four stellar equations. We started by doing this for a star that had the same core conditions as the sun, and then changed the intial conditions to determine final values for many other stars. We had to be careful with the initial conditions we used to ensure that the temperature went to zero at the surface still. The initial conditions were valid as long as the temperature decayed as expected. Using the results, we can plot the relation between luminosity and temperature of the stars to create an Hertzpring-Russell Diagram (further refered to as HR diagram). We created code that would be able to take a stars composition, mass and some initial conditions to determine surface temperature, surface luminosity and surface radius so we can plot the surface luminosity versus the surface temperature or an HR diagram. Before we get there, lets talk about our program that will numerically integrate one star.

To perform the calculations we started from the core and integrated outward using the fourth order Runge-Kutta method. We specified a core pressure, core temperature, core density, and composition of the star. From there, the outward integration calculates total mass, luminosity, pressure, and temperature at every other radius. The integration stops when the boundary conditions are met, that is, the pressure goes to zero. Once this point is reached it means we are at the surface, we can take the temperature here, and use it for the HR diagram, Figure 1 shows samples of our code in greater detail.

Figure 2 shows our results using the initial conditions of the core of the sun. These plots are as we would expect for a 1 solar mass star, but they do not get the exact values as expected. One thing that differs from our star to the sun is that it is only radiative for two radius steps, which is about 2×10^5 meters, and convective for the rest. This is in contrast to the sun, where the radiative zone is about 70% of the whole star. We were also unable to incorporate the optical depth so we used the surface temperature instead. The surface temperature should go to zero but our pressure hit zero first triggering the integration to stop. If we were able to use the temperature at a specific optical depth, we would have expected temperatures that produced a better HR diagram. We expect the final values for mass, luminosity, temperature, and pressure to be approximately $1M_{\odot}$, $1L_{\odot}$, 5770 K, and 0 Pa, respectively.

Knowing our model works properly but does not get proper values we can go further and get a general form of the main sequence of an HR diagram. By running this program for a total number of **number of stars** times we obtained the HR diagram shown in Figure 3.

We can see that the shape resembles the main sequence part of the known HR diagram. We can explore a well known program to get a sense of what the HR diagram should look like and compare it to our own model. We will use MESA to do the comparison. MESA will give us a plot of the total life of a star. We can compare where our model shows the stars on the main sequence and we expect the MESA star to be on the main sequence at some point in its life. MESA considers different characteristics that we do not consider in our model. Our model is static, has no rotation and it is only for main sequence stars and MESA will allow for rotation and it does a full life of a star so it is not static and will not only be on the main sequence.

The main goal was to create an HR diagram by simulating a star manually and plotting the data. To do this

```
 8 #Define  all  our  constants.  This  is  where  we  pass  X,  Y,  Z
 9 initialize.init(0.7,0.25,0.05)
10 val = np.logspace(−5., 5., 100)
11 count = 1
12 r_end = 10**10 #in  units  of  m
13
14 for  i  in  val:
15
16      print  i,  "solar  initial  conditions"
17      print  count,  "out  of  100"
18      count += 1
19      # Initial  core  conditions  of  the  star
20      P0 = i*2.477 *10**16. # units  of  pascal
21      M0 = 10**−4 # units  of  kg
22      L0 = 10**−4 # units  of  J/s  or  W
23      T0 = i*1.571*10**7 # units  of  K
24      rho0 = i*1.622*10**5 #units  of  kg/m^3
25
35      data = open('stars.dat','a')
43      P,M,L,T,r,kappa_array,rho_array = RK(10**−6, r_end, 10.**5., P0, M0, L0,
        T0, rho0, dP, dM, dL, dT, rho, kappa, stop, 0.000001)
77      print >> data,  r[−1], M[−1], T[−1], L[−1]
78      data.close()
```

(a) First *main.py* calls *initialize.init* to define our constants, then the core conditions are specified. We then open our data file, run our Runge-Kutta method, and then print the results to the data file.

Figure 1: Important sections of the code.

Figure 2: Simulation results of the Sun.

we needed to complete many sub-parts along the way. The first objective of our team was to get a profile on github and ensure everyone knows how to pull, add, commit and push properly. In the first few tries there were of course a couple troubles where someone forgot to pull right before pushing to ensure their version was up to date with the original. Eventually, everyone was able to pull, add, commit then push a change with no problems. Splitting up work in a fair and convenient way was a small challenge. Working in a big group makes it easier to write the code because we are able to all bring our ideas together and come up with a solution faster. We learned to work as a group in an efficient way, even though our ideas would not work sometimes we were able to listen to everyone and everyone was able to contribute. Different people have different styles of writing code so working in a group of five each person has to learn to read and interpret someone elses code. At the same time, we learned to comment our code so that the rest of our group would be able to understand what had been written and why.
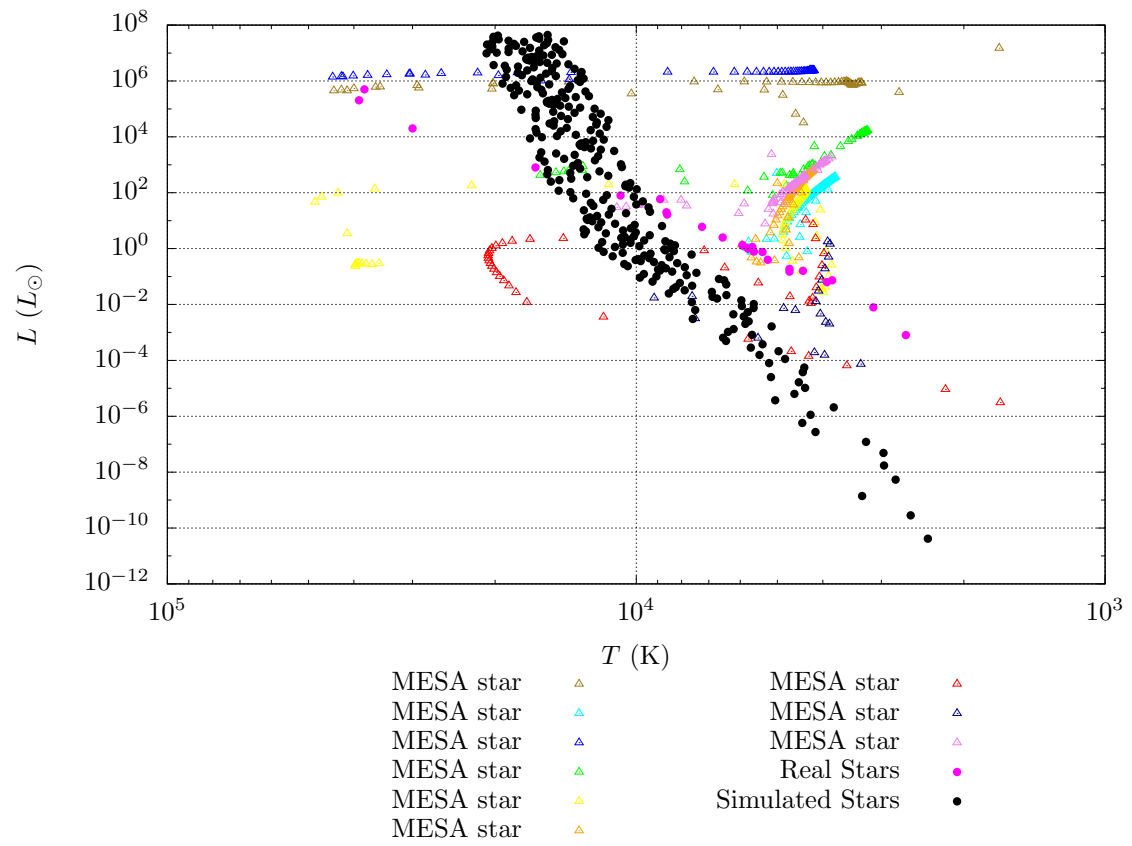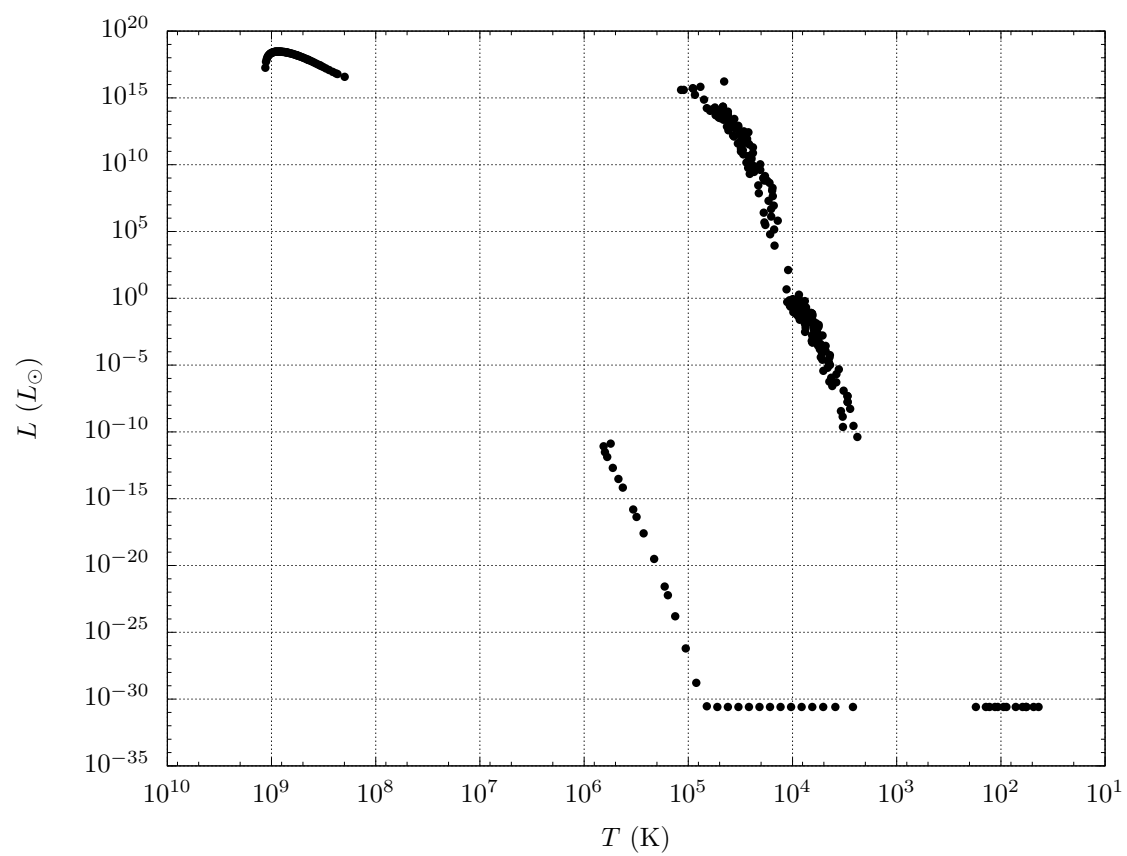
Figure 3: Our reproduction of the HR diagram.

Figure 4: A crappy version of HR