Limitations and Stuff

There is one main limit to our code, and that is the kind of data and amount of data we get from EZ-Web. While EZ-Web is an excellent stellar evolution code, it is limited in that it cannot handle elemental degeneracy or major hydro-dynamical effects, and when the star its evolving runs into either of these things, it stops and only gives data up to that point in time. What does this mean for us? It means that for stars below 2.1 times the mass of the sun, our timeline stops right before the star becomes a white dwarf. For stars above 2.1 times the sun’s mass, our timeline stops before the star starts fusing carbon. While this is somewhat annoying, it’s not failure on us or our project. EZ-Web stops because the conditions and state of the star no longer comply with the already extremely advanced mathematical model it is using. This suggests that if we had tried to take a mathematical approach and not use EZ-Web, we would have run into the same problem as well.

It’s also prudent to note that since we’re using a database, the list of masses available for testing is limited. We solve this by having a selector palate in our UI, but if you just run stellarEvolutionSimulator with some weird value like -2000.57 or something like that, the program will throw an error and continue with an assumed value of one solar mass. The same is true with our timeline bounds.

As for challenges we encountered, we had a lot. The aforementioned mathematical challenge was a doozy, but we also had a bunch of smaller challenges, especially in designing our output.

* It turns out that Mathematica doesn’t like when you try to invert the x-axis and log scale the y‑axis of a graphic at the same time. We had to do a lot of research, and eventually had to use an obscure directive for commands it wasn’t intended for, and modify the data being plotted to fit into those scales. It works, but the directive (ScalingFunctions) appears in red in our code.
* Getting the program to run even close to the speed it does now was also a challenge. We did some research into optimization, and this is why all of our integers have decimal points (this defines them as floating point numbers, which get handled faster) and we use Block instead of module. The article we learned this from is found [here](http://blog.wolfram.com/2011/12/07/10-tips-for-writing-fast-mathematica-code/).
* As mentioned previously, the exterior graphic plotRange used to scale to the maximum radius the star displayed achieved, but this was impractical because for the majority of its life it was approximately 2 orders of magnitude smaller than that value, resulting in an empty box where the sphere should be during those times. We solved this with our sphereRange function.
* Understanding the data as a whole. We consulted multiple times with Dr. Carini, but neither he nor us understood the full extent/limitations of the EZ-Web data until roughly the last week of our development timeline. This was because up to that point, our HR-Diagram was still dysfunctional. It was only when we’d solved that problem that we could understand how far the EZ-Web data went.