1 Introduction

–Initial Problem (graph of obs data)
–First thoughts (Hypothesis)
–What Methods are viable?

2 Method

–Which Method did I use and why?
–General approach (structure of main.py?)
–What does Lumiradius.py do?
For this approach I used two programs: main.py and lumiradius.py.

–Detailed description of main.py[before lumiradius.py?]

At the core of main.py lies a sequence of nested loops that spans a grid in distance-mass-age. The range of these three variables can be easily adjusted, as can the binning. I used a mass range from minmass=$5M_\odot$ to maxmass=$50M_\odot$, distance from 0 to maxdistance=3kpc. maxage is a little more complicated: Because the main sequence age (t_{ms}) of a star is a strictly monotonic increasing function of M the following is true: $maxage = t_{ms}(minmass)$. With a minmass of $5M_\odot$ this translates to maxage=104Myr
The three informations I have about any given star, are its mass, its age and its distance from earth. From these three informations I need to derive its fractional main sequence age (τ) and its apparent magnitude (V) and ultimately the probability density for all stars.
Hurley Pols and Tout published a paper in 2000 in which they approximate the stellar evolution as a function of initial mass (M_{ini}), fractional main sequence age (τ) and metalicity(Z). Now I need to know τ. Using equation 5[citation needed] I know the main sequence age (t_{ms}) and τ then becomes: $\tau = \frac{t}{t_{ms}}$. Since I only include stars on the main sequence, I can safely include the condition: $\tau < 1$ to cut down on computing time. Equation 12 and 13[citation needed] are very powerful equations to compute luminosity and radius for a star on the main sequence.

$$\log \frac{L_{MS}(t)}{L_{ZAMS}} = \alpha_L \tau + \beta_L \tau^\eta + \left(\log \frac{L_{TMS}}{L_{ZAMS}} - \alpha_L - \beta_L \right) \tau^2 - \Delta L(\tau_1^2 - \tau_2^2) \quad (1)$$

$$\log \frac{R_{MS}(t)}{R_{ZAMS}} = \alpha_R \tau + \beta_R \tau^{10} + \gamma \tau^{40} + \left(\log \frac{R_{TMS}}{R_{ZAMS}} - \alpha_R - \beta_R - \gamma \right) \tau^3 - \Delta R(\tau_1^3 - \tau_2^3) \quad (2)$$

[insert dependancies of the variables in those equations]
...insert HRD illustrating lumiradius]

The program does allow for freely changeable metalicities. For my purposes I use Z=0.02 for all stars to simulate a metalicity similar to that of our galactic neighborhood.
I run this program for every possible mass-age tuple and save the results in a matrix This way I don't have to call the program for every distance-mass-age

triple.
With this I now know distance, mass, age, fractional main sequence age, luminosity and radius for any given star. I can now use these informations to compute the apparent magnitudes.

$$M_V = V - 5 \cdot \log_{10}(distance) + 5 - Red \tag{3}$$

$$M_{bol} = M_V + BC \tag{4}$$

$$\frac{L}{L_\odot} = 0.4 \cdot (4.72 - M_{bol}) \tag{5}$$

Where M_V is the absolute visual magnitude, Red is the reddening as a function of distance, M_{bol} is the absolute bolometric magnitude and BC is the Bolometric Correction. Using equations 3, 4 and 5 I can now compute the apparent visual magnitude V:

$$V = 5 \cdot \log_{10}(distance) - 5 + Red + 4.72 - \frac{L}{L_\odot \cdot 0.4} - BC \tag{6}$$

To get an approximation for reddening, I use figure 9 a graphic by AmoresLepine2005, I pick three points in the graph: 1:0.9, 2:2.25 and 3:3.273]

–optimization The binning can be freely adjusted. To optimize runtime of my program I conducted a few tests to adjust the binnings in all three dimensions. [graphics of 50-100-100, 100-50-100, 100-100-50 with caption: this is why I chose 1:2:2]

I execute a pair of nested loops in mass and age to get all the data I need from lumiradius.py and save the results in two dimensional arrays. This way I won't have to call the function unnecessarily often in the core function.

3 results and conclusion

–

4 ending