Astro 9 Final

Benjamin Baum

July 2, 2020

1 Stellar Structure

I chose this problem because it was the one that had interested me the most; though I could tell that it would be challenging and it proved to be just that. I started by making a function called " f " that returned the right side of equation 3. Similarly, I created a function called "H" that returned the right side of equation 4.

$$\frac{dm}{dr} = 4\pi r^2 \rho(r) \qquad Eq.3$$

$$\frac{dp}{dr} = -\frac{G\epsilon(r)m(r)}{c^2 r^2}(1+\frac{p}{\epsilon})(1+4\frac{\pi r^3}{mc^2})\frac{1}{1-\frac{2Gm}{c^2 r}} \qquad Eq.4$$

After that I worked on my Runge-Kutta (RK4) funtion. This function was meant to solve both equations 3 and 4. I attempted to solve both by defining 2 sets of 4 k values like we did in class and on homework 3. I defined them as such:

$$k_1 = f(m_i, r) \qquad k_A = H(p_i, r_i)$$

$$k_1 = f(m_i + k_1\frac{dr}{2}, r_i + \frac{dr}{2}) \qquad k_B = H(p_i + k_A\frac{dr}{2}, r_i + \frac{dr}{2})$$

$$k_3 = f(m_i + k_2\frac{dr}{2}, r_i + \frac{dr}{2}) \qquad k_C = H(p_i + k_B\frac{dr}{2}, r + \frac{dr}{2})$$

$$k_4 = f(m_i + k_3 dr, r_i + dr) \qquad k_D = H(p_i + k_C dr, r + dr)$$

Once each value of k was solved for, my RK4 function used the following equations to solve for mass and pressure respectively.

$$m_{i+1} = m_i + \frac{dr}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

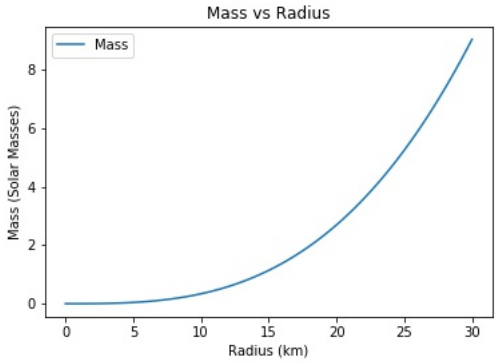
$$p_{i+1} = p_i + \frac{dr}{6}(k_A + 2k_B + 2k_C + k_D)$$

From here I included the boundary conditions that were given in the problem and encountered issues. Inside of my mass and pressure arrays I kept getting values of zeros and later nans. I came to the realization that my issues were with mass and energy densities. Using $\epsilon = \rho c^2$, I converted all energy densities into mass densities. I used the definition of mass density to find it by dividing mass over volume but I was still unable to get anything but zeros and nans in my array. I realized I was doing that wrong and saw that I needed an equation of state(EOS) that relate pressure to energy density. My initial intention was to test my function for a sun like star but I couldn't find an EOS for it. I then used the one given for the neutron star. I wasn't quite sure how to incorporate ϵ_0 into the EOS; other than making the first iteration's value of energy density being equal to ϵ_0. I tried the following two ways of solving for energy density but neither of them gave me data that looks accurate.

$$\epsilon_{i+1} = 2.4216p^{\frac{3}{5}} + 2.8633p$$

$$\epsilon_{i+1} = 2.4216(p\epsilon_0)^{\frac{3}{5}} + 2.8633p\epsilon_0$$

I was eventually able to get rid of the nans and the zeros within the arrays by using either of these versions of energy density and alternating a couple of initial conditions; the most impactful change was making $m[0]$ very small instead of equalling to zero, which was causing a lot of my nan issues. I was eventually able to plot a graph of mass versus radius that had a shape that I was expecting.



I was expecting an exponential graph like this with mass increasing as radius increases. Its difficult by looking at this code alone to determine if my code was functioning properly or not. It becomes more obvious when I graph pressure vs radius that my code is not working properly. I expected pressure to drop exponentially but instead I ended up getting a function that was almost constant. When I graphed energy density vs radius, I found that it had the exact same shape as my pressure function. This has lead me to think that my issues still lie with energy and mater densities. Unfortunately, I wasn't able to debug this in time and this was as far as I got on this problem.

2 Monte Carlo

This problem was quite enjoyable. Going in, I wasn't sure how to apply what we did in class with Monte Carlo on a probability problem like this. I tried looking for information online but I couldn't really find anything other than what I had already known; which was to use random numbers. So I essentially just used random numbers and what I know about probability to solve this problem.

I first set out to figure out how to determine if the center of the circle was within the triangle. When I first chose this problem, I thought this part would be easy. It was a lot more difficult than I expected and I put a lot of time into it. At one point I thought I had found a way using quadrants of the circle but I realized before attempting to code it that it wasn't going to work. Eventually, I found a video online at geeksforgeeks.org that explains how to mathematically find if a point is on a triangle.

First I had to find the area of the triangle using the following equation;

$$A = \frac{x_1(y_2-y_3)+x_2(y_3-y_1)+x_3(y_1-y_2)}{2} \qquad \text{Eq.I}$$

Then I had to find the area of 3 more triangles using the center point of the circle and two of the randomly chosen points. The 3 additional triangles should be composed of points C12,C23, and C13; with C representing the center point and the numbers representing each random point. If the center is within the triangle, then

$$A_{123} = A_{C12} + A_{C23} + A_{C13} \qquad \text{Eq.II}$$

My plan for this code was to do that process over many iterations and then by dividing the number of times the center was in the triangle by the step size we would get a probability.

I created an initializing function for the dots that I named "dot". This function first create a random number between 0 and 1 for the x coordinate of the dot. I then created another random number for the y value but I multiplied it with the equation of a circle,as shown bellow, to ensure that the dot is on a circle, of radius 1, centered at (0,0).

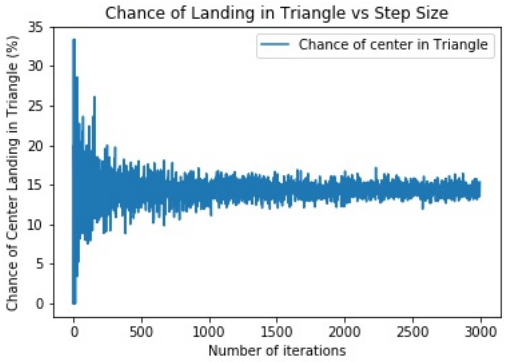
$$y = np.random.random() * (1 - x * *2) * *2$$

The x and y values could also be negative but so far the random numbers are only accounting for positive numbers. I therefore used another random number each to determine if the x and y values should be multiplied by a -1;

with a 50 percent chance of that happening for each value. The function then returns both the x and y values of the dot.

I then made a function called "inTri'. The job of this function was to receive the coordinates of the 3 dots and determine if the center of the circle was in the triangle by using Equations I and II, and the process previously explained. The function ends by returning True if the center was within the triangle or false if it was outside of it.

From here I iterated the functions many times and divided by the number of iterations. I did this many times and averaged the results. It came to be about a 14.34 percent chance that the center will end up in the triangle. The following is a graph to show how the probability converges with iterations.



References

https://www.geeksforgeeks.org/check-whether-a-given-point-lies-inside-a-triangle-or-not/