ASTP720 - Computational Methods

Homework 2

Liza Matrecito

17 September 2020

Problem 1

Write a numerical calculus library, one that can perform at least one derivative (e.g., the symmetric one) and at least three integration functions including the midpoint rule, trapezoidal rule, and Simpson's rule.

For coded solution, click this Github link.

Problem 2

Using your chosen parameters of c, v_{200}, and r_{200}, numerically determine the mass enclosed $M_{enc}(r)$ (and plot), the total mass of the dark matter halo M, and then also M(r), the amount of mass in a little shell around r $\pm$ Δr (i.e., what does the mass profile look like?), and dM(r)/dr. Compare this by repeating, holding c fixed and changing v_{200}. Again compare the first by repeating, holding v_{200} fixed and changing c. Feel free to use your previous tools from last time if you find that you need to (though I don't think you should); in the future you'll be able to use built-in functions for those.

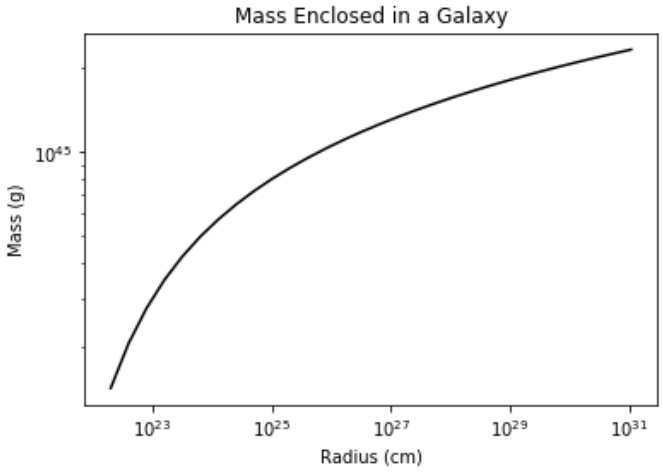

Figure 1: This plot shows the mass enclosed in a galaxy as a function of radius. It makes sense because as r increases, you are enclosing more mass.

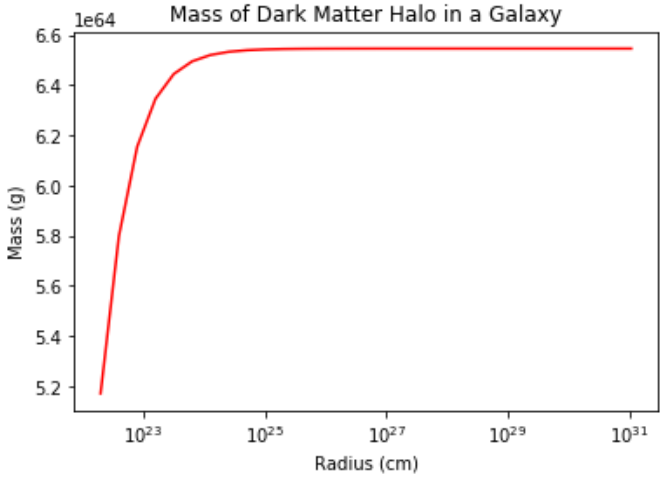

Figure 2: This plot shows the mass of the DM Halo in a galaxy and how it increases with distance. This makes sense because this is similar to the rotational velocity curve of a galaxy. Most of the dark matter is located in the halo which is why the velocity flattens instead of decreasing.

For coded solution, click this Github link.

Problem 3

In preparation for an actual astrophysical calculation moved into the next homework, write a matrix library that includes a Matrix class. There's lots of material online on how to write a class. Your class should be able to:

1. add two matrices together (you can overload __add__(), see below)
2. multiply two matrices together (again, can use __mult__)
3. transpose a matrix
4. invert a matrix
5. calculate the trace of a matrix
6. calculate the determinant of a matrix
7. return the LU decomposition of a matrix, (should return two Matrix objects representing L and U)

You may find it useful to also define some function that returns a single element i, j, swap rows, etc., whatever else you think might be useful but don't feel that you must. Internally, feel free to use numpy's np.array or even np.matrix if you so choose, but obviously don't just have your Matrix class just call the functions to ad, transpose, etc. As usual, please document, make sure to regularly commit to your repository, etc.

For coded solution, click this Github link.