ASTR 400B Homework 5

Due: 5 PM, Feb 20th 2025

In this assignment you will determine the mass distribution of each galaxy at SnapNumber 0 and use this to determine each galaxy's rotation curve.

- You will need the files "MW_000.txt", "M31_000.txt" and "M33_000.txt".
- Note, you will need to import *Read* from *ReadFile* and the class *CenterOfMass* from the program *CenterOfMass* you made in Homework 4. So make sure you have .py files of both codes in your working directory.
- Import relevant plotting modules - see solution to the Labs.

1 Class: Mass Profile

Create a class called *MassProfile*. You will follow the structure of the previous assignment to initialize your class. But you will not specify the particle type at this stage. Also, instead of the requiring the full filename as input, only supply the snapshot number and the galaxy name. i.e. Inputs:

- *galaxy*: A String with Galaxy Name, e.g. "MW", "M31" or "M33"
- *snap*: Snapshot number, e.g. 0, 1, etc

You will use these inputs to reconstruct the filename as follows:

```
# add a string of the filenumber to the value "000"
ilbl = '000' + str(Snap)
# remove all but the last 3 digits
ilbl = ilbl[-3:]
self.filename="%s_"%(galaxy) + ilbl + '.txt'
```

So if you want the file "MW_010.txt", you would input "MW" and 10.

Next, read in the data for the x,y,z positions and mass. Store x,y,z, with appropriate units. Don't assign the mass units just yet.

Store the name of the galaxy as a global property *self.gname*.

2 Mass Enclosed

Create a function called *MassEnclosed* that will compute the mass enclosed within a given radius of the COM position for a specified galaxy and a specified component of that galaxy. The function:

- should be able to calculate the enclosed masses for an array of radii and returns an array of masses (in units of $M_\odot$): so we can use it to calculate the mass profile conveniently.
- should have two input arguments (other than the `self` argument): the particle type and an array of radii (magnitude, not vector).

The usage will be something like the example below. Feel free to use the following numbers as a check; but no worries if you get something a bit different since the COM position may be determined differently.

```
In [2]: MW = MassProfile("MW", 0) # initialize the MassProfile class for MW

In [3]: r = np.arange(0.25, 30.5, 1.5); print(r) # create an array of radii as the input
[ 0.25  1.75  3.25  4.75  6.25  7.75  9.25 10.75 12.25 13.75 15.25 16.75
 18.25 19.75 21.25 22.75 24.25 25.75 27.25 28.75 30.25]

In [4]: MW.MassEnclosed(1, r) # get the enclosed halo masses at each element in 'r'
Out[4]:
<Quantity [0.00000000e+00, 1.26395200e+09, 4.54232750e+09, 9.20315050e+09, 1.60363910e+10,
           2.36991000e+10, 3.14803045e+10, 4.07624520e+10, 5.26909990e+10, 6.34740895e+10,
           7.57581230e+10, 8.82396490e+10, 1.02380112e+11, 1.15651608e+11, 1.29871068e+11,
           1.44169525e+11, 1.58941964e+11, 1.73477412e+11, 1.85563953e+11, 1.98756452e+11,
           2.12659924e+11] solMass>
```

Hints:

- You will need to first determine the COM position by creating a *CenterOfMass* object and calling *COM_P*. Use the disk particles for determining the center of mass position, irrespective of the particle type for which you want to compute the mass profile.
- You will need to loop over the Radius array (e.g., for i in range(size of Radius array)) to identify particles that are enclosed within the radius given at each array element (hint: use np.where, see solutions for ParticleProperties in HW2). You may need to assign units to the radius array.
- In each loop you need to store the sum of the masses of the particles that are within the radius. You should initialize this array outside the loop using *np.zeros*. So this array will store the total mass within each radius.
- when you return the array of mass enclosed, give it units of Msun.

With this function, we can obtain the enclosed mass as a function of radius – the so-called mass profile, which will help us understand the mass distribution inside each galaxy. Figure 1 plots an example mass profile from the usage example above. If you create an array of radii with a smaller step length, you'll get a smoother curve.

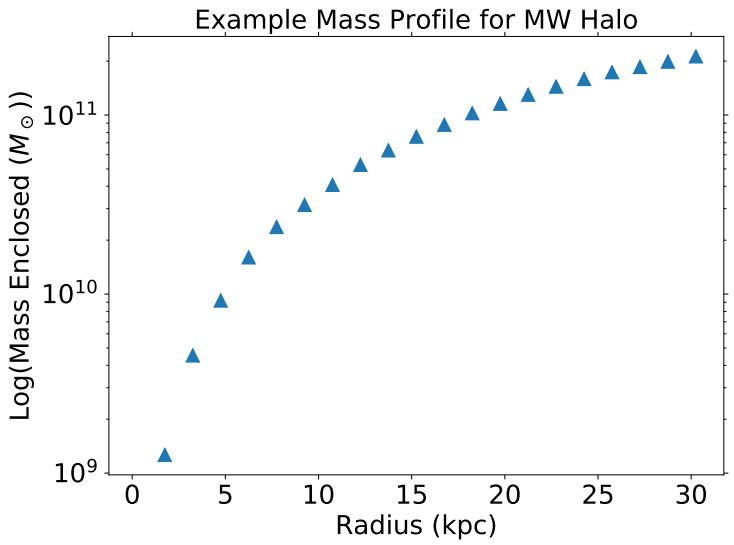

Figure 1: This picture presents an example of the mass profile of the Milky Way Halo particles, where *blue triangles* mark the enclosed mass as a function of radius.

3 Total Mass Enclosed

Create a function called *MassEnclosedTotal* that:

- Takes as input an array of radii (1D array)
- Calls MassEnclosed to compute the mass enclosed within the radius array for each particle type (bulge, disk and halo).
- Returns an array of masses (in units of $M_\odot$) representing the total enclosed mass (bulge+disk+halo) at each radius of the input array.
- NOTE: M33 does not have a bulge, so you need to add a caveat in the code to account for this possibility, e.g. using *self.gname*, which you stored in step 1.

4 Hernquist Mass Profile

Create a function called *HernquistMass* that will compute the mass enclosed within a given radius using the theoretical profile:

$$\rho(r) = \frac{Ma}{2\pi r}\frac{1}{(r+a)^3} \qquad M(r) = \frac{M_{halo}r^2}{(a+r)^2} \tag{1}$$

This function takes as input: the radius, the scale factor a, the halo mass Mhalo. It returns the halo mass in units of $M_\odot$.

5 Circular Velocity

Create a function called *CircularVelocity* with:

- inputs: the particle type and an array with radii.
- returns an array of circular speeds in units of km/s, rounded to two decimal places.

The circular speed is computed using the Mass enclosed at each radius, assuming spherical symmetry.

Note that astropy stores constants. You can use it for e.g. the Gravitational constant. *from astropy.constants import G*. But you will need to adjust the units. For our purposes: G = G.to(u.kpc*u.km**2/u.s**2/u.Msun). With these units you will get V in km/s.

6 Total CircularVelocity

Create a function called *CircularVelocityTotal* that:

- Takes as input an array of radii
- Returns an array of circular velocity (in units of km/s) representing the total Vcirc created by all the galaxy components (bulge+disk+halo) at each radius of the input array.

NOTE: the total circular velocity is NOT just the circular velocity of each individual galaxy component summed together.

7 Hernquist Circular Speed

Create a function called *HernquistVCirc* that computes the circular speed using the Hernquist mass profile. You can either call *HernquistMass* or write out the actual equation.

This function takes as input: the radius, the scale factor a, the halo mass Mhalo. It returns the circular speed in units of km/s, rounded to two decimal places.

8 Plot the Mass Profile for each Galaxy

1. Create a plot of the mass profile (mass enclosed as a function of radius) of each component of the MW to a radius of 30 kpc.
2. You will need to define an array of radii. Don't start the first element at 0 (say 0.1 instead).
3. Use different line types and/or colors to indicate different components. If you get stuck with the plotting look at the solutions for InClassLab1 on the class GitHub repo.
4. Y-axis should be in log (plt.semilogy).

5. Overplot the sum of all components (*MassEnclosedTotal*)

6. Determine the best fitting Hernquist profile:

   Use the total dark matter mass of each galaxy as M_{halo}.

   You will need to find the scale radius (a) where the Hernquist profile best matches the plotted mass profile as a function of radius. Do this by plotting the Hernquist profile (starting with a guess for a) on top of the simulated dark matter halo mass profile. Then keep changing a until the mass distributions reasonably agree. Alternatively, you could print out values for the simulated enclosed mass at 30 kpc and change a until you get a reasonable agreement with the Hernquist profile.

7. Use a different line-type or color for the best fit Hernquist profile and include text that states the best fit scale length.

8. Repeat for M33 and M31.

9 Plot the Rotation Curve for each Galaxy

1. Create a plot of the Rotation Curve of each galaxy - i.e. the circular speed of each galaxy component to a radius of 30 kpc.

2. You will need to define an array of radii. Don't start the first element at 0 (say 0.1 instead).

3. Use different line types and/or colors to indicate different components.

4. Overplot the total circular speed for all galaxy component (*CircularVelocityTotal*).

5. Overplot the best fit theoretical Hernquist circular speed, using the scale radius you determined from the mass enclosed plot.

6. Do the same for M31 and M33.

Note: not all components of a galaxy rotate, but the mass of each component contributes to the definition of the circular speed at a given radius. E.g. the bulge doesn't rotate, but it still has an associated rotation curve. The rotation curve tells you the speed needed to travel on a circular orbit at a given distance, it doesn't tell you the actual speed of the bulge particles themselves.