Orbital Velocity in the Milky Way

Jacob Mathew, Owen Urban, Zac Brutko

Motivations - Why we made this model

We made this model because astrophysicists made models to predict the movement of stars in the Milky Way galaxy. The model predicted that the stars that are further from the central supermassive black hole would rotate around it slower then the stars that are closer to the central black hole. However, when the astrophysicists observed the stars, they noticed that even the furthest stars rotate at about the same speed as the inner stars.

That observation lead to the theory of dark matter that increased the mass around those distant stars, which would explain why they rotate nearly as fast as the inner stars. We made our model to show how, when dark matter is taken into account, that the model more accurately shows the actual rotation curve of the stars of the Milky Way.

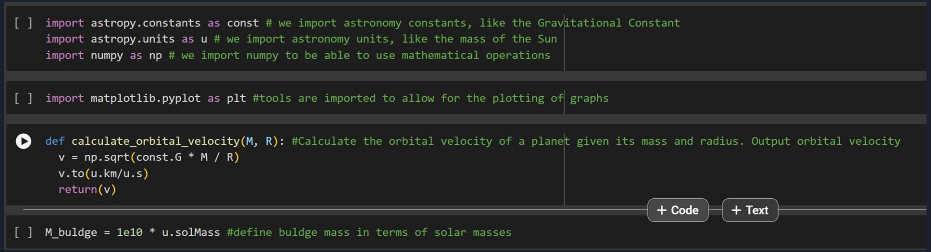
Motivations - Equations we used

For our model, our base equation was the ‘orbital velocity equation.’

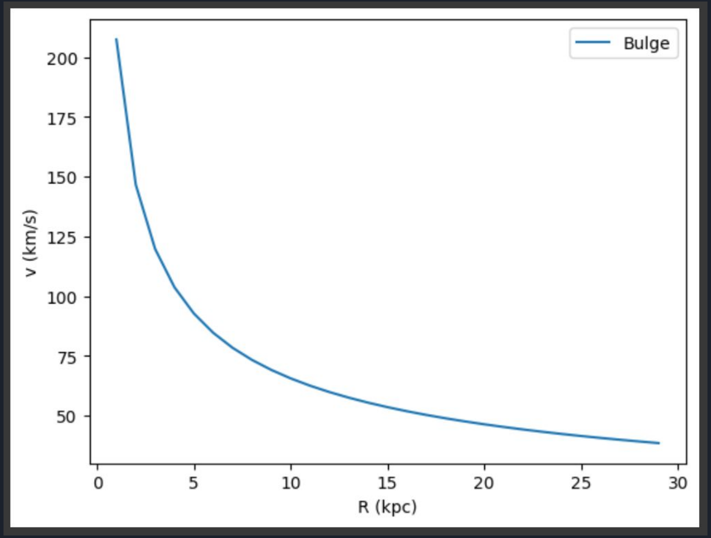
Orbital Velocity Equation: $V = \sqrt{\frac{GM}{R}}$

Methods - Begin Constructing the Model, Plot Bulge Component

- Useful packages are imported to help build our model
- A function is defined to solve for orbital velocity given a mass and radius
- The constant mass of the bulge is defined, and used along with an array of Radii to plot the rotation curve of the bulge

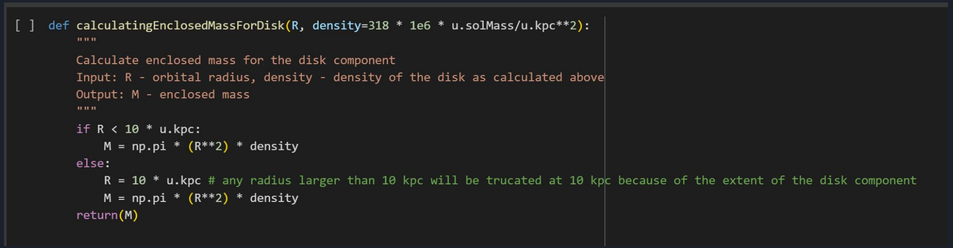


```
[ ] import astropy.constants as const # we import astronomy constants, like the Gravitational Constant
    import astropy.units as u # we import astronomy units, like the mass of the Sun
    import numpy as np # we import numpy to be able to use mathematical operations

[ ] import matplotlib.pyplot as plt #tools are imported to allow for the plotting of graphs

    def calculate_orbital_velocity(M, R): #Calculate the orbital velocity of a planet given its mass and radius. Output orbital velocity
      v = np.sqrt(const.G * M / R)
      v.to(u.km/u.s)
      return(v)

[ ] M_buldge = 1e10 * u.solMass #define buldge mass in terms of solar masses
```

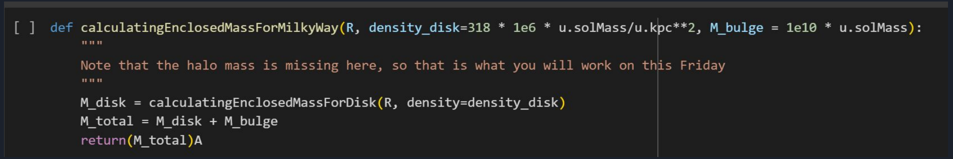


Methods - Adding the Disk Component

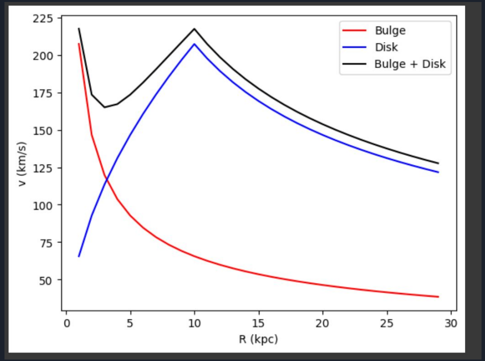
- Two new functions are defined to add the disk component - one for calculating enclosed mass and one for calculating the enclosed mass of the the disk and bulge
- Our first function defines the enclosed mass of the disk based off of the inputs of radius and density, and returns a mass in Solar Masses
- The second function defines the enclosed mass of the disk and bulge, based off of the inputs of radius, density of the disk, and the mass of the bulge
- These functions are used in conjunction with arrays of radii to plot the rotation curves of the bulge, the disk, and the two components combined



```
[ ]  def calculatingEnclosedMassForDisk(R, density=318 * 1e6 * u.solMass/u.kpc**2):
         """
         Calculate enclosed mass for the disk component
         Input: R - orbital radius, density - density of the disk as calculated above
         Output: M - enclosed mass
         """
         if R < 10 * u.kpc:
             M = np.pi * (R**2) * density
         else:
             R = 10 * u.kpc # any radius larger than 10 kpc will be trucated at 10 kpc because of the extent of the disk component
             M = np.pi * (R**2) * density
         return(M)
```

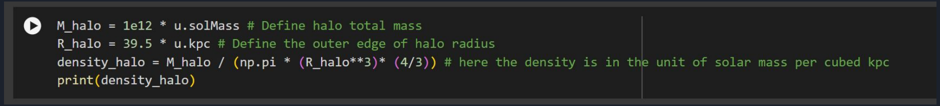


```
[ ]  def calculatingEnclosedMassForMilkyWay(R, density_disk=318 * 1e6 * u.solMass/u.kpc**2, M_bulge = 1e10 * u.solMass):
         """
         Note that the halo mass is missing here, so that is what you will work on this Friday
         """
         M_disk = calculatingEnclosedMassForDisk(R, density=density_disk)
         M_total = M_disk + M_bulge
         return(M_total)A
```

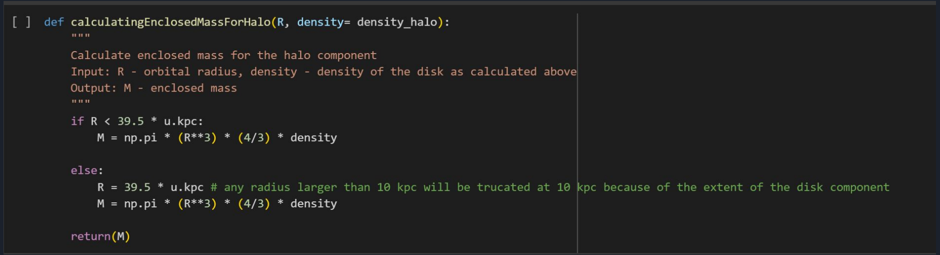


Methods - Adding the Halo Component

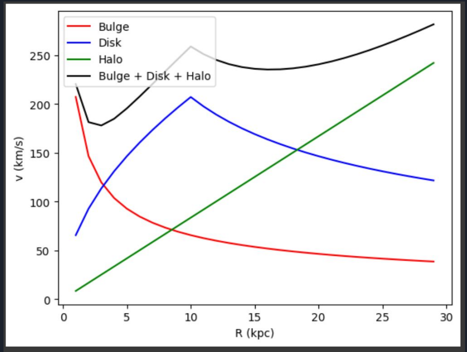
- The halo component is added very similarly to the disk, with slight changes to calculations
- The halo is assumed to be a sphere, meaning calculations for the density and enclosed mass change
- The coding process is the same as before, using these new calculated values to add the rotation curve of the halo and new total rotation curve to the model
- Data is then added and our model is adjusted slightly to form our final results



```
M_halo = 1e12 * u.solMass # Define halo total mass
R_halo = 39.5 * u.kpc # Define the outer edge of halo radius
density_halo = M_halo / (np.pi * (R_halo**3)* (4/3)) # here the density is in the unit of solar mass per cubed kpc
print(density_halo)
```



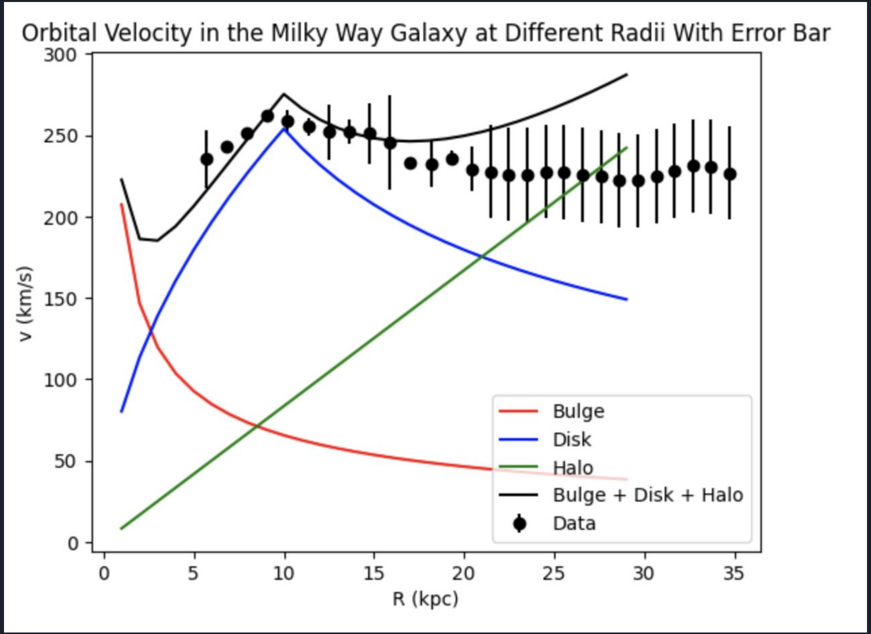
```
[ ] def calculatingEnclosedMassForHalo(R, density= density_halo):
        """
        Calculate enclosed mass for the halo component
        Input: R - orbital radius, density - density of the disk as calculated above
        Output: M - enclosed mass
        """
        if R < 39.5 * u.kpc:
            M = np.pi * (R**3) * (4/3) * density

        else:
            R = 39.5 * u.kpc # any radius larger than 10 kpc will be trucated at 10 kpc because of the extent of the disk component
            M = np.pi * (R**3) * (4/3) * density

        return(M)
```



Results:

- Normally, we would expect the total orbital velocity to decrease at farther distances, but according to our model the orbital velocities are increasing

- It seems that the component that contributes the most to the orbital velocities at these farther distances is the halo component



Conclusion:

- Based on the orbital velocity equation, this would have to mean that there is some extra mass in the galaxy that is affecting the orbital velocities at these farther radii
- scientists have started to call this mass "dark matter"
- From this we can conclude that there is extra mass that's not visible known as dark matter in the Milky Way galaxy and that most of it can be found in the galaxy's outermost part known as the halo.

AI Statement:

- For this project, we used Google’s Gemini in order to help us write the code to generate the graphs of the orbital velocities of the different components in the galaxy