Project 1 - Rotation Curve of the Milky Way

Justin M. Lewis, O'Brein Carr

The Ohio State University

ASTRON 1221: Astronomy Data Analysis

Dr. Ji Wang

September 23, 2024

Abstract

Using the orbital velocity equation (V = sqrt((G*M)/R)), we calculated the orbital velocity of the Milky Way Galaxy from its center out to a radius of 30 kilo-parsecs (kpc). By treating the central black hole as our reference point, we determined the orbital velocities for the galaxy's bulge, disk, and halo components based on their respective radii. The results were plotted on an "orbital velocity vs. radius" graph, enabling us to analyze the galaxy's rotation curve and compare it with the individual rotation curves of each component. This was further compared to the values already given for the M31Andromeda Galaxy.

Introduction

The motivation behind our team's project was to find the orbital velocities of each component of the Milky Way Galaxy and their respective rotation curves. The biggest motivations behind this project were gaining a deeper understanding of the orbital velocity equation, and how to apply it in our Python code.

Body (Methods)

The methods employed by our team involved taking our known equation for the orbital velocity and employing the use of python and Gemini AI to plot the data of each component's velocity vs. their radius on a graph. We imported a library of constants, mathematical operations, units, and plotting commands to carry out these sub-calculations and plot our results. Utilizing the data associated with the M31 galaxy, we also plotted its rotation curve to be compared to the Milky Way's.

In Figure 1.1 one can see the first steps of our methods employed, as we found the orbital velocity of the Milky Way galaxy bulge and plotted our findings using CoLab Python.

Also, in Figure 1.2 one can see the graph created by plotting the orbital velocity in CoLab.

```
G = ac.G
M_Bulge = 1e10 * u.solMass #Mass of Bulge
r = .1*u.kpc #Radius of bulge

def cal_Orbital_V(M, r):
    """
    Function to cal orbitabl velo
    M, Mass of the object
    R, Radius of the object
    """

    V = np.sqrt((ac.G*M)/r)
    return(V)

print(G.si)
print(M_Bulge)
print(r)

    Name   = Gravitational constant
    Value  = 6.6743e-11
    Uncertainty  = 1.5e-15
    Unit  = m3 / (kg s2)
    Reference = CODATA 2018
10000000000.0 solMass
0.1 kpc

r_arr = np.arange(1, 30) * u.kpc # Define a range of orbital radius in kilo parsec
V_arr = cal_Orbital_V(M_Bulge, r_arr) #Calc. Orbital Velo
print(V_arr.to(u.km/u.s)) #Orbitbal Velo in km/s

[207.3865297  146.64442148 119.73466875 103.69326485  92.7460756
  84.66519621  78.38474041  73.32221074  69.12884323  65.58137899
  62.52939142  59.86733437  57.51867436  55.42638148  53.54697172
  51.84663242  50.2986216   48.88147383  47.57773291  46.3730378
  45.25545097  44.21495669  43.24308072  42.33259811  41.47730594
```

Figure 1.1 (above)

```
plt.plot(r_arr.to(u.kpc), V_arr.to(u.km/u.s), label="Bulge") # note that here we can add label to the rotation curve caused by the Bulge
plt.xlabel('R (kpc)')
plt.ylabel('v (km/s)')
plt.title("Velocity vs. Radius")
plt.legend() # to show the legend of a figure
# The combination of labels and legend will come handy to show differnt curves in a plot as shown in future figures.
plt.show()
#
```

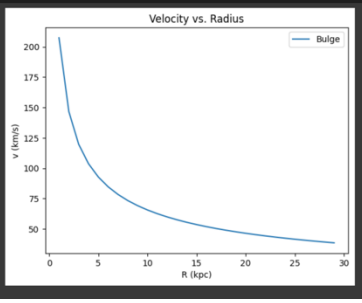

Figure 1.2 (above)

This method was employed for the Disk, and the Halo of the Milky Way Galaxy, but when working to solve the Halo, we used a NFW profile formula to solve for the enclosed mass of the Halo. To make better sense of the code, the definition “calculatingEnclosedMassForHalo”, calculates the enclosed mass of the halo for a given orbital radius. It takes R (the orbital radius)

as input and returns the enclosed mass M_Halo. The additional methods for the Halo can be seen in Figure 1.3 (seen below).

```
[ ]  import numpy as np
     import astropy.units as u
     import matplotlib.pyplot as plt

     # Define constants for the NFW profile
     D_Halo = 1e6 * u.solMass / u.kpc**3  # Characteristic density for halo
     R_Halo = 20 * u.kpc  # Scale radius for halo

     def calculatingEnclosedMassForHalo(R, D_Halo=D_Halo, R_Halo=R_Halo):
         """
         Calculate the enclosed mass for the halo component based on the NFW profile.
         Input: R - orbital radius
         Output: M_halo - enclosed mass for halo
         """
         # Calculate the enclosed mass using the NFW profile
         M_halo = 4 * np.pi * D_Halo * R_Halo**3 * (np.log(1 + R / R_Halo) - (R / (R_Halo + R)))
         return M_halo
```

Figure 1.3

Results

After finding the rotation curves for each component of the Milky Way Galaxy we created a graph that had the results of all the components as shown in figure 1.4.

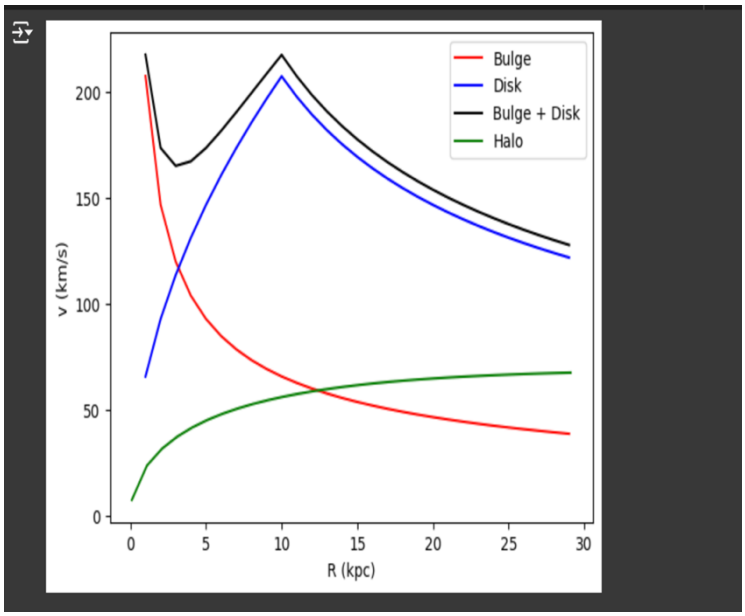

Figure 1.4

Conclusion

After finding our results, we added error bars of real data to compare to our results in an effort to make sense of our findings. The final results graph is shown in figure 1.5, and our team concluded that the results were sensible because the error bars were derived from the M31 Galaxy, and the results we had were derived from the Milky Way. Due to this, it would be sensible to assume that our results would not exactly match M31 but would share some similarities with the respective galaxy.

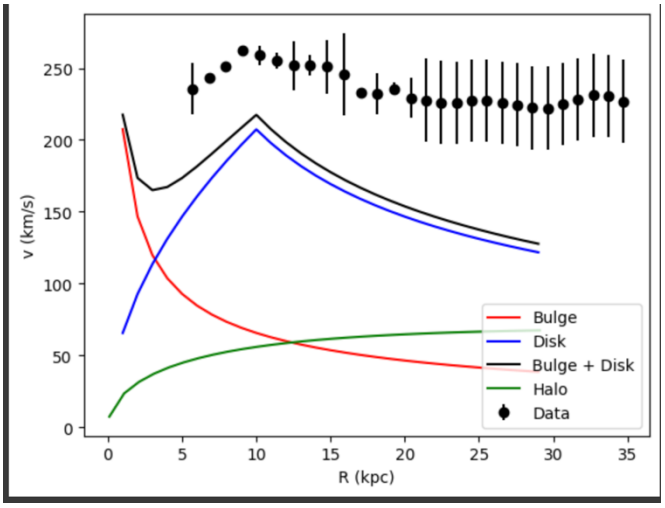

Figure 1.5

Contribution Statement

Justin provided the Figures (1.1-1.5) and wrote the Body, Results, Conclusion, and AI Statement's sections. O'Brein wrote the Abstract, Introduction, and edited the report for format and content.

AI Statement(Citations)

We used Colab's Gen AI as a supportive tool throughout the project. Rather than relying on AI to solve the errors in our code, we turned to Gemini to help identify errors when our code wouldn't run. After employing Gemini's help, we would then adjust the code that Gemini told us to fix, while adjusting to our parameters of the components of the Milky Way Galaxy. Gen AI's main purpose in this project was to aid in the creation of a physical representation of the Milky Way's rotation curve.