MILKYWAY GALAXY AND COMPONENTS

Project 1

Justin Lewis, Obrein Carr

MOTIVATIONS(WHY)

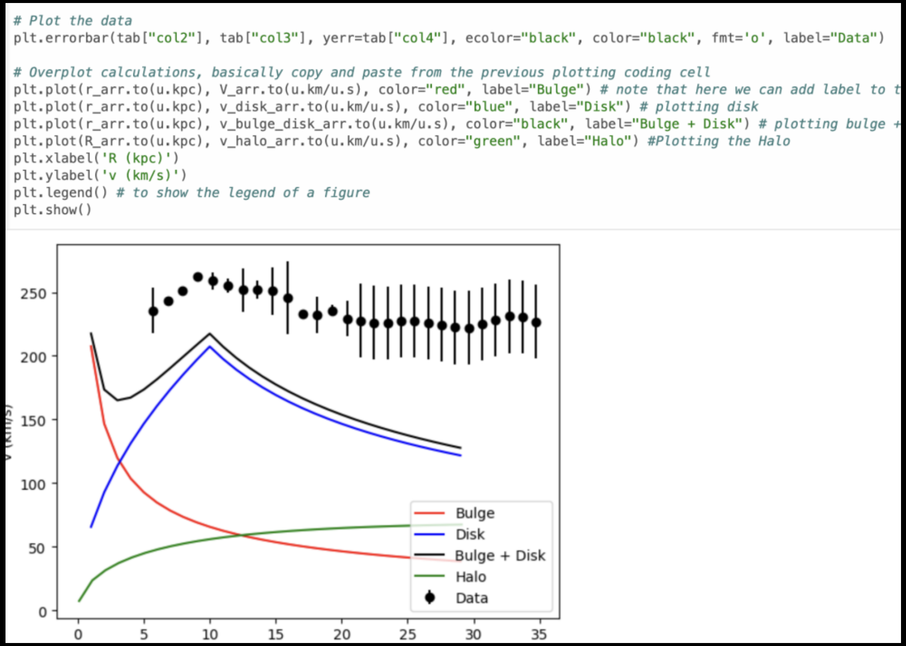
- Our group was tasked with finding the rotation curve of the MilkyWay Galaxy
- The task required us to learn about each component of the galaxy and how they play into the total rotation curve.
- The Halo component is invisible to humans
- How Dark matter affects the rotation curve
- Another factor was to learn the basics to python to then be able to make calculations that would be difficult by hand.

METHODS

STEPS

- The methods used consisted of finding the orbital velocity of each component.
- V = sqrt(GM/R)
- We also plotted each velocity vs its radius in a velocity-radius graph in order to show the rotation curve caused by each component.
- Finally, we added real data to the graphs to compare our calculated data to actual data.

FINDINGS



RESULTS

```python
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
  40.67184468  39.91155625  39.1923702   38.51071177] km / s
```

```python
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 di
        M = np.pi * (R**2) * density
    return(M)
```

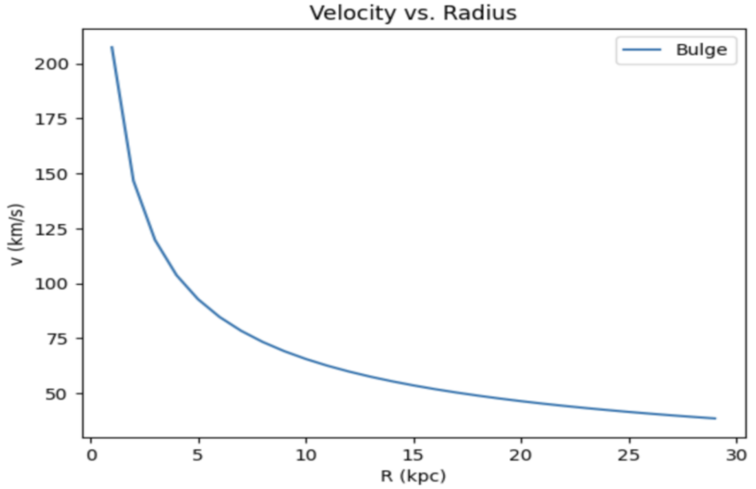
```python
print(calculatingEnclosedMassForDisk(1 * u.kpc).to(1e6 * u.solMass), "at 1 kpc") # to convert to million solar ma
print(calculatingEnclosedMassForDisk(5 * u.kpc).to(1e6 * u.solMass), "at 5 kpc")
```

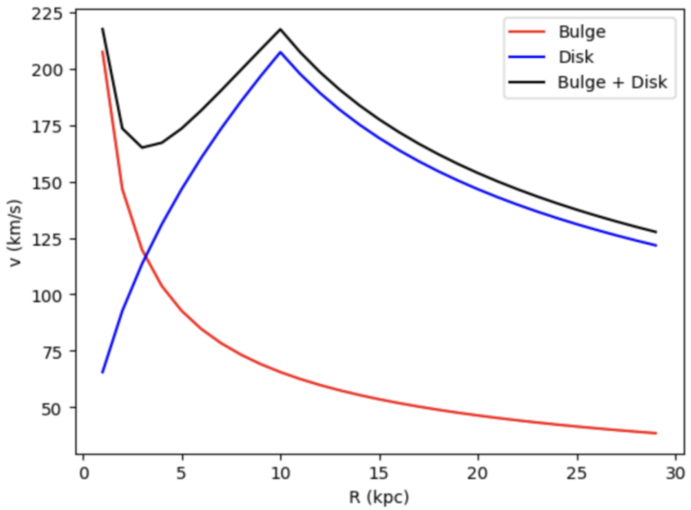
```
999.0264638415543 1e+06 solMass at 1 kpc
24975.66159603886 1e+06 solMass at 5 kpc
```

```python
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
```

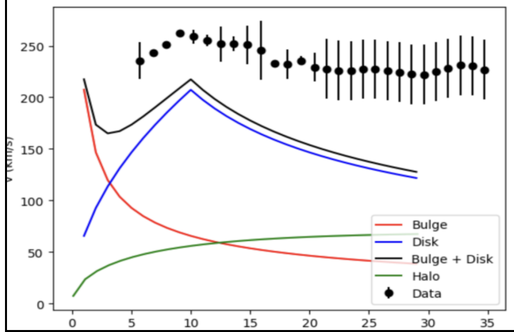
```python
print(calculatingEnclosedMassForHalo(1 * u.kpc).to(0.01 * u.solMass), "at 1 kpc") # to convert to million solar m
print(calculatingEnclosedMassForHalo(5 * u.kpc).to(0.01 * u.solMass), "at 5 kpc")
```

```
11773347683.79249 0.01 solMass at 1 kpc
232664354517.43924 0.01 solMass at 5 kpc
```






```python
# Plot the data
plt.errorbar(tab["col2"], tab["col3"], yerr=tab["col4"], ecolor="black", color="black", fmt='o', label="Data")

# Overplot calculations, basically copy and paste from the previous plotting coding cell
plt.plot(r_arr.to(u.kpc), V_arr.to(u.km/u.s), color="red", label="Bulge") # note that here we can add label to t
plt.plot(r_arr.to(u.kpc), v_disk_arr.to(u.km/u.s), color="blue", label="Disk") # plotting disk
plt.plot(r_arr.to(u.kpc), v_bulge_disk_arr.to(u.km/u.s), color="black", label="Bulge + Disk") # plotting bulge +
plt.plot(R_arr.to(u.kpc), v_halo_arr.to(u.km/u.s), color="green", label="Halo") #Plotting the Halo
plt.xlabel('R (kpc)')
plt.ylabel('v (km/s)')
plt.legend() # to show the legend of a figure
plt.show()
```



CONCLUSION

In conclusion, we can see that our orbital velocity found of the milky way is off from the actual data measurements of the rotation curve.

We believe that Dark matter is cause to the discrepency in the theoretical data vs the tested data.

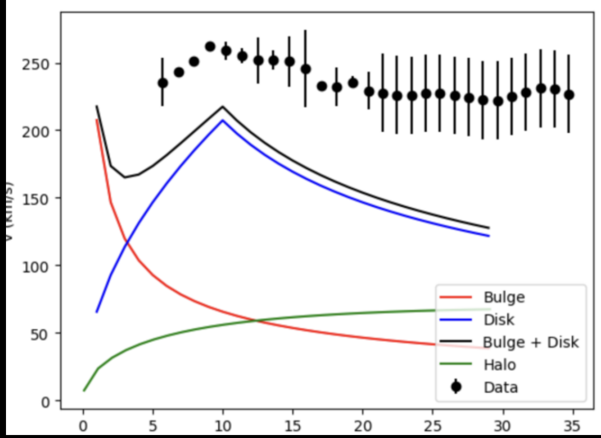
```python
!pip install astropy

import numpy as np
import astropy.units as u
import astropy.constants as ac

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

  V_Milky = np.sqrt((ac.G*M_MilkyWay)/R_MilkyWay)
  return(V_Milky)
```

```
Requirement already satisfied: astropy in /usr/local/lib/python3.10/dist-packages (6.1.3)
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.10/dist-packages (from astropy) (1.26.4)
Requirement already satisfied: pyerfa>=2.0.1.1 in /usr/local/lib/python3.10/dist-packages (from astropy) (2.0.1.4)
Requirement already satisfied: astropy-iers-data>=0.2024.7.29.0.32.7 in /usr/local/lib/python3.10/dist-packages (f
rom astropy) (0.2024.9.2.0.33.23)
Requirement already satisfied: PyYAML>=3.13 in /usr/local/lib/python3.10/dist-packages (from astropy) (6.0.2)
Requirement already satisfied: packaging>=19.0 in /usr/local/lib/python3.10/dist-packages (from astropy) (24.1)
```

```python
V_Milky = cal_Orbital_V(M_MilkyWay, R_MilkyWay) #Calls function to assign V_Milky
print(V_Milky.to(u.km/u.s))
```

```
[32.99963236 33.2302994  33.78801897 34.60087504 35.60710667 36.75572355
 38.00585009 39.32541378 40.68963536 42.07958706 43.48093936 44.88292924
 46.27753788 47.6588482  49.02254878 50.36555336 51.68570974 52.98157753
 54.2522581  55.4972645  56.71642164 57.90978961 59.0776046  60.22023337
 61.33813802 62.43184881 63.50194311 64.54902915 65.57373347 66.57669128] km / s
```

```python
# Plot the data
plt.errorbar(tab["col2"], tab["col3"], yerr=tab["col4"], ecolor="black", color="black", fmt='o', label="Data")

# Overplot calculations, basically copy and paste from the previous plotting coding cell
plt.plot(r_arr.to(u.kpc), V_arr.to(u.km/u.s), color="red", label="Bulge") # note that here we can add label to t
plt.plot(r_arr.to(u.kpc), v_disk_arr.to(u.km/u.s), color="blue", label="Disk") # plotting disk
plt.plot(r_arr.to(u.kpc), v_bulge_disk_arr.to(u.km/u.s), color="black", label="Bulge + Disk") # plotting bulge +
plt.plot(R_arr.to(u.kpc), v_halo_arr.to(u.km/u.s), color="green", label="Halo") #Plotting the Halo
plt.xlabel('R (kpc)')
plt.ylabel('v (km/s)')
plt.legend() # to show the legend of a figure
plt.show()
```



ANY QUESTIONS?