4-19-2023

Analysis of Numerical Simulation of Galaxy Clusters

Caitlyn Ruth Eply

Analysis of Numerical Simulation of Galaxy Clusters

by

Caitlyn Ruth Epley

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

April 23, 2023

Honors Capstone Director: Dr. Max Bonamente

Professor of Physics and Astronomy

Caitlyn Epley 4/19/2023

Student (signature) Date

Massimiliano Bonamente (digitally signed) 4/20/22

Director (signature) Date

JAM Digitally signed by James A Miller
Date: 2023.04.20 11:00:12 -05'00'

Department Chair (signature) Date

William Wilkerson Digitally signed by William Wilkerson
Date: 2023.04.21 11:08:38 -05'00'

Honors College Dean (signature) Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Caitlyn Epley

Student Name (printed)

Student Signature

4/19/2023

Date

Table of Contents

ABSTRACT

This report covers an exploration of modeling spherical galaxy cluster simulations using Python and the Matplotlib Python library. The area of interest is a spherical cluster of particles with an intersecting cylindrical shell, which represents a detector, such as a telescope. The goal of this project is to determine and verify a method for determining the volume of intersection between spherical shells in the cluster and the cylindrical shell of the detector view. This volume can then be used to analyze the quantities, such as number density of hydrogen atoms, from spherical galaxy cluster simulation data from the IllustrisTNG simulation.

INTRODUCTION

Importance of Galaxy Simulations.

Galaxy cluster simulations, such as the IllustrisTNG simulations are important for cosmological research, as they allow researchers to study the evolution of galaxies and the overall universe. For certain applications, simulations can be more useful than observational data, as simulations can cover time spans that are not realistic for real-time experiments (TNG Project Description). For this project, a dataset produced by the IllustrisTNG simulation is used, which contains radius values for the spherical cluster, as well as density values for the types of particles that inhabit the cluster, such as hydrogen, warm-hot intergalactic medium (WHIM), etc.

Goal of this Project

For this report, the focus is on a sphere consisting of concentric shells with an intersecting cylindrical shell, which represents a detector. The goal is to find the volume of intersection between each spherical shell and the cylindrical shell that represents the detector view. Data produced by the IllustrisTNG simulation can then be projected onto this volume. This method allows for analysis of the simulation data based on where it is in the sphere as well as where it lies in view of the detector.

METHODS

Intersection of a Sphere with a Concentric Cylinder

The first area of interest is the intersection between a solid sphere and a concentric solid cylinder, as shown in Figure 1 and Figure 2. This area of interest is important, as the formula

derived below is necessary for determining the volume of intersection between a cylindrical shell and a spherical shell. The sphere represents a three-dimensional distribution of gas in space, and the concentric cylinder represents the view from a detector, such as a telescope. The goal of this simulation is to find the volume of intersection between the sphere and the view from the detector based on the number of points that are present in the intersection compared to the total number of points present in the region. This volume can then be compared to the derived formula for the volume of intersection.

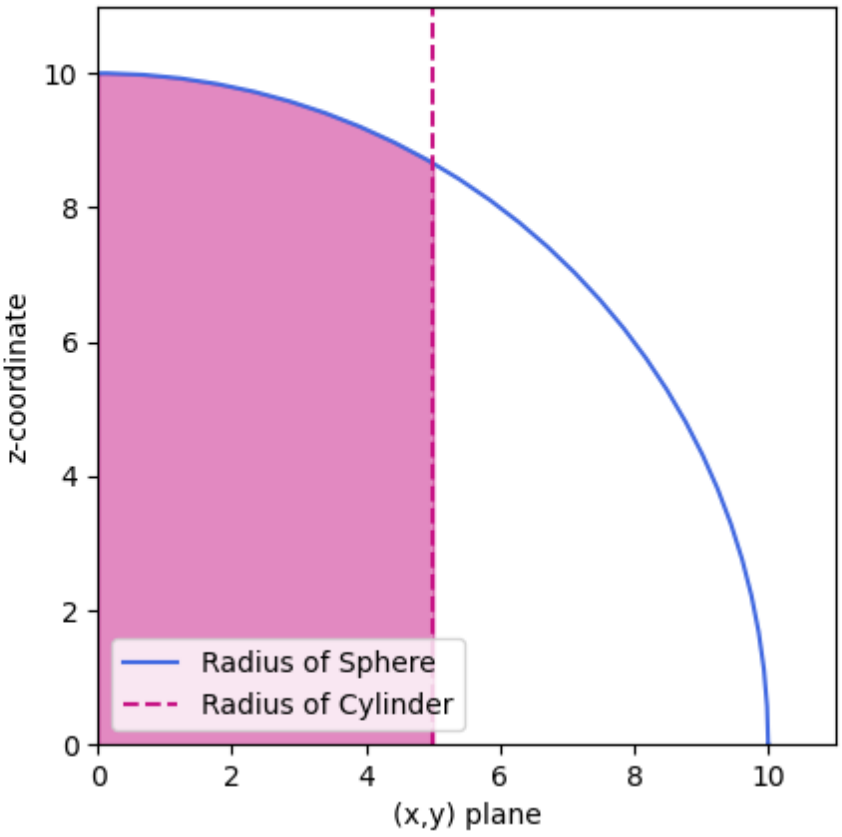

***Figure 1**: Two-dimensional representation of the intersection between a solid sphere and a solid cylinder.*

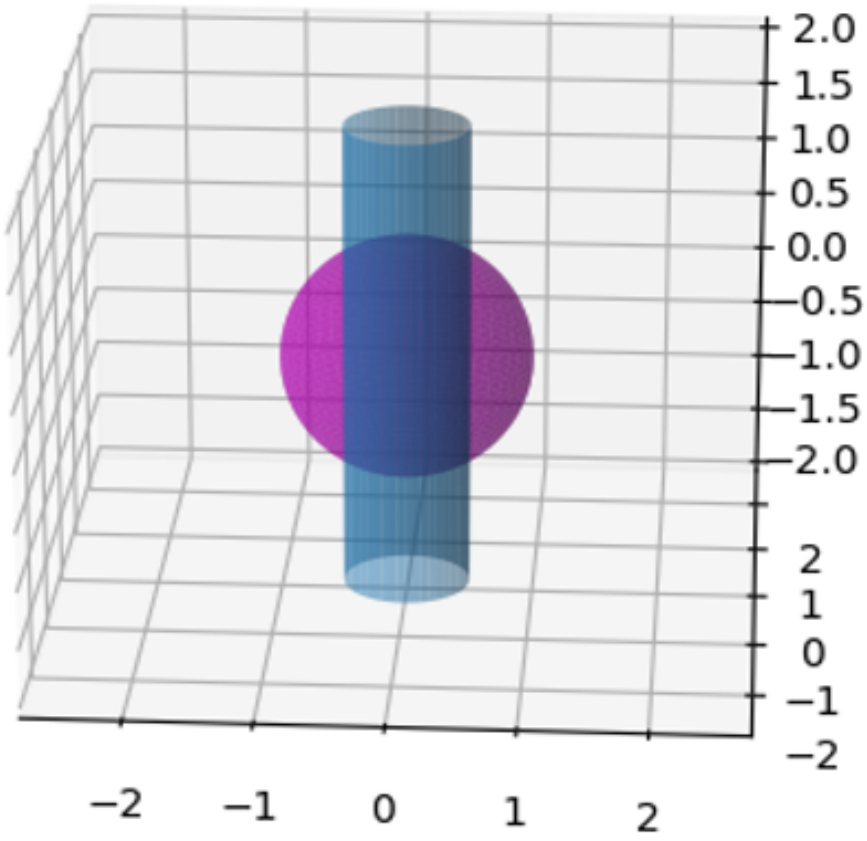

***Figure 2**: Three-dimensional representation of the intersection between a solid sphere and a solid cylinder.*

Find the volume of intersection of a sphere with a concentric cylinder:

The area of interest is the volume of intersection of a sphere and a cylinder where the radius of the sphere is greater than that of the cylinder (in the case where the cylinder's radius is greater than or equal to the sphere's, the total volume would be the entire sphere, as the sphere would be fully encased in the cylinder). The volume sought out should appear to be a cylinder with "spherical caps" instead of flat edges.

To represent this volume, the bounds of the variables needed to determine the volume of intersection are:

$$\rho\epsilon[0,\rho_{max}],\theta\epsilon[0,2\pi],z\epsilon[0,z(\rho,R)]$$

Where ρ is the radius of the cylinder, R is the radius of the sphere, and $z(\rho, R)$ is the cap created by the intersection of the shapes. The volume integral is therefore:

$$V(\rho_{max}, R) = 2\rho d\theta d\rho dz = 2\int_0^{2\pi} d\theta \int_0^{\rho_{max}} \rho d\rho \int_0^{\sqrt{R^2-\rho^2}} dz \qquad \textbf{(1)}$$

Solving equation (1) results in the formula for the volume of the intersection between the radius and the sphere when $\rho_{max} < R$:

$$V(\rho_{max}, R) = \frac{4\pi}{3}\left(R^3 - (R^2 - \rho^2)^{\frac{3}{2}}\right) \qquad \textbf{(2)}$$

Verify this formula using Python:

The script to perform this simulation is found in the appendices at the end of this paper, under "Intersection of Whole Sphere/Whole Cylinder and Intersection of Spherical Shell/Cylindrical Shell Python Script". This script produces the results from this section and the next section (intersection of spherical shell with cylindrical shell).

The simulation data in this case consists of data points, each with x, y, and z-coordinate values. These points are randomly generated and are all between -1 and 1 unit. Not every point will be included as the area of interest here only includes the points that lie on/in the intersection of the sphere and the cylinder. The data points that are in the intersection are filtered out so that the volume of interest can be calculated.

To test that the prediction would produce an accurate result, the code was supplied with 10,000 random Cartesian points to represent the overall volume. The total volume of the region is a cube with a volume of $8units^3$. In this case, the radius of the sphere, R, is just 1 unit, and the tested value for the radius of the cylinder is 0.5 units.

Each random coordinate, consisting of an x, y, and z value, is tested to see if it lies inside, or on, the sphere and cylinder. If the point is included in both the sphere and cylinder, then the x, y, and z values are added to their respective lists (in the code, these are xintWhole, yintWhole, and zintWhole, which are arrays). These final lists represent all the points that are included in the intersection of a whole sphere and a whole cylinder. Also, for each point in space that is included in the intersection, a counter variable (called pointCountWhole) is iterated. This

represents the total number of points included in the intersection, which is needed to calculate the volume of the simulation.

The filtered data can be plotted to show that all the points included are in the area of interest. Figure 3 shows a three-dimensional scatter plot of the points that are included in the intersection of a sphere and cylinder.

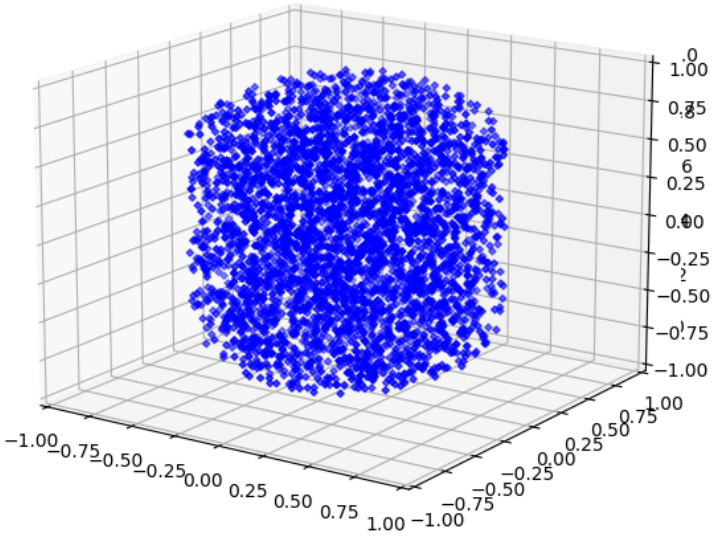

***Figure 3**: Three-dimensional scatter plot of the random values that are included in the intersection of an entire sphere with an entire cylinder.*

Once all the data has been filtered, the volume of this intersection can be found using:

$$vol(\rho_{max}, R) = R^3 * numberofpoints \in \frac{the\cap}{totalpoints} \qquad \textbf{(3)}$$

Where the volume is a ratio of the points in the intersection to the total points, multiplied by the total volume.

Once the prediction for the volume of the intersection is made, it must be compared to the value produced by the derived formula for the volume of intersection.

Error analysis for intersection of sphere and cylinder:

To verify that the volume of the intersection calculated by finding the ratio of points is a precise approximation, the predicted volume must be compared to the actual volume calculated using the derived formula for the volume. To do this, a general error calculation formula is used:

$$\%error = \frac{(predicted - actual)}{actual} * 100 \qquad \textbf{(4)}$$

Since the program produces a volume based on a ratio of points, it is important to have enough data points to produce an accurate result. The randomly generated values, as well as the simulation values, are not uniformly distributed across the total volume, so a small dataset may not produce accurate results, depending on how the data points are distributed.

To determine if a parameter set produces precise volume approximations, it is necessary to test the set with multiple random datasets. Figure 4 represents the error for a singular set of parameters. The parameters for this example are a sphere with radius of 1 unit, a cylinder with radius of 0.5 units, 10,000 random data points per occurrence, and 1,000 trials. Each trial has a unique set of random numbers. If the parameters allow for an accurate approximation, then the histogram of all the trials should fit to a Gaussian curve, as shown in the figure. This shows that the majority of the trials have low error.

It is also important to note that a greater cylindrical radius compared to the spherical radius will result in higher accuracy of the model. This is because the ratio of the area of intersection to the entire region is greater, and therefore the probability that a randomly generated point will fall into this intersection is also greater.

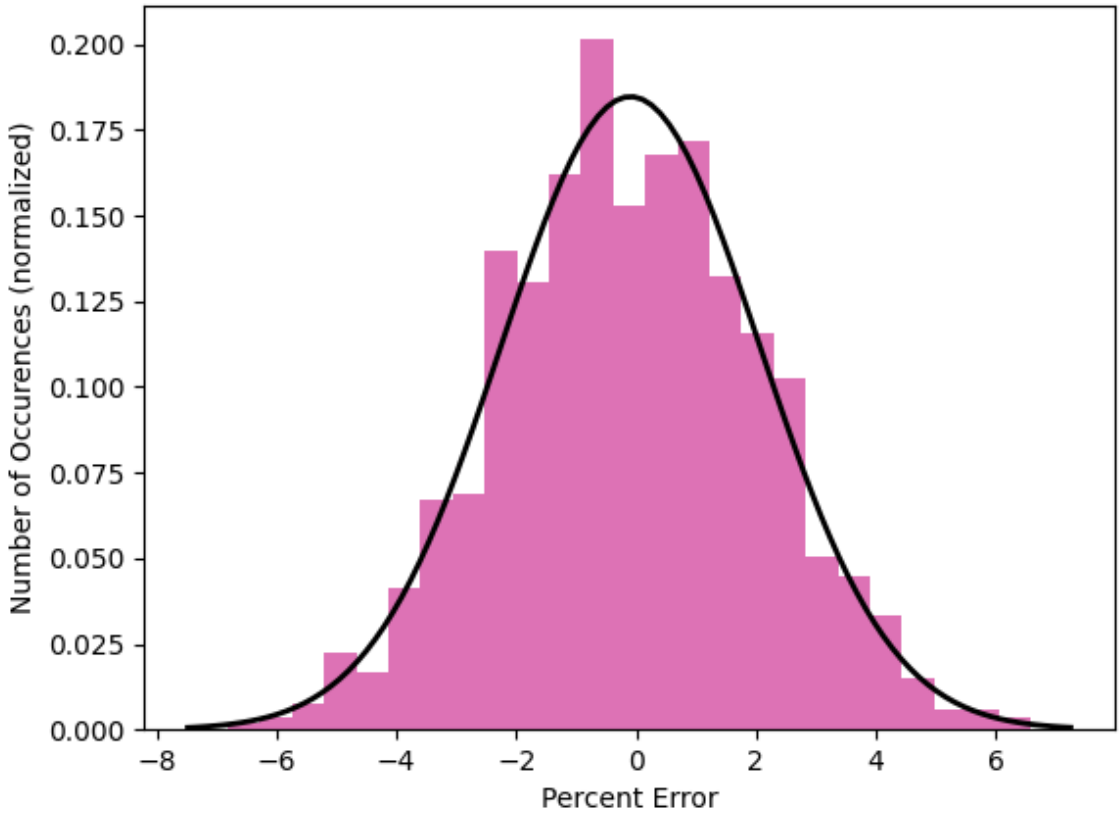

***Figure 4**: Example of the distribution of error for the intersection of an entire sphere with an entire cylinder.*

Developing a Method for Working with the Intersection of a Spherical Shell with a Concentric Cylindrical Shell

Now, building off the work done in the previous section, it is necessary to derive the formula for calculating the volume between a spherical shell and a concentric cylindrical shell, as shown in Figure 4. This representation is useful because it allows for a more complete analysis of the given data, based on the location of the points in the sphere and where they fall in the view of the detector, rather than just the overall volume of intersection. This volume of intersection is also the volume that the simulation data will be applied to later.

As with the intersection of a whole sphere and cylinder, the area of interest here is where the maximum radius of the sphere is greater than the maximum radius of the cylinder. The volume

of this intersection should also be smaller than that of the entire sphere and cylinder, as it is the intersection between layers of each, rather than the total overall volume of intersection. Figures 5 and 6 show the two-dimensional and three-dimensional view of this intersection. This volume of intersection appears as two “rings” of points near the top and bottom of the sphere.

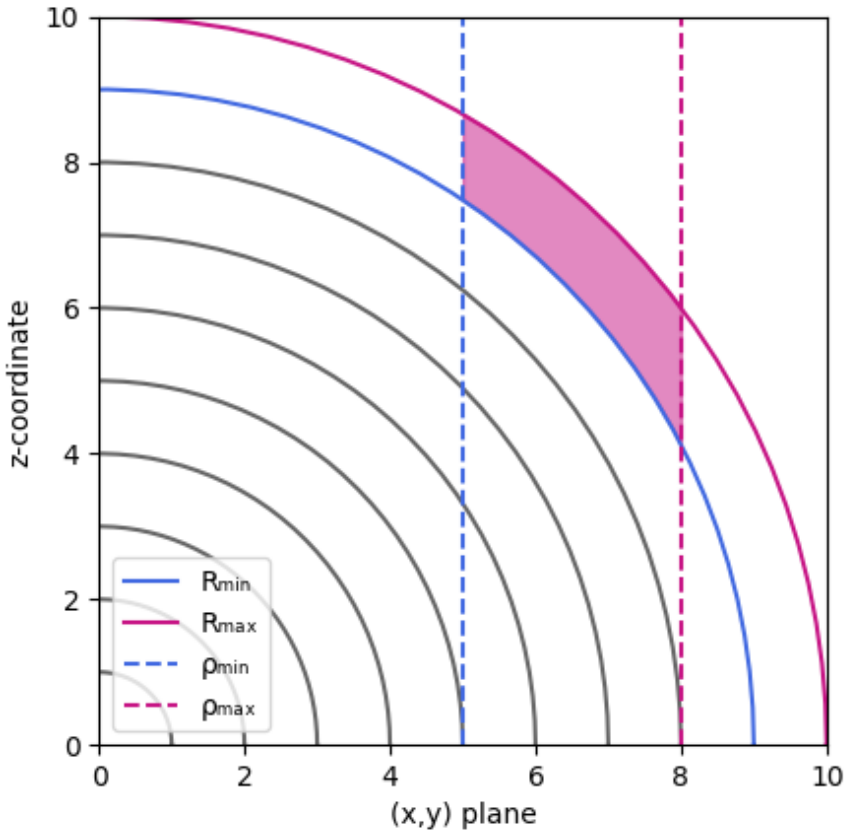

***Figure 5**: Two-dimensional representation of the intersection between a spherical shell with minimum radius R_{min} and maximum radius R_{max} and a cylindrical shell with a minimum radius ρ_{min} and a maximum radiusρ_{max}.*

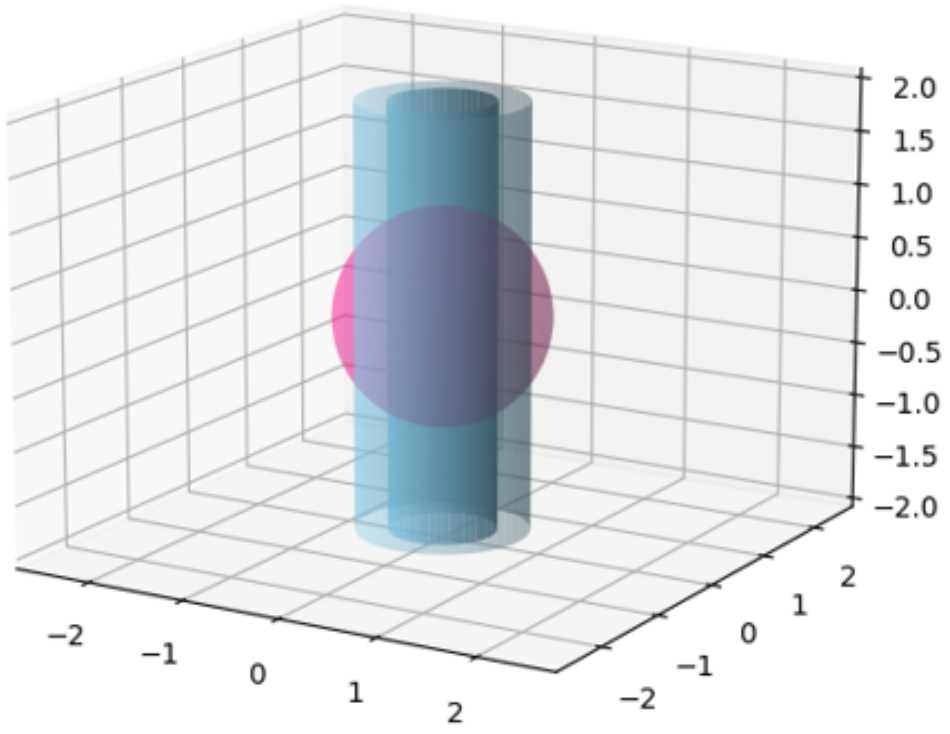

***Figure 6**: Three-dimensional representation of the intersection between a spherical shell with minimum radius R_{min} and maximum radius R_{max} and a cylindrical shell with a minimum radius ρ_{min} and a maximum radiusρ_{max}.*

To find the volume of the intersection of a spherical shell with a concentric cylindrical shell:

The area of interest is the volume of intersection of a spherical shell and a cylindrical shell where the maximum radius of the sphere is greater than the maximum radius of the cylinder.

To represent this volume, the bounds of the variables needed to determine the volume of intersection are:

$$\rho\epsilon[\rho_{min}, \rho_{max}], \theta\epsilon[0,2\pi], z\epsilon[z(\rho_{min}, R_{min}), z(\rho_{max}, R_{max})]$$

Where, as in the previous example, ρ is the radius of the cylinder, R is the radius of the sphere, and $z(\rho, R)$ is the cap created by the intersection of the shapes. The volume integral is therefore:

$$V(\rho_{max}, R) = 2\rho d\theta d\rho dz = 2\int_0^{2\pi} d\theta \int_{\rho_{min}}^{\rho_{max}} \rho d\rho \int_{\sqrt{R_{min}^2-\rho_{min}^2}}^{\sqrt{R_{max}^2-\rho_{max}^2}} dz \qquad \textbf{(5)}$$

To simplify (5), the intersection between the spherical and cylindrical shells can be viewed as the difference in volume between the intersection of a spherical shell of radii R_{max}, R_{min} and a whole cylinder of radius ρ_{max} and the intersection of a spherical shell of radii R_{max}, R_{min} and a whole cylinder of radius ρ_{min}.

The volume for the intersection between a cylindrical shell and a whole cylinder is:

$$V(\rho, R_{max,} R_{min}) = 4\pi \int_0^{\rho} \rho d\rho \int_{\sqrt{R_{min}^2-\rho^2}}^{\sqrt{R_{max}^2-\rho^2}} dz \qquad \textbf{(6)}$$

Equation (6) is equivalent to the difference in volume between the intersection of a whole sphere of radius R_{max} and a whole cylinder of radius ρ_{max} and the intersection of a whole sphere of radius R_{min} and a whole cylinder of radius ρ_{min}. This builds off equation (2) and appears as:

$$V(\rho_{max}, \rho_{min}, R) = \frac{4\pi}{3}\left(R^3 - (R^2 - \rho_{max}^2)^{\frac{3}{2}}\right) - \frac{4\pi}{3}\left(R^3 - \left(R^2 - \rho_{min}^2\right)^{\frac{3}{2}}\right) \qquad \textbf{(7)}$$

From equation (7), the volume of intersection between the spherical shell and cylindrical shell can be derived as:

$$V(\rho_{max}, \rho_{min}, R_{max}, R_{min}) = \frac{4\pi}{3}\left({R_{max}}^3 - \left({R_{max}}^2 - \rho_{max}^2\right)^{\frac{3}{2}}\right) - \frac{4\pi}{3}\left(R_{max}^3 - \left({R_{max}}^2 - \rho_{min}^2\right)^{\frac{3}{2}}\right) -$$
$$\frac{4\pi}{3}\left({R_{min}}^3 - \left({R_{min}}^2 - \rho_{max}^2\right)^{\frac{3}{2}}\right) - \frac{4\pi}{3}\left(R_{min}^3 - \left({R_{min}}^2 - \rho_{min}^2\right)^{\frac{3}{2}}\right) \qquad \textbf{(8)}$$

Verify this formula using Python:

The script to perform this simulation is found in the appendices at the end of this paper, under "Intersection of Whole Sphere/Whole Cylinder and Intersection of Spherical Shell/Cylindrical Shell Python Script". This script produces the results from this section and the previous section (intersection of whole sphere with whole cylinder).

As with the previous example, the simulation data in this case consists of data points, each with x, y, and z-coordinate values. These points are randomly generated and are all between -1 and 1 unit. Not every point will be included as the area of interest here only includes the points that lie on/in the intersection of the sphere and the cylinder. The data points that are in the intersection are filtered out so that the volume of interest can be calculated.

To test that the prediction would produce an accurate result, the code was supplied with 10,000 random Cartesian points to represent the overall volume. The total volume of the region is a cube with a volume of $8units^3$. In this case, the maximum and minimum radii of the spherical shell are 1 and 0.8 units respectively. The maximum and minimum radius of the cylindrical shell are 0.8 and 0.5 units, respectively.

Each random coordinate, consisting of an x, y, and z value, is tested to see if it lies inside, or on, the spherical shell and cylindrical shell. If the point is included in both shells, then the x, y, and z values are added to their respective lists (in the code, these are xintShells, yintShells, and zintShells, which are arrays). These final lists represent all the points that are included in the intersection of a whole sphere and a whole cylinder. Also, for each point in space that is included in the intersection, a counter variable (called pointCountShells) is iterated. This represents the total number of points included in the intersection, which is needed to calculate the volume of the simulation.

The filtered data can be plotted to show that all the points included are in the area of interest. Figure 7 shows a three-dimensional scatter plot of the points that are included in the intersection of a sphere and cylinder.

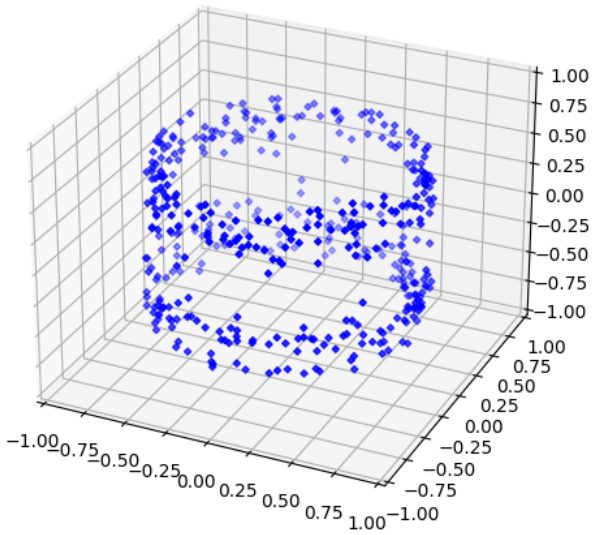

***Figure 7**: Three-dimensional scatter plot of the random values that are included in the intersection of a spherical shell with a cylindrical shell.*

Once all the data has been filtered, the volume of this intersection can be found using equation (3), where the volume is a ratio of the points in the intersection to the total points, multiplied by the total volume. The script to perform this simulation is found in the appendices at the end of this paper.

Once the prediction for the volume of the intersection is made, it must be compared to the value produced by the derived formula for the volume of intersection.

Error calculation:

To verify that the volume of the intersection calculated by finding the ratio of points is a precise approximation, the predicted volume must be compared to the actual volume calculated using the derived formula for the volume. To do this, the general error calculation formula from equation (4) is used.

As with the previous example, the produces a volume based on a ratio of points, so it is important to have enough data points to produce an accurate result.

To determine if a parameter set produces precise volume approximations, it is necessary to test the set with multiple random datasets. Figure 8 represents the error for a singular set of parameters. The parameters for this example are a spherical shell with radii of 1 unit and 0.8 units, a cylindrical shell with radii of 0.8 and 0.5 units, 10,000 random data points per occurrence, and 1,000 trials. Each trial has a unique set of random numbers. If the parameters allow for an accurate approximation, then the histogram of all the trials should fit to a Gaussian curve, as shown in the figure. This shows that the majority of the trials have low error.

Density of values from the example dataset plotted against the radius.

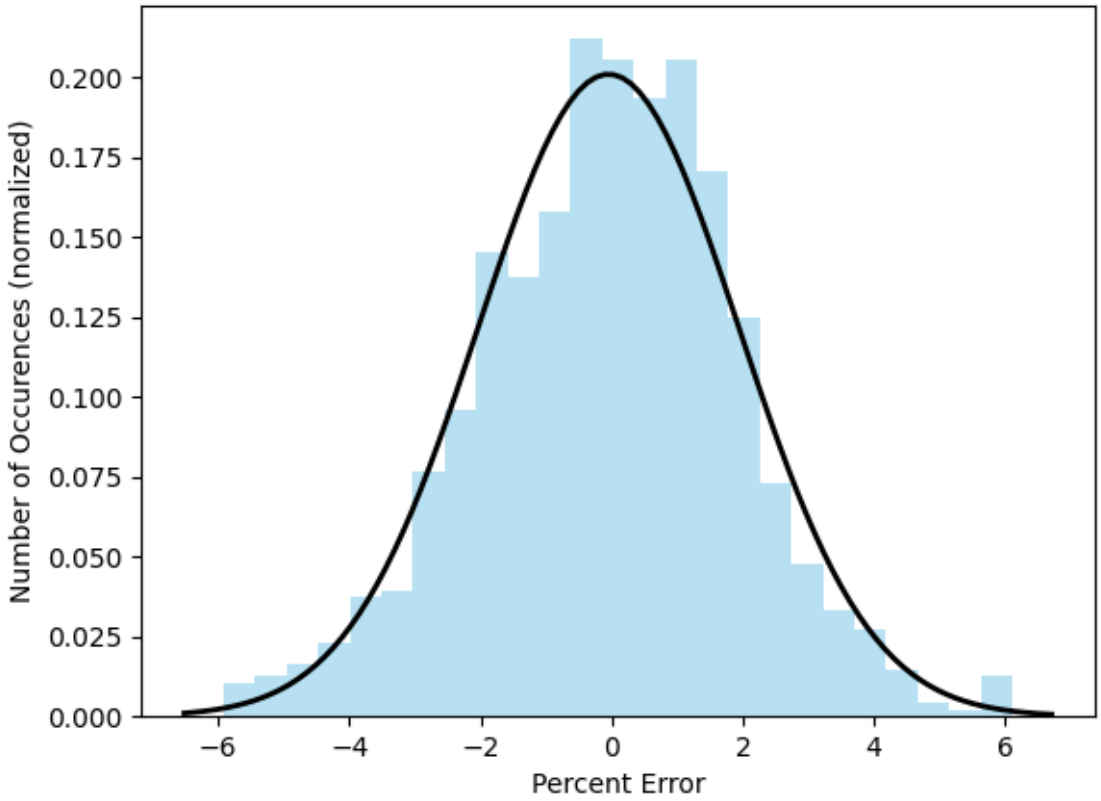

***Figure 8**: Example of the distribution of error for the intersection of a spherical shell with a cylindrical shell.*

RESULTS

Application to Galaxy Simulation Data

For this example, a dataset produced using the IllustrisTNG galaxy simulation was used. Figure 9 shows the plot of the number density of hydrogen, hot gas, and warm-hot intergalactic medium (WHIM) against the radius of the galaxy cluster sphere. This plot can be reproduced by running the script "IllustrisTNG Data Application Python Script" which is available in the appendix at the end of this document.

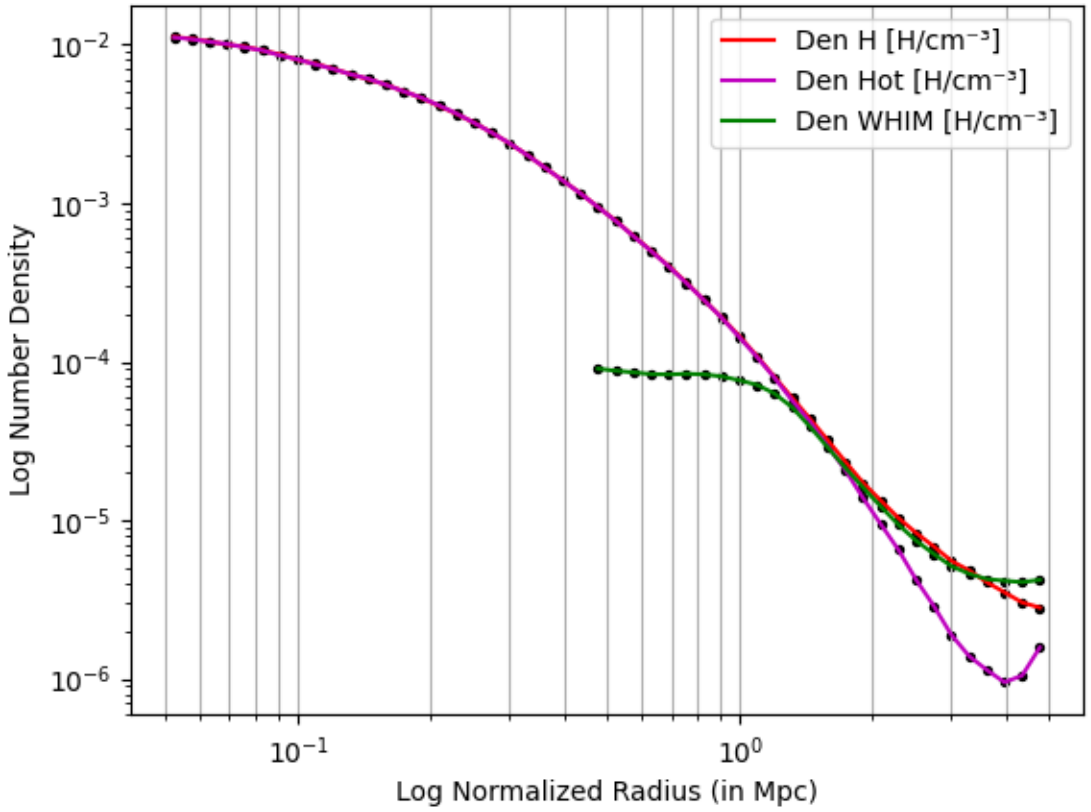

***Figure 9**: Density of values from the example dataset plotted against the radius.*

The values in this dataset can be projected onto the intersection between spherical shells in the cluster and cylindrical shells produced by the detector. Figure 10 shows one such example, where the number density of hydrogen is projected onto spherical shells from zero to the maximum spherical radius intersecting with a cylindrical shell with radii of 0.5 of the maximum spherical radius and 0.8 of the maximum spherical radius (for this example the maximum

spherical radius is 4.78 units, so the minimum and maximum cylindrical radius values come out to 2.39 units and 3.82 units, respectively). The plot does not include any values where the spherical radius is less than the minimum radius of the sphere, 2.49 units, as they are not included in the area of intersection.

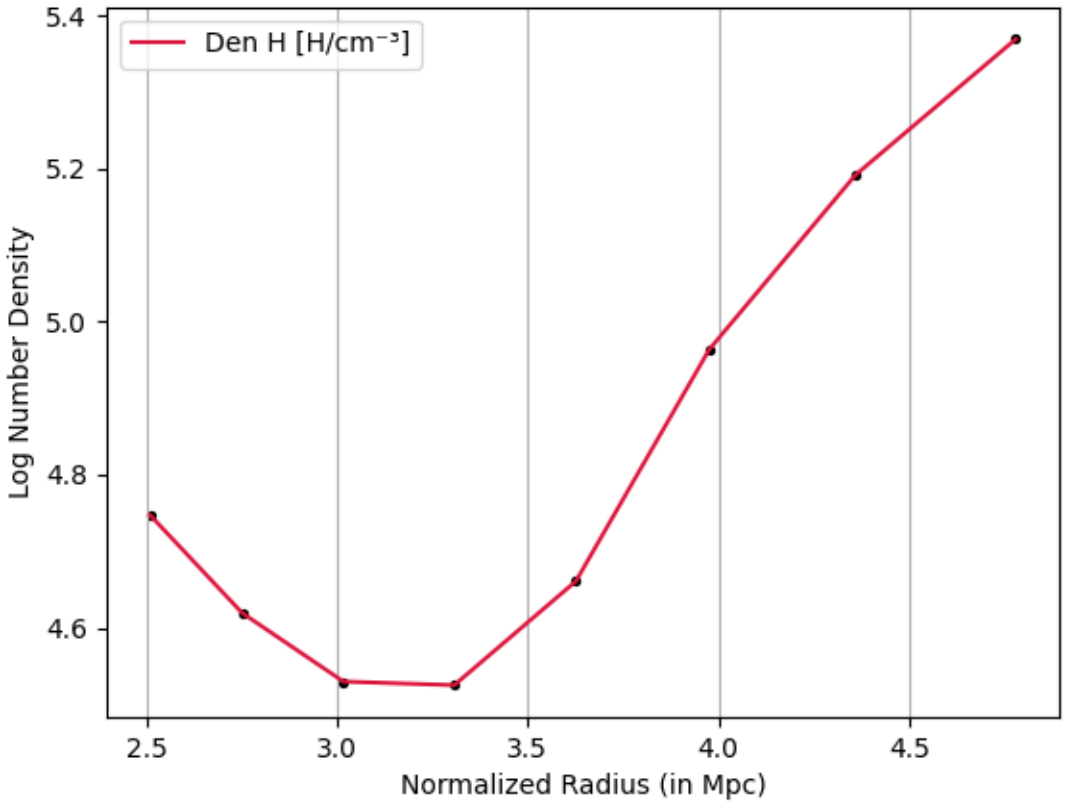

***Figure 10**: Graph of hydrogen number density included in the intersection of all spherical shells with a cylindrical shell of minimum radius 0.5 and maximum radius 0.8.*

The values plotted in Figure 10 can then be multiplied by the volume of the area in the intersection, which is useful if mass of hydrogen particles if the quantity of interest. Equation (8) is used to determine the volume between each intersection of the cylindrical shell with all the spherical shells. For both Figure 10 and Figure 11, any spherical shell where the maximum radius is less than that of the minimum radius of the cylindrical shell will have a volume of zero. This is because only the volumes that are included in the intersection are included. Since the

minimum radius of the cylinder is 2.39, all the spherical radii that are less than this do not appear in the plot.

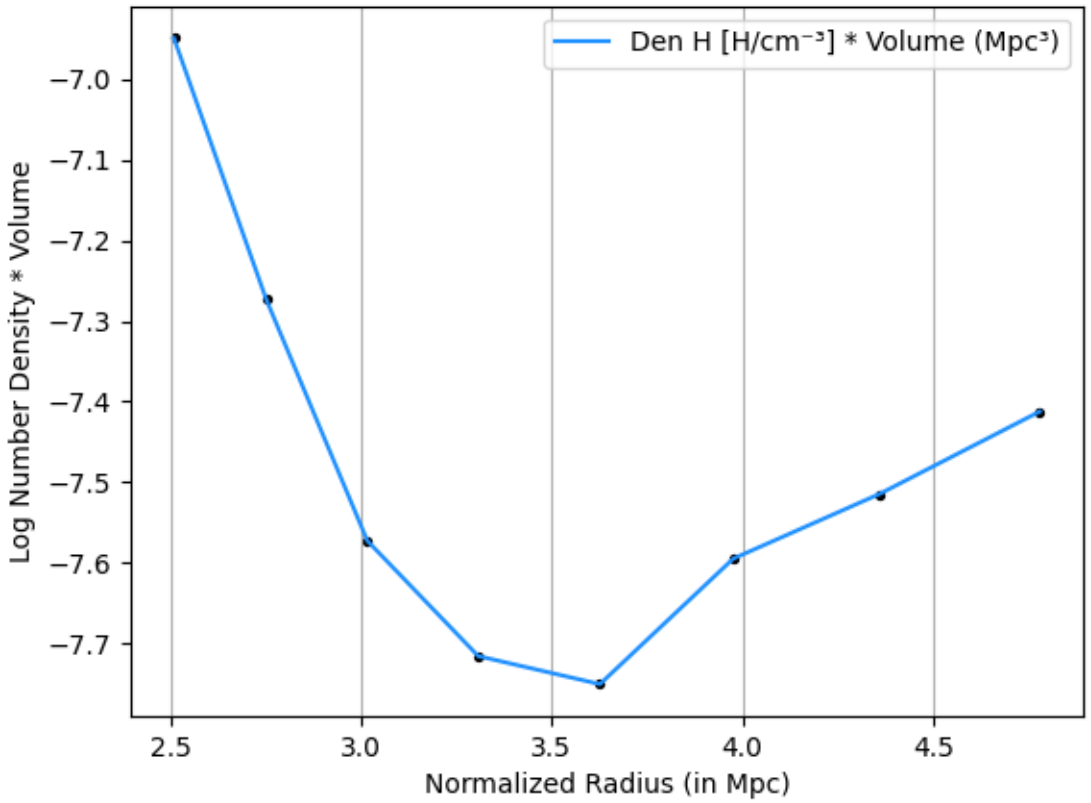

***Figure 11**: Graph of hydrogen number density multiplied by volume included in the intersection of all spherical shells with a cylindrical shell of minimum radius 0.5 and maximum radius 0.8.*

CONCLUSION

The goal of this project was to develop a method to project existing galaxy cluster simulation data onto a sphere intersected with a cylindrical shell (representing a detector, such as a telescope). This method was developed by deriving a formula for the volume of intersection of a spherical shell with set radii with a cylindrical shell with set radii.

The derived formula was verified using Python by comparing the volume produced by the formula to a volume produced by finding the volume from a ratio. This ratio was found by generating random data points, filtering them out if they were not included in the volume of

intersection of the spherical shell and the cylindrical shell, and comparing the total number of points to the number of points that were included in the intersection.

Once it was determined that the formula produced precise results, the data from the IllustrisTNG was added. The data consisted of a list of radius values with corresponding values for density of several quantities. The data for the density of hydrogen was combined with the volume of intersection between the spherical shells in the area of interest and the view from the detector to produce a plot of density vs. radius. Then each density value was multiplied by the corresponding volume in the intersection to produce a plot of mass vs. radius.

In the future, a code repository may be set up to allow other researchers to apply these scripts to their own datasets. It would be interesting to see how these scripts, as well as the overall concepts, apply to different datasets and simulations. It would also allow others to expand upon the use cases of these scripts and improve the accuracy of the results.

REFERENCES

"TNG Project Description." *IllustrisTNG* , https://www.tng-project.org/about/.

APPENDIX

Note: To run scripts, all packages must be downloaded onto local machine. Additional formatting may be needed, as indentations are part of the code in Python (code will not run/run correctly if lines are not indented correctly).

Intersection of Whole Sphere/Whole Cylinder and Intersection of Spherical Shell/Cylindrical Shell Python Script

```
import numpy as np
import matplotlib.pyplot as plt
import random
import math

# function to calculate percent error
def percent_error(actual,predicted):
	percentError = ((predicted-actual)/actual)*100
	return percentError

# function to calculate the volume of the entire region
def region_volume(R):
	regionVolume = (R+R)**3 #cube of all of the generated data
	return regionVolume

# function to calculate predicted volume based on data points
def predicted_volume(R,intersectionPoints,totalPoints):
	predictedVolume = region_volume(R)*intersectionPoints/totalPoints
	return predictedVolume

# function to calculate volume from derived formula
def actual_volume(R,rho):
	actualVolume = (4/3)*np.pi * (R**3 - (R**2 - rho**2)**(3/2))
	return actualVolume

def gen_simulation_vol(R,rho,totalDataPoints):
	#arrays for all values in the region
	x = []
	y = []
	z = []
	i = 0
	pointCount = 0

	# assign all of the values in the region to the arrays
	for i in range(totalDataPoints):
		x.append(random.uniform(-R,R))
```

```
            y.append(random.uniform(-R,R))
            z.append(random.uniform(-R,R))

            #arrays for values included in the intersection
            xint = [] # x-values
            yint = [] # y-values
            zint = [] # z-values

        for j in range(len(x)):
            # Formula for sphere
            if(math.sqrt((x[j]**2)+(y[j]**2)) <= rho):
                # Formula for cylinder
                if((x[j])**2 + (y[j])**2 + (z[j]**2) <= R**2):
                    # If (x,y,z) point is a part of both the sphere and cylinder, count it
                    xint.append(x[j])
                    yint.append(y[j])
                    zint.append(z[j])
                    pointCount += 1

        # the volume calculated from this simulation
        return predicted_volume(R,pointCount,totalDataPoints)

def main():
        x = []
        y = []
        z = []
        i = 0
        pointCountWhole = 0 #holds number of points that are included in the intersection
                            #for the whole sphere and cylinder
        pointCountShells = 0 #holds number of points that are included in the intersection
                            #for the spherical shell and cylindrical shell

        R = 1 # radius of the whole sphere
        rho = 0.5 # radius of the whole cylinder
        # radii of the spherical shell
        Rmax = 1
        Rmin = 0.8
        # radii of the cylindrical shell
        rhoMax = 0.8
        rhoMin = 0.5
        totalDataPoints = 10000 # number of particles in the area considered

        for i in range(totalDataPoints):
            x.append(random.uniform(-R,R))
```

```
        y.append(random.uniform(-R,R))
        z.append(random.uniform(-R,R))

        #arrays for values included in the intersection
        #of the whole sphere and the whole cylinder
        xintWhole = [] # x-values
        yintWhole = [] # y-values
        zintWhole = [] # z-values
        #arrays for values included in the intersection
        #of the spherical shell and cylindrical shell
        xintShells = [] # x-values
        yintShells = [] # y-values
        zintShells = [] # z-values

for j in range(len(x)):
        # Formula for sphere
        if(math.sqrt((x[j]**2)+(y[j]**2)) <= rho):
                # Formula for cylinder
                if((x[j])**2 + (y[j])**2 + (z[j]**2) <= R**2):
                        # If (x,y,z) point is a part of both count it
                        xintWhole.append(x[j])
                        yintWhole.append(y[j])
                        zintWhole.append(z[j])
                        pointCountWhole += 1

# print the results of the intersection of the whole sphere
# and the whole cylinder
print("The number of points in the intersection was: ",pointCountWhole)
print("The volume of the region is: ", region_volume(R))
simVol = predicted_volume(R,pointCountWhole,totalDataPoints)
simvol2 = gen_simulation_vol(R,rho,totalDataPoints)
print("The predicted volume of the intersection is: ", round(simVol,4))
calcVol = actual_volume(R,rho)
print("The actual volume of the intersection is: ", round(calcVol,4))
errorCalc = percent_error(calcVol,simVol)
print("The percent error for this calculation is: ", round(errorCalc,4), "%")

#graph of the points in the intersection
#of the whole sphere and the whole cylinder
fig = plt.figure()
fig = plt.figure(figsize=(12,10))
ax = plt.axes(projection='3d')
ax = fig.add_subplot(projection = '3d')
ax.scatter3D(xintWhole, yintWhole, zintWhole, s= 7, c ='b', marker = 'D')
```

```
        ax.set_xlim3d(-R, R)
        ax.set_ylim3d(-R, R)
        ax.set_zlim3d(-R, R)
        plt.show()

        for j in range(len(x)):
                # Formula for sphere
                # r^2 = (x-a)^2 + (y-b)^2 + (z-c)^2
                # a=b=c=0 because the center of the sphere is at (0,0,0)
                if(math.sqrt((x[j]**2)+(y[j]**2)) <= rhoMax and (x[j]**2)+(y[j]**2) >= rhoMin):
                        # Formula for cylinder
                        # x^2 + y^2 = r^2 and z=z
                        if((x[j])**2 + (y[j])**2 + (z[j]**2) <= Rmax**2 and (x[j])**2 + (y[j])**2 +
(z[j]**2) >= Rmin**2):
                                # If (x,y,z) point is a part of both the sphere and cylinder, count it
                                xintShells.append(x[j])
                                yintShells.append(y[j])
                                zintShells.append(z[j])
                                pointCountShells += 1

        #print the results for the intersection of the spherical shell
        #and the cylindrical shell
        print("The number of points in the intersection was: ",pointCountShells)
        print("The volume of the region is: ", region_volume(Rmax))
        simVol = gen_simulation_vol(Rmax,rhoMax,totalDataPoints) -
gen_simulation_vol(Rmin,rhoMin,totalDataPoints)
        print("The predicted volume of the intersection is: ", round(simVol,4))
        calcVol = actual_volume(Rmax,rhoMax)-actual_volume(Rmin,rhoMin)
        print("The actual volume of the intersection is: ", round(calcVol,4))
        errorCalc = percent_error(calcVol,simVol)
        print("The percent error for this calculation is: ", round(errorCalc,4), "%")

        #graph of the points in the intersection of the shells
        fig = plt.figure()
        fig = plt.figure(figsize=(12,10))
        ax = plt.axes(projection='3d')
        ax.scatter3D(xintShells, yintShells, zintShells, c= 'b', s= 7,  marker = 'D')
        ax.set_xlim3d(-Rmax, Rmax)
        ax.set_ylim3d(-Rmax, Rmax)
        ax.set_zlim3d(-Rmax, Rmax)
        plt.show()

if __name__=="__main__":
        main()
```

IllustrisTNG Data Application Python Script

#Note: to run this script without making any changes, the dataset used must be included in the #same directory as the script. The example data can be replaced with other datasets, but #modifications may need to be made to the script.

```
import numpy as np
import matplotlib.pyplot as plt

# function to calculate volume of intersection of whole sphere and cylinder
def volume(R,rho):
	actualVolume = (4/3)*np.pi * (R**3 - np.sign(R**2 - rho**2)*(np.abs(R**2 -
rho**2))**(3/2))
	return actualVolume

# function to calculate volume of intersection of  spherical shell with cylindrical shell
def shell_volume(Rmax,Rmin,rhoMax,rhoMin):
	calcVol = volume(Rmax,rhoMax)-volume(Rmin,rhoMin)
	return calcVol

def main():
	# read in the simulation data
	#R200 is a plain number (in Mpc)

	#RNH, NH: arrays, radius (r/r200) and density of H (# of atoms/cm^3)
	#NHErr: error in density
	#1. NHHOT, NHHOTErr: for hot gas (log T>7)
	#2. NHWHIM, NHWHIMErr: for the WHIM
	#3. NHWCGM, NHWCGMErr: for all gas with 5<log(T)<7
	R200, RNH, NH, NHErr, NHHOT, NHHOTErr, NHWHIM, NHWHIMErr, NHWCGM,
NHWCGMErr = np.load('ALL_CLUSTERS_PROFILE_LOG_nH.npy', allow_pickle=True)

	plt.loglog(RNH, NH, c='r',label = 'Den H [H/cm\u207b\u00b3]')
	plt.scatter(RNH, NH,s=8,color='black')
	plt.loglog(RNH, NHHOT, c='m', label = 'Den Hot [H/cm\u207b\u00b3]')
	plt.scatter(RNH, NHHOT,s=8,color='black')
	plt.loglog(RNH, NHWHIM, c='g',label = 'Den WHIM [H/cm\u207b\u00b3]')
	plt.scatter(RNH, NHWHIM,s=8,color='black')
	plt.grid(True, which="both", axis='x')
	plt.xlabel('Log Normalized Radius (in Mpc)')
	plt.ylabel('Log Number Density')
	plt.legend()
	plt.show()
```

```
        # radii of the cylindrical shell
        rhoMax = 0.8 * RNH[len(RNH)-1]
        rhoMin = 0.5 * RNH[len(RNH)-1]
        # radii of the spherical shell
        RMin = 0
        RMax = RNH[0]

        # array to hold the density * volume values
        projectionNH = []
        projectionNHV = []
        if(RMin < rhoMin):
                projectionNH.append(0)
                projectionNHV.append(0)
        else:
                projectionNH.append(shell_volume(RMax,RMin,rhoMax,rhoMin))
                projectionNHV.append(shell_volume(RMax,RMin,rhoMax,rhoMin)*NH[0])
        # populate the projection array from the data and their respective volumes
        for i in range(len(RNH)-1):
                if(RNH[i+1] < rhoMin):
                        projectionNH.append(0)
                        projectionNHV.append(0)
                else:
                        RMin = RNH[i]
                        RMax = RNH[i+1]
                        projectionNH.append(shell_volume(RMax,RMin,rhoMax,rhoMin))

        projectionNHV.append(shell_volume(RMax,RMin,rhoMax,rhoMin)*NH[i+1])

        # plot of number density vs. radius for shell intersection
        plt.plot(RNH, np.log(projectionNH), c='crimson', label = 'Den H [H/cm\u207b\u00b3]')
        plt.scatter(RNH, np.log(projectionNH),s=8,color='black')
        plt.grid(True, which="both", axis='x')
        plt.xlabel('Normalized Radius (in Mpc)')
        plt.ylabel('Log Number Density')
        plt.legend()
        plt.show()

        # plot of number density * volume for shell intersection
        plt.plot(RNH, np.log(projectionNHV), c='dodgerblue', label = 'Den H
[H/cm\u207b\u00b3] * Volume (Mpc\u00b3)')
        plt.scatter(RNH, np.log(projectionNHV),s=8,color='black')
        plt.grid(True, which="both", axis='x')
        plt.xlabel('Normalized Radius (in Mpc)')
```

```
        plt.ylabel('Log Number Density * Volume')
        plt.legend()
        plt.show()

if __name__=="__main__":
        main()
```