Computational Astrophysics (PHYS11037)

Cosmological *N*-body simulation II

A. Introduction

In this exercise, the density field of the Particle-Mesh *N*-body cosmological simulation from the previous exercise will be analyzed more quantitatively.

For the exercise, you will:

1. Quantify the distribution of the dark matter density fluctuations
2. Quantify the evolution of dark matter density fluctuations
3. Construct a dark matter halo catalog and image the haloes

The assignment write up should include:

1. A Methods section listing the commands used.
2. A Results section of your principal results, with answers to the questions below along with supporting figures and possibly tables.

B. Analysing the simulations

Move into your working directory on the CPLab system, for example:

```
> cd /storage/teaching/2021-22/compastro/snnnnnnn/PM
```

You'll need to answer some questions in this exercise. Open up a document to make a report with your answers. (You may use Libre Office: > loffice , or some other resource if you prefer.) **Please save the file as a PDF document finished.** You may upload the completed report to Learn to receive feedback.

0. To analyse the simulation results you will use yt. You might like to see what the simulation looks like. The yt online manual is at: http://yt-project.org/docs/3.0/

Either start up python or Python-3 from a Jupyter notebook (see previous notes).

As before, import basic yt modules into python:

```
>>> import yt
>>> from yt.units import *
```

Import the "overdensity" field:

```
>>>  from dm_overdensity import _overdensity
```

1. Load the z = 60 data file:

```
>>> ds0 = yt.load("RD0000/RD0000")
```

Add the “dm_overdensity” field to the data set:

```
>>> ds0.add_field(("gas",'dm_overdensity'),function=_overdensity)
```

and create a data container for all the data:

```
>>> ad0 = ds0.all_data()
```

Save the density and overdensity into YTArray's:

```
>>> den0 = ad0["all_cic"]
```

```
>>> oden0 = ad0[“dm_overdensity”]
```

Q1:
i. What are the minimum and maximum values of the density (g/ cm^3)? To see operations that may be performed on a YTArray, see:

http://yt-project.org/doc/reference/api/generated/yt.units.yt_array.YTArray.html?highlight=ytarray

ii. Under NGP grid assignment, what would be the minimum non-zero density you would expect? [Hint: the mass of a particle (in g) is given by:

```
>>> ad0["particle_mass"].mean()
```

iii. How does this compare with the minimum density in the simulation? Explain the difference, including their relative sizes.

iv. What are the minimum and maximum overdensities? Are these values reasonable for an initial density field? Briefly justify your answer.

2. Load the data at z = 4 and z = 0, and construct the overdensity arrays. Be sure to use distinct names for the arrays.

Q2:
i. What are the minimum and maximum overdensities for each of z = 4 and z = 0?
ii. Compare the variances in the overdensity at z = 60, z = 4 and z = 0. What is the trend with the expansion factor a = 1/ (1+z)? What is the physical origin of the trend?

How does the ratio at z = 0 to z = 60 compare with the ratio in the linear power spectra between z = 0 and z = 60 (see Assignment 1)? What is the mathematical relation between the two?

3. Compute histograms for the overdensities at z = 60, z = 4 and z = 0 and compare. This may conveniently be done using numpy, a python library of mathematics functions. Import the numpy library:

```
>>> import numpy as np
```

Convert the YTArrays for the overdensities to numpy arrays, eg, for the z = 60 data:

```
>>> np_oden0 = oden0.value
```

The numpy command to construct a histogram is

```
>>>hist1, bins = np.histogram(data,bins),
```

where 'data' is the numby array of data, and 'bins' are the bins. For example, for bins from 0-0.5, 0.5-1.0, 1.0-1.5 and 1.5-2.0, use bins=[0,0.5,1.,1.5,2.]. But you may wish to use more sensible bins. The histogram values may be written to a file hist1.dat using:

```
>>> np.savetxt('hist1.dat',hist1,fmt='%.12e')
```

To print the histogram instead as *n* data pairs of bin centre and histogram value, where *n* is the number of elements in *hist1*, given by:

```
>>> n = hist1.size
```

you could do:

```
>>> bincen = np.zeros(n)
>>> for i in range(n):
...     bincen[i] = 0.5*(bins[i]+bins[i+1])
```

to define *n* bin centres, and then write the data to file:

```
>>> f = open('hist1.dat','w')
>>> for i in range(n):
...     s = str("%12.4e %12.4e\n"% (bincen[i],hist1[i]))
...     f.write(s)
...
>>> f.close()
```

(The '...   ' above refers to a carriage return followed by a tab indentation, required in

python for a for-loop.)
In fact, because of the wide dynamic range in overdensities at z = 0, it is more convenient to make the histogram in $\log_{10}$. The numpy command is *np.log10(data)*. (You may need to add a very small number to the overdensities if any cells are empty, since log10(0) is undefined.)

Q3: How do the dark matter overdensity histograms compare between z = 60, z = 4 and z = 0? Make a table or plot the histograms and include in the report.

4. Collapsed haloes in which galaxies form have overdensities exceeding 200. These regions may be located by making an overdensity cut above a threshold of 200:

```
>>> dense_ad1 = ad1.cut_region(["obj['dm_overdensity'] > 200"])
```

Make projection plots confined to these regions at z = 4 and z = 0, and include in the report:

```
>>> proj1_dense = yt.ProjectionPlot(ds1, 'z', "dm_overdensity",
weight_field="dm_overdensity", data_source=dense_ad1)
```

Q4: Compare these with the corresponding projection plots showing the total dark matter overdensity fields at z = 4 and z = 0. How do the haloes relate to the overall structure of the dark matter distribution?

5. Much more sophisticated tools are available for finding collapsed haloes. The HOP method searches for coherent peaks by finding a single peak and then 'hopping' in the area until it judges it has found all the particles associated with the collapsed region. Find the haloes at z = 4 and z = 0 using the hop algorithm. Instructions are at: Halo finding Note that the documentation is in flux. For the version of yt on CPLab, the halo finding module is imported using:

```
>>> from yt.analysis_modules.halo_analysis.api import HaloCatalog
```

The halo-finding may take about 5 minutes each. [A runtime error may be reported. It may be ignored.]
The haloes are saved in the file "halo_catalogs/catalog/catalog.0.h5."

Methods for analysing the haloes are described at:
Halo analysis

Make plots of the haloes on "all_cic" (not "density") projection plots at z = 4 and z = 0 over the full simulation volume and include in your report.

Q5:
i. How many haloes does HOP find at z = 4 and z =0?

ii. Visually compare the distribution of HOP haloes with the overdense regions found in part 4 at $z = 4$ and $z = 0$. In which regions do they agree well? Where do they disagree?

Q6:
The halo masses may be filtered on mass to find the cumulative numbers above a given mass threshold.

First save the halo catalog by copying to a new file, eg catalog_all.0.h5. (You'll need a separate window to do this as a linux command.)

Now you may apply filters in sequence f*rom the least massive first* (why?) to work out the number of haloes above a set of thresholds. For example, to find the numbers above 1e11 $M_\odot$ (where $M_\odot$ is a solar mass), 3e11 $M_\odot$, 1e12 $M_\odot$ and 3e12 $M_\odot$, do (if the haloes are in the halo container my_halos):

```
>>> hc.add_filter("quantity_value", "particle_mass", ">", 1e11, "Msun")
>>> hc.create()
[yt will write the number of haloes to screen]
>>> hc.add_filter("quantity_value", "particle_mass", ">", 3e11, "Msun")
>>> hc.create()
```

and so on.

How many haloes are above the thresholds listed above?

The saved halo catalog(s) may be read into yt at a later session. This may be done using (eg for catalog.0.h5):

```
>>> from yt.analysis_modules.halo_analysis.api import HaloCatalog [if not yet done]
>>> my_halos = yt.load("halo_catalogs/catalog/catalog.0.h5")
>>> hc = HaloCatalog(halos_ds=my_halos, output_dir="halo_catalogs/catalog")
```

6. To finish up the yt session, quit python:

```
>>> quit()
```

7. Upload your report to Learn, PM assignment 2, to receive feedback. Be sure also to keep your results since you may find them helpful for the assessment.

C. References

Some useful online manual pages:

Enzo:

General Enzo documention:
http://enzo.readthedocs.io/

How to run a cosmology simulation:
Enzo cosmology simulation instructions

Additional scripts for a variety of problems are at:
Sample Enzo test problem scripts

yt: http://yt-project.org/doc/
To plot the density field:
http://yt-project.org/doc/visualizing/plots.html
http://yt-project.org/doc/cookbook/simple_plots.html

numpy manual: https://docs.scipy.org/doc/numpy/index.html
numpy basics: https://docs.scipy.org/doc/numpy/user/basics.html

python basics: http://www.afterhoursprogramming.com/tutorial/Python/Introduction/

Some useful literature:

1. Davis M., Efstathiou G., Frenk C. S., White, S. D. M. (1985) ApJ 292:371
“The evolution of large-scale structure in a universe dominated by cold dark matter”
(http://articles.adsabs.harvard.edu//full/1985ApJ...292..371D)

2. Efstathiou G., Davis M., White S.D.M., Frenk C.S. (1985) ApJS 57:241
"Numerical techniques for large cosmological N-body simulations"
(http://adsabs.harvard.edu/abs/1985ApJS...57..241E)

3. Peacock J. "Cosmological Physics" (1999, Cambridge University Press, Cambridge), Section 17.2.