Quijote simulations

Release 0.1

Francisco Villaescusa-Navarro

Apr 21, 2024

QUIJOTE

The Quijote simulations is a suite of more than 82,000 full N-body simulations designed to:

- Quantify the information content on cosmological observables
- Provide enough statistics to train machine learning algorithms

This video shows how changes in the cosmological parameters values affects the cosmic web:

We have run three simulations: 1) with the fiducial Quijote cosmology (shown on the left), 2) with a low value of $\Omega_{\rm m}$ equal to 0.2 (shown on the right), and 3) with a high value of the Hubble constant h equal to 0.9 (shown on the top after 1 minutes and 15 seconds). The initial phases of the simulations are all the same, and changes are driven by the differences in the value of the cosmological parameters. The simulation with high h has been flipped along the x axis to mimick the effect of a mirror along the middle of the screen. The video illustrates how changes in the value of the cosmological parameters induce differences in the positions, masses, and internal properties of dark matter halos. Quijote has been designed to quantify how well can we infer the value of the cosmological parameters given the statistical properties of the cosmic web.

Historically, Quijote was developed from the HADES simulations. Nowadays, it contains the full HADES data.

CHAPTER

ONE

NEWS

January 2024: Data from the Big Sobol Sequence (BSQ), a collection of 32,768 N-body simulations varying 5 cosmological parameters ($\Omega_{\rm m}$, $\Omega_{\rm b}$, h, n_s, σ_8) is made publicly available. See *Big Sobol Sequence* for details.

January 2024: Interested in modified gravity? Check *Modified Gravity* for Quijote-MG, a set of 4,048 N-body simulations with modified gravity.

November 2023: The Sancho suite, a collection of 240,000 galaxy mock catalogs in redshift-space spanning across 11 cosmologies, 3 massive neutrino cosmologies, 6 primordial non-Gaussianity amplitudes, and 11 Halo Occupation Distribution (HOD) models (together with their corresponding power spectra and bispectra) is now publicly available. Check *Sancho Suite* for details.

October 2023: Quijote now contains FoF halo catalogs that include the IDs of the particles belonging to the different halos. Check *Halo catalogs* for details.

September 2023: Data from the Ulagam simulations is made publicly available. Check the Ulagam website for details.

July 2023: We release Gigantes: the largest collection of detailed void catalog created to-date. Check the Gigantes website for details.

July 2023: We have run Rockstar on the Quijote snapshots and all the halo catalogs are publicly available. The rockstar catalogs are located in the New York cluster. Check *Data access* for details.

June 2023: We have created a series of tutorial to facilitate the usage of Quijote data. Check out *Tutorials* for details.

June 2023: We release Quijote-ODD, a collection of 1,000 N-body simulations with parity-violation initial conditions. We also make publicly available the FoF and Rockstar halo catalogs generated from these simulations. Check *Parity-violation* for more details.

March 2023: All data located in the Princeton cluster has been moved to the New York cluster. This means that all Quijote data is now accessible through binder. The Quijote snapshots have also being compressed by Lehman Garrison.

February 2023: We are moving all data located in Princeton to the cluster in New York. You may find some files temporarily missing. We are also compressing all Quijote snapshots due to storage limitations. You can still read the data with Pylians3, but if you are using hdf5 you need to use the hdf5plugin module to deal with the compression. See *Snapshots* for more details.

December 2022: We have created 3D matter overdensity grids for all PNG simulations and made them publicly available. Data can be accessed through Globus and binder.

October 2022: All the halo catalogs of the primordial non-Gaussianities simulations are now publicly available. Check *Primordial non-Gaussianities* for more details.

September 2022: All Quijote data located in the San Diego and New York clusters (almost 800 Terabytes) can now be accessed via Binder, a system that allows reading and manipulating the data without having to download it. Check *Data access* for further details.

July 2022: The snapshots of Quijote-PNG are now publicly available. Check *Primordial non-Gaussianities* for more details.

June 2022: The nwLH latin-hypercube, containing 2,000 simulations varying $\Omega_{\rm m}$, $\Omega_{\rm b}$, h, n_s, σ_8, M_ν, w is now publicly available! Check *Latin-hypercubes* for more details.

CHAPTER

TWO

SCIENCE

The Quijote simulations is a suite of more than 82,000 full N-body simulations that have been designed to accomplish two main goals:

- Quantify the information content on generic cosmological observables
- Provide enough data to train machine learning algorithms

For the first goal, Quijote provides a set of more than 40,000 simulations designed to calculate the information content on a generic cosmological observable by means of evaluating its Fisher matrix.

For the second goal, Quijote provides not only thousands of simulations on different latin-hypercubes and Sobol sequences, but the a total number of more than 82,000 N-body simulations, with billion of halos, galaxies, voids and millions of summary statistics such as power spectra, bispectra…et, to train machine learning algorithms, where having more data is always better.

The large number of simulations and data products available in Quijote allows many other scientific applications. See *Publications* for a list of different scientific usages of the data.

CHAPTER

THREE

FEATURES

- Simulations run with the TreePM code Gadget-III
- More than 60 Million CPU hours used
- Boxes of 1 Gpc/h. Combined total volume of more than 82,000 (Gpc/h)^3 at a single redshift
- 17,100 simulations for a fiducial Planck cosmology
- Between 500 and 1,000 simulations/cosmology for more than 30 different cosmologies
- 1,000 Separate Universe simulations
- 4,000 simulations with primordial non-Gaussianities
- 1,000 simulations with parity violation
- 4,048 simulations with modified gravity
- 8,000 simulations in different latin-hypercubes
- 32,768 simulations in a Sobol Sequence
- More than 12 trillions of particles at a single redshift from all simulations
- Billions of halos and voids identified
- Full snapshots at redshifts 0, 0.5, 1, 2, 3 and 127 (initial conditions)
- More than 300,000 halo catalogues
- More than 300,000 void catalogues
- More than 1 million power spectra
- More than 1 million bispectra
- More than 1 million correlation functions
- More than 1 million marked power spectra
- More than 1 million probability distribution functions
- More than 1 Petabyte of data publicly available
- All data can be downloaded via globus
- All data can be accessed and manipulated without downloading it via binder

CHAPTER

FOUR

DATA ACCESS

Quijote contains over 1 petabyte of data. Given this large size, the data is currently distributed across two different clusters in New York (Rusty cluster) and San Diego (GordonS cluster). The data can be accessed in two different ways:

- **Globus**. A system designed to easily transfer large amounts of data in a very efficient manner.
- **Binder**. A system that allows reading and manipulating the data online, without the need to download the data.

The table below describes the data each cluster contains and provides the links to the associated globus and binder systems.

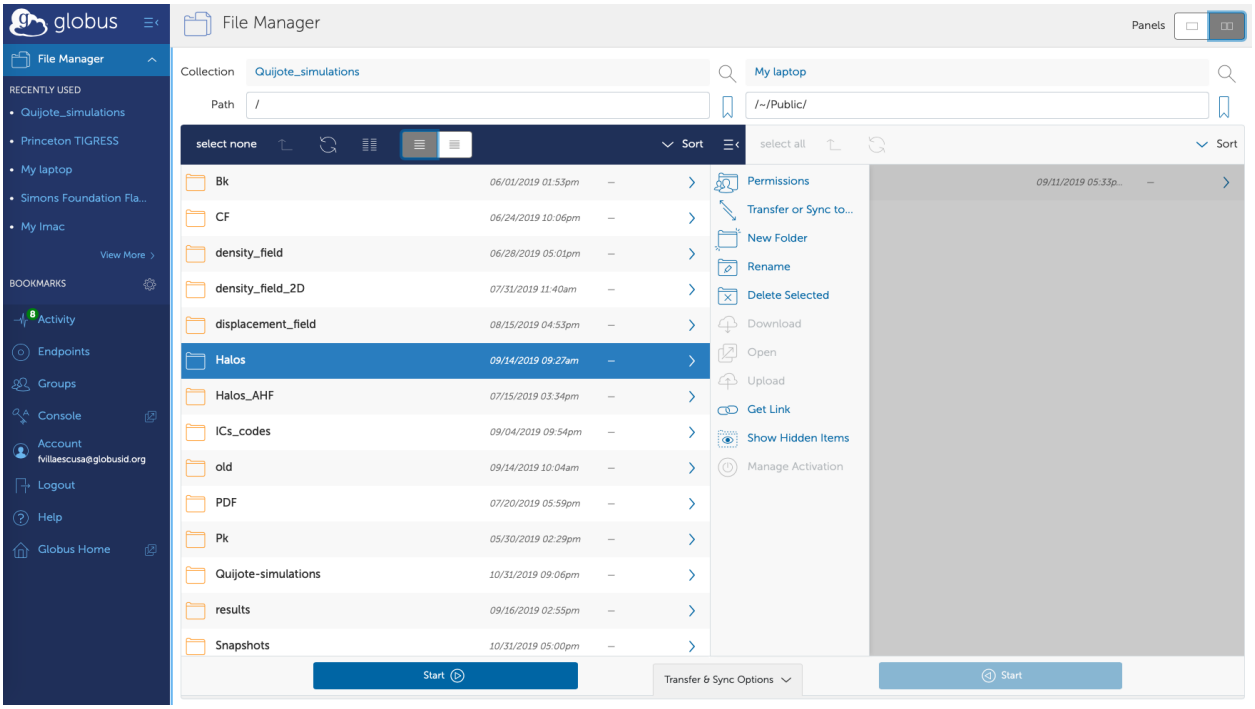
Warning: We are currently moving all data located in the Princeton cluster to New York. Besides, due to storage constrains we are compressing all snapshots. Thus, the data may be temporarily unavailable in the below links. Note that you need to install the latest version of Pylians, or use hdf5plugin to read the compressed snapshots. For more details see *Snapshots*. Please Reach out if you experience problems.

Cluster	Content	Access
San Diego	• The snapshots 8,000 - 14,999 of the fiducial cosmology • The snapshots of the standard & fixed LH latin hypercube • All spherical overdensity void catalogues • All power spectra • All bispectra • All correlation functions • All pdfs • 235 Terabytes	globus
New York	• The snapshots of all other simulations • All halo catalogs (FoF + Rockstar) • The 3D density fields • The HADES data (if available) • The 3D density fields • 700 Terabytes	globus

4.1 Globus

The data can be accessed through globus by clicking in the links from the above table. Note that to download the data to your local machine (e.g. laptop) you will need to install the globus connect personal. For further details see here. We now provide some simple instructions to use globus.

The simplest way to transfer data is to use the globus graphical environment. Just type the above names in collection (e.g. Quijote_simulations for the data in San Diego) or click the associated link. You will need to choose where the data is being moved in the other collection (e.g. your laptop or another supercomputer). Once the collection points are set, select the data you want to transfer and destiny folder and click in Start.



In some cases, there are so many files in a given directory, that globus may not be able to list them all and will return an error. If this is the case, it is advisable to use the path line. For instance, if by clicking in Snapshots you get a time out error, you may want to just type in the path line: `/Snapshots/` or `/~/Snapshots/`. This may show you the different content of the data and allow you to navigate it. You can also go to a given directory directly from there. E.g. to access the first realization of the fiducial cosmology, type in path: `/Snapshots/fiducial/0/` or `/~/Snapshots/fiducial/0/`.

In some cases, the above option may not be desirable. For instance, imagine that you want to download all linear matter power spectra of the high-resolution latin-hypercube simulations. One of such files (realization 45) is located in `/Snapshots/latin_hypercube_HR/45/ICs/Pk_mm_z=0.000.txt`, while the file for the realization 89 is located in `/Snapshots/latin_hypercube_HR/89/ICs/Pk_mm_z=0.000.txt`.

Thus, to download all those files without involving downloading the full HR latin-hypercube folder, will require that you access each simulation folder, then the ICs folder and then transfer the file individually. For 2,000 files this is unpractical. For these situations, we recommend using the globus Command Line Interface (CLI). The first step is to install the CLI package, if you don't have it. Next, login into globus by typing in a terminal

```
globus login
```

Then, the following command allow you to determine the associated endpoint of the Quijote simulations:

```
globus endpoint search "Quijote_simulations"
```

```
ID                                   | Owner                        | Display Name
------------------------------------ | ---------------------------- | --------------------
c42757fe-d570-11e9-98e2-0a63aa6b37da | fvillaescusa@globusid.org | Quijote_simulations
```

You should do the same to know the endpoint of the machine where you are transfering the data to. You can then explore the filesystem of the Quijote simulations (or your machine) as:

```
ep1=c42757fe-d570-11e9-98e2-0a63aa6b37da
globus ls $ep1:/Snapshots/latin_hypercube_HR/45/ICs/
```

The above command will list the content in the /Snapshots/latin_hypercube_HR/45/ICs/ directory. A single file can be transfered as:

```
ep1=c42757fe-d570-11e9-98e2-0a63aa6b37da
ep2=ddb59af0-6d04-11e5-ba46-22000b92c6ec
globus transfer $ep1:/Snapshots/latin_hypercube_HR/45/ICs/Pk_mm_z=0.000.txt $ep2:/
↪Quijote_simulations/linear_Pk/45/Pk_mm_z=0.000.txt --label "single file transfer"
```

Where ep2 should be the endpoint of the machine where you are transfering the data. Entire folders can be moved as follows:

```
ep1=c42757fe-d570-11e9-98e2-0a63aa6b37da
ep2=ddb59af0-6d04-11e5-ba46-22000b92c6ec
globus transfer $ep1:/Snapshots/latin_hypercube_HR/45/ICs $ep2:/Quijote_simulations/45/
↪ICs  --recursive --label "single folder transfer"
```

Many folders can be moved with a single command as

```
ep1=c42757fe-d570-11e9-98e2-0a63aa6b37da
ep2=ddb59af0-6d04-11e5-ba46-22000b92c6ec
globus transfer $ep1:/Snapshots/fiducial/ $ep2:/Quijote_simulations/fiducial/ --batch --
↪label "CLI 10 folders" < folders.txt
```

where folders.txt is a text file containing

```
--recursive 0 0
--recursive 1 1
--recursive 2 2
--recursive 3 3
--recursive 4 4
--recursive 5 5
--recursive 6 6
--recursive 7 7
--recursive 8 8
--recursive 9 9
```

For more options and details see Command Line Interface (CLI).

4.2 Binder

Binder is a system that allows users to read and manipulate data that is hosted at the Flatiron Institute through either a Jupyter notebook or a unix shell. The user can find some basic documentation here. The links to the binder for the New York and San Diego cluster can be found in the table above. Note that the data in the Princeton cluster cannot be accessed through binder. Our binder environments contains the following packages:

- nbgitpuller
- sphinx-gallery
- pandas
- matplotlib
- astropy
- matplotlib
- scipy
- h5py
- corner
- future
- numba
- unyt
- Pylians
- pyfftw
- CAMELS-library

Note: The first time you log into binder it could take a while. This is because the system is downloading and installing all required packages. Clicking show you can see the progress.

Warning: Two important things need to be taken into account when using Binder. First, the Binder environment is ephemeral - after a few days of inactivity its contents are deleted, so one has to be vigilant about downloading any analysis results in time. Second, Binder is not designed to carry out long and heavy calculations. In this case we recommend the user to download the data and work with it locally.

CHAPTER

FIVE

DATA ORGANIZATION

The Quijote data is organized into different folders:

- **Snapshots**. This folder contains the snapshots of the simulations
- **Halos**. This folder contains the halo catalogues (both FoF and Rockstar)
- **Voids**. This folder contains the void catalogues
- **Linear_Pk**. This folder contains the linear power spectra of each cosmological model
- **Pk**. This folder contains the non-linear power spectra
- **Marked_Pk**. This folder contains the marked power spectra
- **Bk**. This folder contains the bispectra
- **CF**. This folder contains the correlation functions
- **PDF**. This folder contains the pdfs
- **3D_cubes**. This folder contains the 3D density fields

Each of the above folders contain several subfolders, that represent the different cosmological models, e.g. `h_p`, `fiducial`, and `Om_m`. See *Structure and types* for a description of the different cosmological models and simulations present in Quijote.

CHAPTER

SIX

TUTORIALS

We provide multiple tutorials showing how to read and manipulate Quijote data. Inside each tutorial there is a link to open the notebook in our binder, that contains all Quijote data.

6.1 Reading snapshots

[1]:
```
import numpy as np
import readgadget
```

get the name of the snapshot

[2]:
```
snapshot = '/home/jovyan/Data/Snapshots/Om_p/32/snapdir_004/snap_004'
```

read the header of the snapshot

[3]:
```
# read header
header   = readgadget.header(snapshot)
BoxSize  = header.boxsize/1e3  #Mpc/h
Nall     = header.nall         #Total number of particles
Masses   = header.massarr*1e10 #Masses of the particles in Msun/h
Omega_m  = header.omega_m      #value of Omega_m
Omega_l  = header.omega_l      #value of Omega_l
h        = header.hubble       #value of h
redshift = header.redshift     #redshift of the snapshot
Hubble   = 100.0*np.sqrt(Omega_m*(1.0+redshift)**3+Omega_l)#Value of H(z) in km/s/(Mpc/h)

print('BoxSize = %.3f Mpc/h'%BoxSize)
print('Total number of particles:',Nall)
print('Masses of the particles:',Masses, 'Msun/h')
print('Omega_m = %.3f'%Omega_m)
print('Omega_L = %.3f'%Omega_l)
print('h = %.3f'%h)
print('redshift = %.3f'%redshift)
print('H(z=%.1f)=%.3f (km/s)/(Mpc/h)'%(redshift,Hubble))
```

```
BoxSize = 1000.000 Mpc/h
Total number of particles: [        0 134217728         0         0         0         0]
Masses of the particles: [0.00000000e+00 6.77240019e+11 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00] Msun/h
```

(continues on next page)

(continued from previous page)

```
Omega_m = 0.328
Omega_L = 0.672
h = 0.671
redshift = 0.000
H(z=0.0)=100.000 (km/s)/(Mpc/h)
```

For N-body simulations, we only care about particle type 1 (and type 2 if neutrinos are included)

[4]:
```
mass_c = Masses[1]
N_c = Nall[1]
print('Mass of a DM particle = %.3e Msun/h'%mass_c)
print('Number of DM particles = %d'%N_c)
```

```
Mass of a DM particle = 6.772e+11 Msun/h
Number of DM particles = 134217728
```

[5]:
```
# we can check the value of Omega_m
rho_crit = 2.775e11 #critical density at z=0 in (Msun/h)/(Mpc/h)^3
estimated_Omega_m = N_c*mass_c/BoxSize**3/rho_crit
print('%.4f should be similar to\n%.4f'%(estimated_Omega_m,Omega_m))
```

```
0.3276 should be similar to
0.3275
```

Now lets read the positions, velocities, and IDs of the DM particles

[6]:
```
ptype = [1] #DM is 1, neutrinos is [2]
pos   = readgadget.read_block(snapshot, "POS ", ptype)/1e3 #positions in Mpc/h
vel   = readgadget.read_block(snapshot, "VEL ", ptype)     #peculiar velocities in km/s
ids   = readgadget.read_block(snapshot, "ID  ", ptype)-1   #IDs starting from 0
```

Lets print some information about these quantities

[7]:
```
print('%.3f < X  < %.3f Mpc/h'%(np.min(pos[:,0]), np.max(pos[:,0])))
print('%.3f < Y  < %.3f Mpc/h'%(np.min(pos[:,1]), np.max(pos[:,1])))
print('%.3f < Z  < %.3f Mpc/h'%(np.min(pos[:,2]), np.max(pos[:,2])))
print('%.3f < Vx < %.3f km/s'%(np.min(vel[:,0]), np.max(vel[:,0])))
print('%.3f < Vy < %.3f km/s'%(np.min(vel[:,1]), np.max(vel[:,1])))
print('%.3f < Vz < %.3f km/s'%(np.min(vel[:,2]), np.max(vel[:,2])))
print('%d < IDs < %d'%(np.min(ids), np.max(ids)))
```

```
0.000 < X  < 999.992 Mpc/h
0.000 < Y  < 999.992 Mpc/h
0.000 < Z  < 999.992 Mpc/h
-4777.000 < Vx < 5332.000 km/s
-4387.000 < Vy < 4999.000 km/s
-4977.000 < Vz < 4632.000 km/s
0 < IDs < 134217727
```

You can get the position, velocity, and ID of a particle just by calling its index

[8]:
```
# lets consider the particle number 10
print('position =',pos[10],'Mpc/h')
```

(continues on next page)

(continued from previous page)

```
print('velocity =',vel[10],'km/s')
print('ID =',ids[10])
```

```
position = [  9.89725 996.024    15.1425 ] Mpc/h
velocity = [ 356.25  -327.125 -153.   ] km/s
ID = 10
```

The particles IDs can be used to track particles across times. Lets take the particle with ID equal to 623 and find its position across redshifts

[9]:
```
part_ID = 620
for snapnum in [0,1,2,3,4]:
    snapshot = '/home/jovyan/Data/Snapshots/Om_p/32/snapdir_%03d/snap_%03d'%(snapnum,
↪snapnum)

    # read header
    header   = readgadget.header(snapshot)
    redshift = header.redshift     #redshift of the snapshot

    # read positions and ids
    pos = readgadget.read_block(snapshot, "POS ", [1])/1e3 #positions in Mpc/h
    ids = readgadget.read_block(snapshot, "ID  ", [1])-1   #IDs starting from 0

    index = np.where(ids==part_ID)[0]
    position = pos[index][0]
    print('z=%.1f -----> (X,Y,Z)=(%.2f, %.2f, %.2f) Mpc/h'%(redshift,position[0],
↪position[1],position[2]))
```

```
z=3.0 -----> (X,Y,Z)=(2.14, 16.49, 86.31) Mpc/h
z=2.0 -----> (X,Y,Z)=(2.87, 16.15, 86.48) Mpc/h
z=1.0 -----> (X,Y,Z)=(4.18, 15.80, 86.84) Mpc/h
z=0.5 -----> (X,Y,Z)=(4.97, 15.25, 87.17) Mpc/h
z=0.0 -----> (X,Y,Z)=(6.31, 14.28, 87.94) Mpc/h
```

Keep in mind the simulations have periodic boundary conditions. For instance, this is the incorrect and correct way to compute the distance between them

[10]:
```
particle1 = pos[3]
particle2 = pos[4]
print('Position of particle 1: (%.3f, %.3f, %.3f) Mpc/h'%(particle1[0], particle1[1],␣
↪particle1[2]))
print('Position of particle 2: (%.3f, %.3f, %.3f) Mpc/h'%(particle2[0], particle2[1],␣
↪particle2[2]))
```

```
Position of particle 1: (4.534, 3.950, 998.616) Mpc/h
Position of particle 2: (4.922, 4.519, 2.621) Mpc/h
```

[11]:
```
# this would be the incorrect way to compute the distance
d = np.sqrt(np.sum((particle1-particle2)**2))
print('Incorrect distance = %.3f Mpc/h'%d)

# this would be the correct way to compute the distance
d = particle1-particle2
indexes = np.where(d>BoxSize/2)
```

(continues on next page)

(continued from previous page)

```
d[indexes]-=BoxSize
indexes = np.where(d<-BoxSize/2)
d[indexes]+=BoxSize
d = np.sqrt(np.sum(d**2))
print('Correct distance   = %.3f Mpc/h'%d)
```

```
Incorrect distance = 995.995 Mpc/h
Correct distance   = 4.064 Mpc/h
```

In simulations with massive neutrinos, you can read both dark matter and neutrino positions, velocities, and IDs

[12]:
```
# get the name of the snapshot
snapshot = '/home/jovyan/Data/Snapshots/Mnu_p/284/snapdir_002/snap_002'

# read header
header   = readgadget.header(snapshot)
BoxSize  = header.boxsize/1e3  #Mpc/h
Nall     = header.nall         #Total number of particles
Masses   = header.massarr*1e10 #Masses of the particles in Msun/h
Omega_m  = header.omega_m      #value of Omega_m
Omega_l  = header.omega_l      #value of Omega_l
h        = header.hubble       #value of h
redshift = header.redshift     #redshift of the snapshot
Hubble   = 100.0*np.sqrt(Omega_m*(1.0+redshift)**3+Omega_l)#Value of H(z) in km/s/(Mpc/h)

print('BoxSize = %.3f Mpc/h'%BoxSize)
print('Total number of particles:',Nall)
print('Masses of the particles:',Masses, 'Msun/h')
print('Omega_m = %.3f'%Omega_m)
print('Omega_L = %.3f'%Omega_l)
print('h = %.3f'%h)
print('redshift = %.3f'%redshift)
print('H(z=%.1f)=%.3f (km/s)/(Mpc/h)'%(redshift,Hubble))
```

```
BoxSize = 1000.000 Mpc/h
Total number of particles: [        0 134217728 134217728         0         0         0]
Masses of the particles: [0.00000000e+00 6.51631041e+11 4.92989376e+09 0.00000000e+00
 0.00000000e+00 0.00000000e+00] Msun/h
Omega_m = 0.318
Omega_L = 0.682
h = 0.671
redshift = 1.000
H(z=1.0)=179.513 (km/s)/(Mpc/h)
```

As can be seeing, particle type 2 (neutrinos) have millions particles and the masses are not zero

[13]:
```
mass_c = Masses[1]
mass_n = Masses[2]
N_c = Nall[1]
N_n = Nall[2]
print('Mass of a DM particle = %.3e Msun/h'%mass_c)
print('Mass of a NU particle = %.3e Msun/h'%mass_n)
print('Number of DM particles = %d'%N_c)
```

(continues on next page)

(continued from previous page)

```
print('Number of NU particles = %d'%N_n)

Omega_m_estimated = (N_c*mass_c + N_n*mass_n)/BoxSize**3/rho_crit
Omega_c_estimated = (N_c*mass_c)/BoxSize**3/rho_crit
Omega_n_estimated = (N_n*mass_n)/BoxSize**3/rho_crit
print('Omega_cb = %.3f'%Omega_c_estimated)
print('Omega_nu = %.3e'%Omega_n_estimated)
print('Omega_m  = %.3f'%Omega_m_estimated)
```

```
Mass of a DM particle = 6.516e+11 Msun/h
Mass of a NU particle = 4.930e+09 Msun/h
Number of DM particles = 134217728
Number of NU particles = 134217728
Omega_cb = 0.315
Omega_nu = 2.384e-03
Omega_m  = 0.318
```

Now lets read the positions, velocities, and IDs of the dark matter and neutrino particles

[14]:
```
pos_c = readgadget.read_block(snapshot, "POS ", [1])/1e3 #positions in Mpc/h
vel_c = readgadget.read_block(snapshot, "VEL ", [1])     #peculiar velocities in km/s
ids_c = readgadget.read_block(snapshot, "ID  ", [1])-1   #IDs starting from 0

pos_n = readgadget.read_block(snapshot, "POS ", [2])/1e3 #positions in Mpc/h
vel_n = readgadget.read_block(snapshot, "VEL ", [2])     #peculiar velocities in km/s
ids_n = readgadget.read_block(snapshot, "ID  ", [2])-1   #IDs starting from 0
```

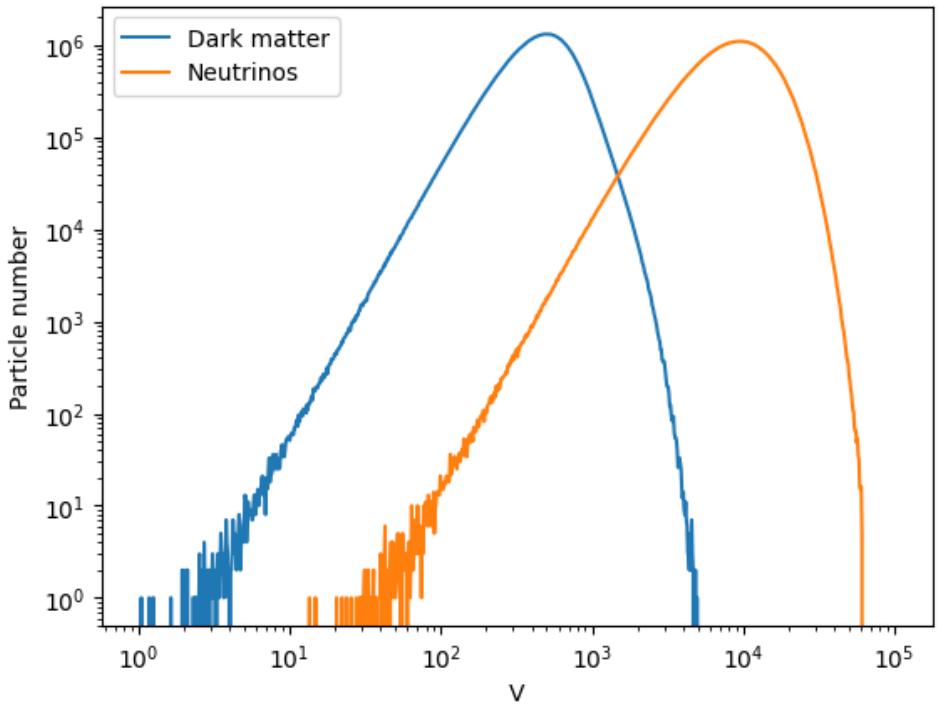
Lets make a plot with the distribution of the dark matter and neutrino velocities

[15]:
```
# lets compute the modulus of the dark matter and neutrino velocities
Vc = np.sqrt(vel_c[:,0]**2 + vel_c[:,1]**2 + vel_c[:,2]**2)
Vn = np.sqrt(vel_n[:,0]**2 + vel_n[:,1]**2 + vel_n[:,2]**2)
print('%.3f < Vc < %.3f'%(np.min(Vc), np.max(Vc)))
print('%.3f < Vn < %.3f'%(np.min(Vn), np.max(Vn)))

bins_histo = np.logspace(0,5,1000)
histo_Vc, edges = np.histogram(Vc, bins_histo)
histo_Vn, edges = np.histogram(Vn, bins_histo)
```

```
0.811 < Vc < 4902.378
13.396 < Vn < 60315.773
```

As can be seen, neutrinos have, on average, larger velocities than dark matter

[16]:
```
import matplotlib.pyplot as plt
plt.xscale('log')
plt.yscale('log')
plt.xlabel('V')
plt.ylabel('Particle number')
plt.plot(edges[1:], histo_Vc)
plt.plot(edges[1:], histo_Vn)
plt.legend(['Dark matter', 'Neutrinos'])
plt.show()
```



nbsphinx-code-borderwhite

6.2 Reading FoF halo catalogs

[1]:
```
import numpy as np
import readgadget
import readfof
import redshift_space_library as RSL
```

[2]:
```
snapdir = '/home/jovyan/Data/Halos/FoF/EQ_p/420' #folder hosting the catalogue
snapnum = 4                                      #number of the catalog (4-->z=0, 3-->
↪z=0.5, 2-->z=1, 1-->z=2, 0-->z=3)
```

In Quijote, snapnum={4,3,2,1,0} corresponds to redshift {0, 0.5, 1, 2, 3}, but we recommend reading it directly from the header of the corresponding snapshot. The header of the snapshot also contains information that is needed for some particular operations such as move halos to redshift-space.

[3]:
```
# get the name of the corresponding snapshot
snapshot = '/home/jovyan/Data/Snapshots/EQ_p/420/snapdir_%03d/snap_%03d'%(snapnum,
↪snapnum)

# read the redshift, boxsize, cosmology...etc in the header
header   = readgadget.header(snapshot)
BoxSize  = header.boxsize/1e3  #Mpc/h
Nall     = header.nall         #Total number of particles
Masses   = header.massarr*1e10 #Masses of the particles in Msun/h
Omega_m  = header.omega_m      #value of Omega_m
```

(continues on next page)

(continued from previous page)

```
Omega_l  = header.omega_l      #value of Omega_l
h        = header.hubble       #value of h
redshift = header.redshift     #redshift of the snapshot
Hubble   = 100.0*np.sqrt(Omega_m*(1.0+redshift)**3+Omega_l)#Value of H(z) in km/s/(Mpc/h)

print('BoxSize = %.3f Mpc/h'%BoxSize)
print('Number of particles in the snapshot:',Nall)
print('Omega_m = %.3f'%Omega_m)
print('Omega_l = %.3f'%Omega_l)
print('h = %.3f'%h)
print('redshift = %.1f'%redshift)
```

```
BoxSize = 1000.000 Mpc/h
Number of particles in the snapshot: [        0 134217728         0         0         0
↪        0]
Omega_m = 0.318
Omega_l = 0.682
h = 0.671
redshift = 0.0
```

To read the halo catalog we do

[4]:
```
# read the halo catalogue
FoF = readfof.FoF_catalog(snapdir, snapnum, long_ids=False,
                          swap=False, SFR=False, read_IDs=False)

# get the properties of the halos
pos_h  = FoF.GroupPos/1e3            #Halo positions in Mpc/h
vel_h  = FoF.GroupVel*(1.0+redshift) #Halo peculiar velocities in km/s
mass_h = FoF.GroupMass*1e10          #Halo masses in Msun/h
Np_h   = FoF.GroupLen                #Number of CDM particles in the halo. Even in
↪simulations with massive neutrinos, this will be just the number of CDM particles
```

Lets print some information

[5]:
```
print('%.3f < X_h  < %.3f Mpc/h'%(np.min(pos_h[:,0]), np.max(pos_h[:,0])))
print('%.3f < Y_h  < %.3f Mpc/h'%(np.min(pos_h[:,1]), np.max(pos_h[:,1])))
print('%.3f < Z_h  < %.3f Mpc/h'%(np.min(pos_h[:,2]), np.max(pos_h[:,2])))
print('%.3f < Vx_h < %.3f km/s'%(np.min(vel_h[:,0]), np.max(vel_h[:,0])))
print('%.3f < Vy_h < %.3f km/s'%(np.min(vel_h[:,1]), np.max(vel_h[:,1])))
print('%.3f < Vz_h < %.3f km/s'%(np.min(vel_h[:,2]), np.max(vel_h[:,2])))
print('%.3e < M_h < %.3e Msun/h'%(np.min(mass_h), np.max(mass_h)))
print('%d < Np < %d'%(np.min(Np_h), np.max(Np_h)))
```

```
0.003 < X_h  < 999.997 Mpc/h
0.004 < Y_h  < 999.997 Mpc/h
0.001 < Z_h  < 1000.000 Mpc/h
-1920.503 < Vx_h < 2176.723 km/s
-1891.656 < Vy_h < 2023.595 km/s
-1776.037 < Vz_h < 1855.787 km/s
1.313e+13 < M_h < 4.270e+15 Msun/h
20 < Np < 6504
```

By construction, we only keep halos that contain at least 20 dark matter particles. We can verify that the minimum

mass of a halo corresponds to that

[6]:
```
Minimum_mass = 20*Masses[1] #This is 20 times the mass of a single DM particle
print('%.3e should be equal to\n%.3e'%(Minimum_mass, np.min(mass_h)))
```

```
1.313e+13 should be equal to
1.313e+13
```

To get the information about a particular halo, just select its index

[7]:
```
index = 45 #the index of the halo
print('Halo position:',pos_h[index],'Mpc/h')
print('Halo velocity:',vel_h[index],'km/s')
print('Halo mass: %.3e Msun/h'%mass_h[index])
print('Number of particles in the halo: %d'%Np_h[index])
```

```
Halo position: [557.547   854.5584  387.07922] Mpc/h
Halo velocity: [ 231.89758 -459.18256  655.8717 ] km/s
Halo mass: 1.910e+15 Msun/h
Number of particles in the halo: 2909
```

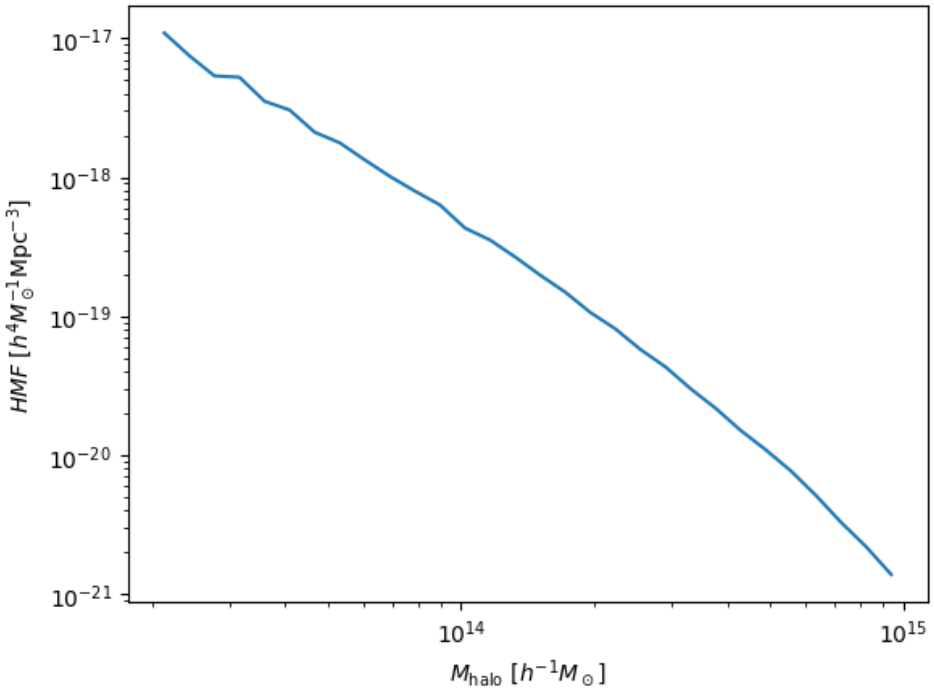
In some cases, we may want to work with halos in redshift-space rather than real-space. We can move halos to redshift-space along a given axis as follows

[8]:
```
# move halos to redshift-space. After this call, pos_h will contain the
# positions of the halos in redshift-space
axis = 0 #axis along which to displace halos
RSL.pos_redshift_space(pos_h, vel_h, BoxSize, Hubble, redshift, axis)
```

Lets finally compute a simple summary statistics from the catalog: the halo mass function

[9]:
```
min_mass = 2e13 #minimum mass in Msun/h
max_mass = 1e15 #maximum mass in Msun/h
bins     = 30   #number of bins in the HMF

# Correct the masses of the FoF halos
mass_h = mass_h*(1.0-Np_h**(-0.6))

bins_mass = np.logspace(np.log10(min_mass), np.log10(max_mass), bins+1)
mass_mean = 10**(0.5*(np.log10(bins_mass[1:])+np.log10(bins_mass[:-1])))
dM        = bins_mass[1:] - bins_mass[:-1]

# compute the halo mass function (number of halos per unit volume per unit mass)
HMF = np.histogram(mass_h, bins=bins_mass)[0]/(dM*BoxSize**3)
```

[10]:
```
import matplotlib.pyplot as plt
plt.xscale('log')
plt.yscale('log')
plt.xlabel(r'$M_{\rm halo}~[h^{-1}M_\odot]$')
plt.ylabel(r'$HMF~[h^4M_\odot^{-1}{\rm Mpc}^{-3}]$')
plt.plot(mass_mean, HMF)
plt.show()
```



nbsphinx-code-borderwhite

6.3 Creating density fields

[1]:
```
import numpy as np
import readgadget
import MAS_library as MASL
```

Define the value of the parameters

[2]:
```
snapshot = '/home/jovyan/Data/Snapshots/fiducial/0/snapdir_004/snap_004' #location of␣
↪the snapshot
grid     = 512    #the density field will have grid^3 voxels
MAS      = 'CIC'  #Mass-assignment scheme:'NGP', 'CIC', 'TSC', 'PCS'
verbose  = True   #whether to print information about the progress
ptype    = [1]    #[1](CDM), [2](neutrinos) or [1,2](CDM+neutrinos)
```

Read the header and the particle positions

[3]:
```
# read header
header   = readgadget.header(snapshot)
BoxSize  = header.boxsize/1e3  #Mpc/h
redshift = header.redshift     #redshift of the snapshot
Masses   = header.massarr*1e10 #Masses of the particles in Msun/h

# read positions, velocities and IDs of the particles
pos = readgadget.read_block(snapshot, "POS ", ptype)/1e3 #positions in Mpc/h
```

Print some information about the data

[4]:
```
print('BoxSize: %.3f Mpc/h'%BoxSize)
print('Redshift: %.3f'%redshift)
print('%.3f < X < %.3f'%(np.min(pos[:,0]), np.max(pos[:,0])))
print('%.3f < Y < %.3f'%(np.min(pos[:,1]), np.max(pos[:,1])))
print('%.3f < Z < %.3f'%(np.min(pos[:,2]), np.max(pos[:,2])))
```

```
BoxSize: 1000.000 Mpc/h
Redshift: 0.000
0.000 < X < 999.992
0.000 < Y < 999.992
0.000 < Z < 999.992
```

Define the matrix that will contain the value of the density / overdensity field

[5]:
```
delta = np.zeros((grid,grid,grid), dtype=np.float32)
```

Now construct the 3D density field

[6]:
```
# construct 3D density field
MASL.MA(pos, delta, BoxSize, MAS, verbose=verbose)
```

```
Using CIC mass assignment scheme
Time taken = 6.275 seconds
```

We can make some tests to make sure the density field has been computed properly

[7]:
```
# the sum of the values in all voxels should be equal to the number of particles
print('%.3f should be equal to\n%.3f'%(np.sum(delta, dtype=np.float64), pos.shape[0]))
```

```
134217728.019 should be equal to
134217728.000
```

As this point, delta contains the effective number of particles in each voxel. If you want instead the effective mass in each voxel you can just do

[8]:
```
delta *= Masses[1]

# now check that the mass in the density field is equal to the total mass in the␣
↪simulation
print('%.3e should be equal to\n%.3e'%(np.sum(delta, dtype=np.float64), pos.
↪shape[0]*Masses[1]))
```

```
8.812e+19 should be equal to
8.812e+19
```

Lets take a slice in the cube and plot it

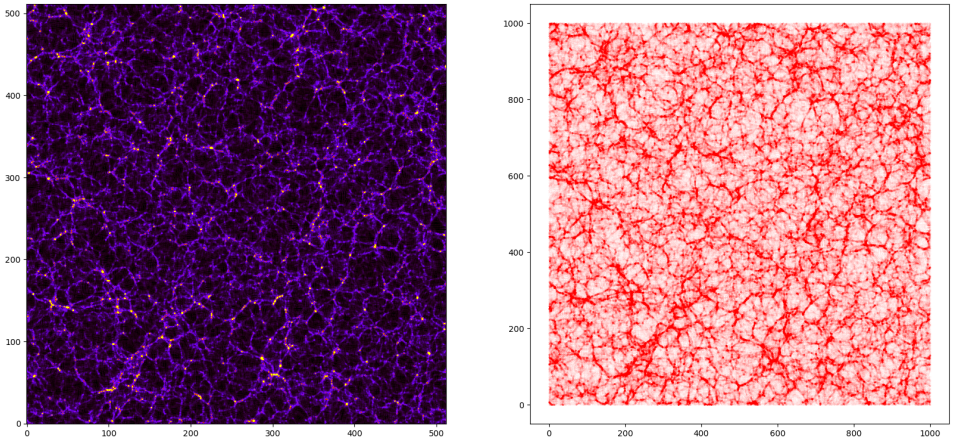
[9]:
```
# the box is 1000 Mpc/h and every voxel has ~2 Mpc/h size. We can take ~5 slices to␣
↪consider a region with a ~10 Mpc/h witdh
mean_density = np.mean(delta[:5,:,:],axis=0) #Take the first 5 component along the first␣
↪axis and compute the mean value
print('Image shape:',mean_density.shape)
print('%.3e < mass < %.3e'%(np.min(mean_density), np.max(mean_density)))
```

(continues on next page)

(continued from previous page)

```
# now lets consider the particles in that slide
indexes = np.where((pos[:,0]<10))
pos_slide = pos[indexes]
print('%.3f < X < %.3f'%(np.min(pos_slide[:,0]), np.max(pos_slide[:,0])))
print('%.3f < Y < %.3f'%(np.min(pos_slide[:,1]), np.max(pos_slide[:,1])))
print('%.3f < Z < %.3f'%(np.min(pos_slide[:,2]), np.max(pos_slide[:,2])))

import matplotlib.pyplot as plt
from pylab import *
fig = figure(figsize=(20,10))
ax1 = fig.add_subplot(121)
ax2 = fig.add_subplot(122)
ax2.set_aspect('equal')
ax1.imshow(mean_density.T, cmap='gnuplot',vmin=0.0, vmax=1e13, origin='lower')
ax2.scatter(pos_slide[:,1], pos_slide[:,2], s=0.001,c='r')
plt.show()
```

```
Image shape: (512, 512)
0.000e+00 < mass < 1.504e+14
0.000 < X < 10.000
0.000 < Y < 999.992
0.000 < Z < 999.992
```

nbsphinx-code-borderwhite 

If needed, the overdensity is easy to calculate

[10]:
```
# at this point, delta contains the effective number of particles in each voxel
# now compute overdensity and density constrast
delta /= np.mean(delta, dtype=np.float64);  delta -= 1.0

print('%.3f < delta < %.3f'%(np.min(delta), np.max(delta)))
print('<delta> = %.3f'%np.mean(delta))
print('shape of the matrix:', delta.shape)
print('density field data type:', delta.dtype)
```

```
-1.000 < delta < 1195.511
<delta> = -0.000
shape of the matrix: (512, 512, 512)
density field data type: float32
```

6.3.1 Lets now compute density fields in redshift-space

Define the value of the parameters

[11]:
```
import redshift_space_library as RSL

snapshot = '/home/jovyan/Data/Snapshots/Mnu_p/0/snapdir_003/snap_003' #location of the␣
↪snapshot
grid     = 512    #the density field will have grid^3 voxels
MAS      = 'CIC'  #Mass-assignment scheme:'NGP', 'CIC', 'TSC', 'PCS'
axis     = 0      #axis along which to move particles to redshift-space (0-X), (1-Y), (2-
↪Z)
verbose  = True   #whether to print information about the progress
```

Lets read the header and the particle positions and masses (for both dark matter and neutrinos)

[12]:
```
# read header
header   = readgadget.header(snapshot)
BoxSize  = header.boxsize/1e3  #Mpc/h
Nall     = header.nall         #Total number of particles
Masses   = header.massarr*1e10 #Masses of the particles in Msun/h
Omega_m  = header.omega_m      #value of Omega_m
Omega_l  = header.omega_l      #value of Omega_l
h        = header.hubble       #value of h
redshift = header.redshift     #redshift of the snapshot
Hubble   = 100.0*np.sqrt(Omega_m*(1.0+redshift)**3+Omega_l)#Value of H(z) in km/s/(Mpc/h)

# read positions and velocities of the particles
pos_c = readgadget.read_block(snapshot, "POS ", [1])/1e3 #positions in Mpc/h
pos_n = readgadget.read_block(snapshot, "POS ", [2])/1e3 #positions in Mpc/h
vel_c = readgadget.read_block(snapshot, "VEL ", [1])     #velocities in km/s
vel_n = readgadget.read_block(snapshot, "VEL ", [2])     #velocities in km/s
```

Print some information about the data read

[13]:
```
print('BoxSize = %.3f Mpc/h'%BoxSize)
print('Rdshift = %.3f'%redshift)
print('Mass DM = %.3e'%Masses[1])
print('Mass NU = %.3e'%Masses[2])
print('%.3f < X_DM < %.3f'%(np.min(pos_c[:,0]), np.max(pos_c[:,0])))
print('%.3f < Y_DM < %.3f'%(np.min(pos_c[:,1]), np.max(pos_c[:,1])))
print('%.3f < Z_DM < %.3f'%(np.min(pos_c[:,2]), np.max(pos_c[:,2])))
print('%.3f < X_NU < %.3f'%(np.min(pos_n[:,0]), np.max(pos_n[:,0])))
print('%.3f < Y_NU < %.3f'%(np.min(pos_n[:,1]), np.max(pos_n[:,1])))
print('%.3f < Z_NU < %.3f'%(np.min(pos_n[:,2]), np.max(pos_n[:,2])))
```

```
BoxSize = 1000.000 Mpc/h
Rdshift = 0.500
Mass DM = 6.516e+11
Mass NU = 4.930e+09
0.000 < X_DM < 999.992
0.000 < Y_DM < 999.992
0.000 < Z_DM < 999.992
0.000 < X_NU < 999.992
```

(continues on next page)

(continued from previous page)

```
0.000 < Y_NU < 999.992
0.000 < Z_NU < 999.992
```

Now lets move particles to redshift space along the X-axis

[14]:
```
# move dark matter particles to redshift-space
RSL.pos_redshift_space(pos_c, vel_c, BoxSize, Hubble, redshift, axis)

# move neutrino particles to redshift-space
RSL.pos_redshift_space(pos_n, vel_n, BoxSize, Hubble, redshift, axis)
```

Now construct the density field of matter (DM+NU)

[15]:
```
# define the matrix holding the density field
delta = np.zeros((grid,grid,grid), dtype=np.float32)

# define two arrays with the masses of the DM and NU particles
mass_c = np.ones(pos_c.shape[0], dtype=np.float32)*Masses[1]
mass_n = np.ones(pos_n.shape[0], dtype=np.float32)*Masses[2]

# construct the density field
MASL.MA(pos_c, delta, BoxSize, MAS, W=mass_c, verbose=verbose)
MASL.MA(pos_n, delta, BoxSize, MAS, W=mass_n, verbose=verbose)
```

```
Using CIC mass assignment scheme with weights
Time taken = 5.882 seconds


Using CIC mass assignment scheme with weights
Time taken = 26.614 seconds
```

Make some checks

[16]:
```
# check the total mass in the density field
Mtot1 = np.sum(delta, dtype=np.float64)
Mtot2 = np.sum(mass_c, dtype=np.float64) + np.sum(mass_n, dtype=np.float64)
print('%.3e should be equal to\n%.3e'%(Mtot1,Mtot2))
```

```
8.812e+19 should be equal to
8.812e+19
```

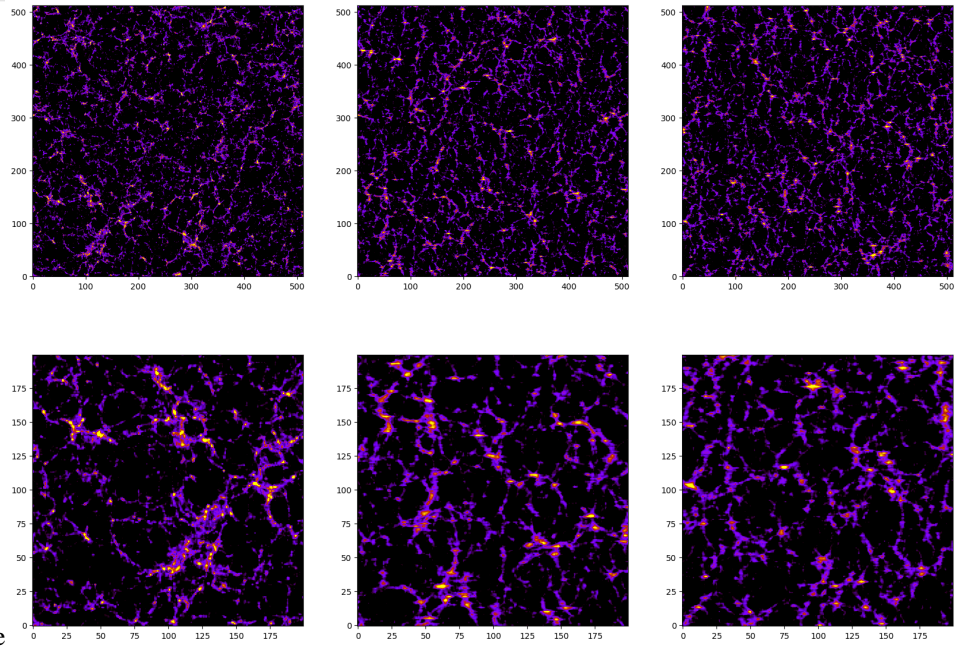
If needed, the overdensity field can be easily computed

[17]:
```
delta /= np.mean(delta, dtype=np.float64);  delta -= 1.0
print('%.3f < delta < %.3f'%(np.min(delta), np.max(delta)))
print('<delta> = %.3f'%np.mean(delta))
```

```
-1.000 < delta < 98.331
<delta> = 0.000
```

Lets plot the density field along three different projections to see the effect of the redshift-space distortions

[18]:
```
df_x = np.mean(delta[:5,:,:]+1,axis=0) #projection into the YZ plane
df_y = np.mean(delta[:,:5,:]+1,axis=1) #projection into the XZ plane
df_z = np.mean(delta[:,:,:5]+1,axis=2) #projection into the XY plane

import matplotlib.pyplot as plt
from pylab import *
fig = figure(figsize=(20,14))
ax1 = fig.add_subplot(231)
ax2 = fig.add_subplot(232)
ax3 = fig.add_subplot(233)
ax4 = fig.add_subplot(234)
ax5 = fig.add_subplot(235)
ax6 = fig.add_subplot(236)
ax1.imshow(df_x.T, cmap='gnuplot',vmin=1, vmax=10, origin='lower')
ax2.imshow(df_y.T, cmap='gnuplot',vmin=1, vmax=10, origin='lower')
ax3.imshow(df_z.T, cmap='gnuplot',vmin=1, vmax=10, origin='lower')
ax4.imshow(df_x[:200,:200].T, cmap='gnuplot',vmin=1, vmax=10, origin='lower')
ax5.imshow(df_y[:200,:200].T, cmap='gnuplot',vmin=1, vmax=10, origin='lower')
ax6.imshow(df_z[:200,:200].T, cmap='gnuplot',vmin=1, vmax=10, origin='lower')
plt.show()
```



nbsphinx-code-borderwhite

As can be seen, the images in the middle and right columns are blurrier than the ones of the left column. This is due to the effects of the redshift-space distortions along that are placed along the X axis.

6.4 Computing power spectra

[1]:
```
import numpy as np
import readgadget
import readfof
import MAS_library as MASL
import Pk_library as PKL
import redshift_space_library as RSL
```

We start by computing the matter power spectrum of a snapshot

[2]:
```
snapshot = '/home/jovyan/Data/Snapshots/s8_p/397/snapdir_004/snap_004' #location of the␣
↪snapshot

# density field parameters
grid    = 512    #the density field will have grid^3 voxels
MAS     = 'CIC'  #Mass-assignment scheme:'NGP', 'CIC', 'TSC', 'PCS'
verbose = True   #whether to print information about the progress

# power spectrum parameters
axis = 0    #axis along which redshift-space distortions have been placed. In real-space␣
↪this parameter doesnt matter
threads = 1 #number of openmp threads to compute the power spectrum
```

First, lets read the particle positions:

[3]:
```
# read the redshift, boxsize, cosmology...etc in the header
header   = readgadget.header(snapshot)
BoxSize  = header.boxsize/1e3  #Mpc/h
Nall     = header.nall         #Total number of particles
Masses   = header.massarr*1e10 #Masses of the particles in Msun/h
Omega_m  = header.omega_m      #value of Omega_m
Omega_l  = header.omega_l      #value of Omega_l
h        = header.hubble       #value of h
redshift = header.redshift     #redshift of the snapshot
Hubble   = 100.0*np.sqrt(Omega_m*(1.0+redshift)**3+Omega_l)#Value of H(z) in km/s/(Mpc/h)

# read positions of the dark matter particles
pos = readgadget.read_block(snapshot, "POS ", [1])/1e3 #positions in Mpc/h
```

Second, lets compute the density field:

[4]:
```
# define the matrix hosting the density field
delta = np.zeros((grid,grid,grid), dtype=np.float32)

# construct 3D density field
MASL.MA(pos, delta, BoxSize, MAS, verbose=verbose)

# compute the overdensity field
delta /= np.mean(delta, dtype=np.float64);  delta -= 1.0
```

(continues on next page)

(continued from previous page)

```
# print some information
print('%.3f < delta < %.3f'%(np.min(delta), np.max(delta)))
print('< delta > = %.3f'%np.mean(delta))
```

```
Using CIC mass assignment scheme
Time taken = 5.516 seconds

-1.000 < delta < 1044.773
< delta > = -0.000
```

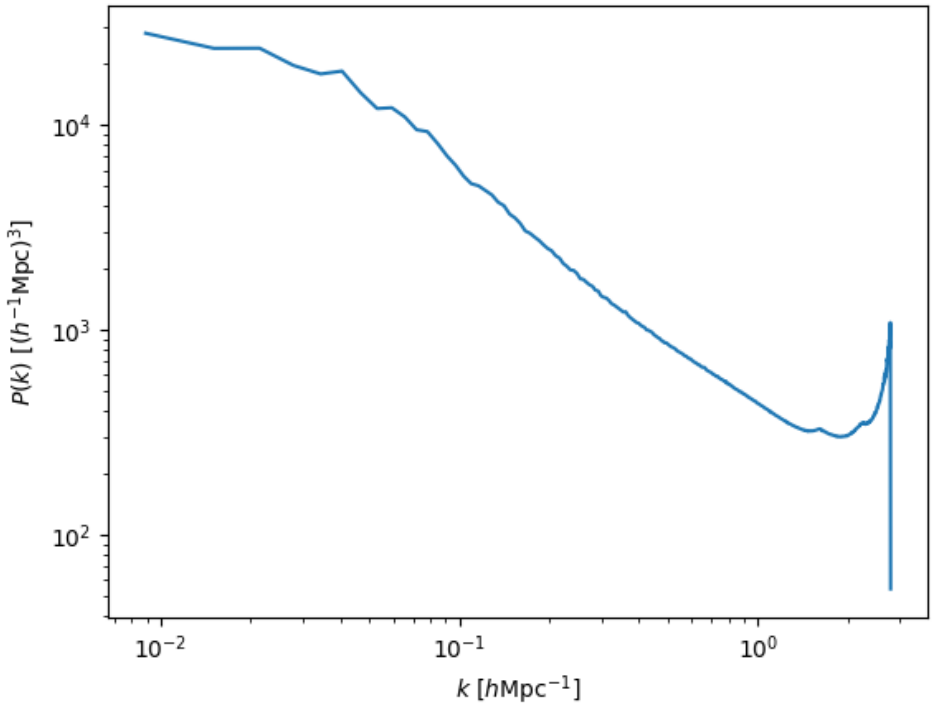
Third, compute the power spectrum

[5]:
```
# compute power spectrum
Pk = PKL.Pk(delta, BoxSize, axis, MAS, threads, verbose)

# Pk is a python class containing the 1D, 2D and 3D power spectra, that can be retrieved␣
↪as

# 1D P(k)
k1D      = Pk.k1D
Pk1D     = Pk.Pk1D
Nmodes1D = Pk.Nmodes1D

# 2D P(k)
kpar     = Pk.kpar
kper     = Pk.kper
Pk2D     = Pk.Pk2D
Nmodes2D = Pk.Nmodes2D

# 3D P(k)
k        = Pk.k3D
Pk0      = Pk.Pk[:,0] #monopole
Pk2      = Pk.Pk[:,1] #quadrupole
Pk4      = Pk.Pk[:,2] #hexadecapole
Pkphase = Pk.Pkphase #power spectrum of the phases
Nmodes   = Pk.Nmodes3D
```

```
Computing power spectrum of the field...
Time to complete loop = 7.10
Time taken = 12.18 seconds
```

Lets see how the 3D matter power spectrum looks like:

[6]:
```
import matplotlib.pyplot as plt
plt.xscale('log')
plt.yscale('log')
plt.xlabel(r'$k~[h{\rm Mpc}^{-1}]$')
plt.ylabel(r'$P(k)~[(h^{-1}{\rm Mpc})^3]$')
plt.plot(k, Pk0)
plt.show()
```



nbsphinx-code-borderwhite

6.4.1 Now lets compute the power spectrum of halos with masses above 1e14 in redshift-space

[7]:
```
snapdir = '/home/jovyan/Data/Halos/FoF/s8_p/397/' #folder hosting the catalogue
snapnum = 4                                       #number of the catalog (4-->z=0, 3-->
↪z=0.5, 2-->z=1, 1-->z=2, 0-->z=3)
```

Lets read the halo catalog

[8]:
```
# read the halo catalogue
FoF = readfof.FoF_catalog(snapdir, snapnum, long_ids=False,
                          swap=False, SFR=False, read_IDs=False)

# get the properties of the halos
pos_h  = FoF.GroupPos/1e3            #Halo positions in Mpc/h
vel_h  = FoF.GroupVel*(1.0+redshift) #Halo peculiar velocities in km/s
mass_h = FoF.GroupMass*1e10          #Halo masses in Msun/h
Np_h   = FoF.GroupLen                #Number of CDM particles in the halo. Even in␣
↪simulations with massive neutrinos, this will be just the number of CDM particles
```

Lets move halos to redshift-space along the z axis:

[9]:
```
# move halos to redshift-space. After this call, pos_h will contain the
# positions of the halos in redshift-space
axis = 2 #axis along which to displace halos
RSL.pos_redshift_space(pos_h, vel_h, BoxSize, Hubble, redshift, axis)
```

Lets now select all halos with masses above 1e14 Msun/h

[10]:
```
indexes = np.where(mass_h>1e14)[0]
pos_h  = pos_h[indexes]
vel_h  = vel_h[indexes]
mass_h = mass_h[indexes]
Np_h   = Np_h[indexes]

print('%.3e < Mass M < %.3e Msun/h'%(np.min(mass_h), np.max(mass_h)))
print('%d halos with masses above 1e14 Msun/h'%pos_h.shape[0])
```

```
1.005e+14 < Mass M < 4.589e+15 Msun/h
39096 halos with masses above 1e14 Msun/h
```

Now lets construct the density field of these halos

[11]:
```
# define the matrix hosting the density field
delta_h = np.zeros((grid,grid,grid), dtype=np.float32)

# construct 3D density field
MASL.MA(pos_h, delta_h, BoxSize, MAS, verbose=verbose)

# compute the overdensity field
delta_h /= np.mean(delta_h, dtype=np.float64);  delta_h -= 1.0

# print some information
print('%.3f < delta < %.3f'%(np.min(delta_h), np.max(delta_h)))
print('< delta > = %.3f'%np.mean(delta_h))
```

```
Using CIC mass assignment scheme
Time taken = 0.248 seconds

-1.000 < delta < 5290.054
< delta > = 0.000
```

We can compute the power spectrum now

[12]:
```
# compute power spectrum
axis = 2 #we have placed the redshift-space distortions along the z-axis for the halos
Pk_h = PKL.Pk(delta_h, BoxSize, axis, MAS, threads, verbose)

# Pk is a python class containing the 1D, 2D and 3D power spectra, that can be retrieved␣
↪as

# 1D P(k)
k1D_h      = Pk_h.k1D
Pk1D_h     = Pk_h.Pk1D
Nmodes1D_h = Pk_h.Nmodes1D

# 2D P(k)
kpar_h     = Pk_h.kpar
kper_h     = Pk_h.kper
Pk2D_h     = Pk_h.Pk2D
Nmodes2D_h = Pk_h.Nmodes2D
```

(continues on next page)

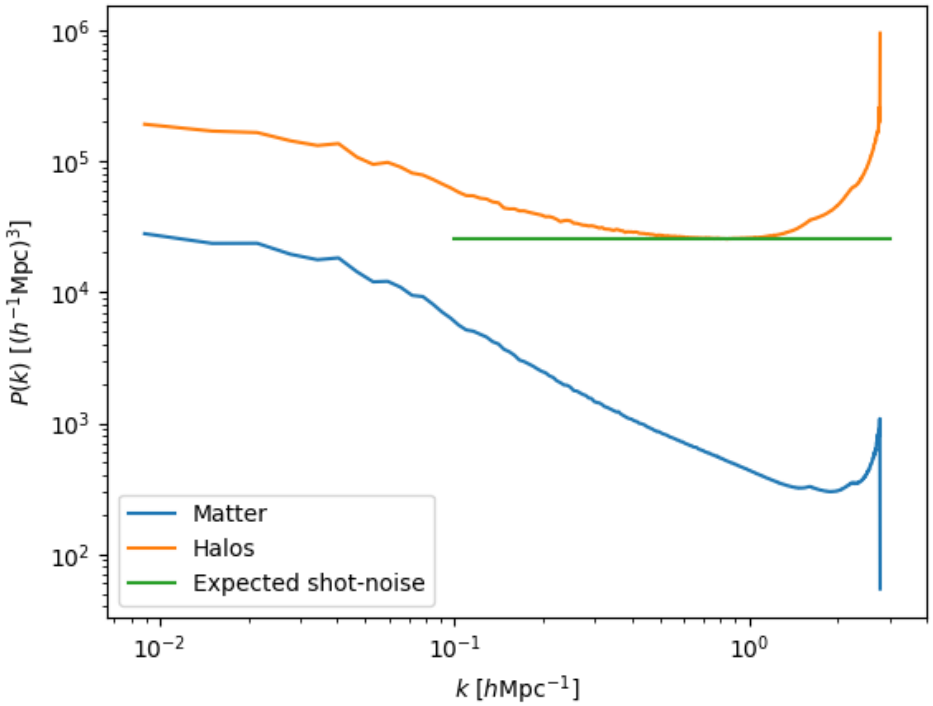
(continued from previous page)

```
# 3D P(k)
k_h       = Pk_h.k3D
Pk0_h     = Pk_h.Pk[:,0] #monopole
Pk2_h     = Pk_h.Pk[:,1] #quadrupole
Pk4_h     = Pk_h.Pk[:,2] #hexadecapole
Pkphase_h = Pk_h.Pkphase #power spectrum of the phases
Nmodes_h  = Pk_h.Nmodes3D
```

```
Computing power spectrum of the field...
Time to complete loop = 7.12
Time taken = 12.30 seconds
```

Lets compare ths power spectrum with the one from matter:

[13]:
```
plt.xscale('log')
plt.yscale('log')
plt.xlabel(r'$k~[h{\rm Mpc}^{-1}]$')
plt.ylabel(r'$P(k)~[(h^{-1}{\rm Mpc})^3]$')
plt.plot(k, Pk0)
plt.plot(k_h, Pk0_h)
plt.plot([1e-1,3],[BoxSize**3/pos_h.shape[0], BoxSize**3/pos_h.shape[0]])
plt.legend(['Matter', 'Halos', 'Expected shot-noise'])
plt.show()
```



nbsphinx-code-borderwhite

As can be seen, halos are more strongly clustered than matter, as expected as we are taking galaxy clusters with masses above 1e14 Msun/h. On small scales, the power spectrum of halos saturates at the expected shot-noise level. On the smallest scales we have, the power spectrum of both halos and matter is affected by aliasing so should not be trusted.

6.4.2 Finally, lets show an example of how to compute marked power spectra

Our goal is to compute the power spectrum of halos but instead of giving each halo the same weight, we want to weight each halo by the overdensity of neutrinos from the cosmic neutrino background.

[14]:
```
snapshot = '/home/jovyan/Data/Snapshots/Mnu_ppp/261/snapdir_004/snap_004' #location of␣
↪the snapshot
snapdir  = '/home/jovyan/Data/Halos/FoF/Mnu_ppp/261/'                     #folder␣
↪hosting the catalogue
snapnum  = 4                                                              #number of the␣
↪catalog (4-->z=0, 3-->z=0.5, 2-->z=1, 1-->z=2, 0-->z=3)

# read header
header   = readgadget.header(snapshot)
BoxSize  = header.boxsize/1e3  #Mpc/h
Nall     = header.nall         #Total number of particles
Masses   = header.massarr*1e10 #Masses of the particles in Msun/h
Omega_m  = header.omega_m      #value of Omega_m
Omega_l  = header.omega_l      #value of Omega_l
h        = header.hubble       #value of h
redshift = header.redshift     #redshift of the snapshot
Hubble   = 100.0*np.sqrt(Omega_m*(1.0+redshift)**3+Omega_l)#Value of H(z) in km/s/(Mpc/h)

# read the neutrino positions
pos_n = readgadget.read_block(snapshot, "POS ", [2])/1e3 #positions in Mpc/h
```

Now, lets compute the density field of neutrinos

[15]:
```
grid = 512

# define the matrix that will contain the neutrino density field
delta_n = np.zeros((grid,grid,grid), dtype=np.float32)

# compute the neutrino density field
MASL.MA(pos_n, delta_n, BoxSize, MAS, verbose=verbose)

# compute the overdensity field
delta_n /= np.mean(delta_n, dtype=np.float64);  delta_n -= 1.0

# print some information
print('%.3f < delta < %.3f'%(np.min(delta_n), np.max(delta_n)))
print('< delta > = %.3f'%np.mean(delta_n))
```

```
Using CIC mass assignment scheme
Time taken = 23.728 seconds

-1.000 < delta < 51.771
< delta > = 0.000
```

Now, lets read the halo catalog

[16]:
```
# read the halo catalogue
FoF = readfof.FoF_catalog(snapdir, snapnum, long_ids=False,
```

(continues on next page)

(continued from previous page)

```
                            swap=False, SFR=False, read_IDs=False)

# get the properties of the halos
pos_h  = FoF.GroupPos/1e3            #Halo positions in Mpc/h
vel_h  = FoF.GroupVel*(1.0+redshift) #Halo peculiar velocities in km/s
mass_h = FoF.GroupMass*1e10          #Halo masses in Msun/h
Np_h   = FoF.GroupLen                #Number of CDM particles in the halo. Even in␣
↪simulations with massive neutrinos, this will be just the number of CDM particles
```

We need to compute the overdensity of neutrinos in the location of the dark matter halos. For this we do

[17]:
```
# definte the array hosting the neutrino overdensities
delta_n_h = np.zeros(pos_h.shape[0], dtype=np.float32)

# interpolate to find the neutrino overdensity in the positions of the halos
MASL.CIC_interp(delta_n, BoxSize, pos_h, delta_n_h)
```

Now, we can construct a density field weigthing each halo by its neutrino overdensity

[18]:
```
# matrix that will host the density field
delta = np.zeros((grid,grid,grid), dtype=np.float32)

# compute the "marked" halo density field
MASL.MA(pos_h, delta, BoxSize, MAS, W=delta_n_h, verbose=verbose)

# print some information about the density field
print('%.3f < delta < %.3f'%(np.min(delta), np.max(delta)))
print('< delta > = %.3f'%np.mean(delta))
```

```
Using CIC mass assignment scheme with weights
Time taken = 0.520 seconds

-0.808 < delta < 21.207
< delta > = 0.002
```

We now compute the power spectrum of this field. This is equivalent to say that we are computing the marked power spectrum of the halos where the mark is the neutrino overdensity

[19]:
```
# compute power spectrum
axis = 0 #we are working in real-space, so this value doesnt matter
Pk_h = PKL.Pk(delta, BoxSize, axis, MAS, threads, verbose)

# Pk is a python class containing the 1D, 2D and 3D power spectra, that can be retrieved␣
↪as

# 1D P(k)
k1D_h      = Pk_h.k1D
Pk1D_h     = Pk_h.Pk1D
Nmodes1D_h = Pk_h.Nmodes1D

# 2D P(k)
kpar_h     = Pk_h.kpar
```

(continues on next page)

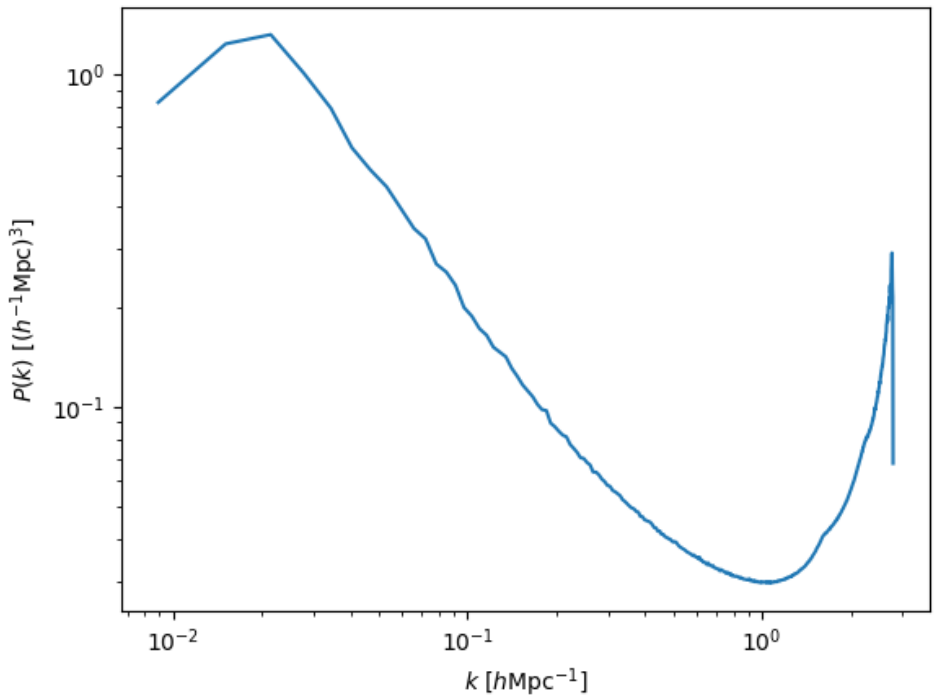
(continued from previous page)

```
kper_h     = Pk_h.kper
Pk2D_h     = Pk_h.Pk2D
Nmodes2D_h = Pk_h.Nmodes2D

# 3D P(k)
k_h       = Pk_h.k3D
Pk0_h     = Pk_h.Pk[:,0] #monopole
Pk2_h     = Pk_h.Pk[:,1] #quadrupole
Pk4_h     = Pk_h.Pk[:,2] #hexadecapole
Pkphase_h = Pk_h.Pkphase #power spectrum of the phases
Nmodes_h  = Pk_h.Nmodes3D
```

```
Computing power spectrum of the field...
Time to complete loop = 7.23
Time taken = 12.47 seconds
```

Now lets see how this marked power spectrum looks like:

[20]:
```
plt.xscale('log')
plt.yscale('log')
plt.xlabel(r'$k~[h{\rm Mpc}^{-1}]$')
plt.ylabel(r'$P(k)~[(h^{-1}{\rm Mpc})^3]$')
plt.plot(k, Pk0_h)
plt.show()
```



nbsphinx-code-borderwhite

Pretty weird, right? :)

[ ]:

CHAPTER

SEVEN

PUBLICATIONS

1. **Deep Learning for Cosmological Parameter Inference from Dark Matter Halo Density Field**
   Zhiwei Min, Xu Xiao, Jiacheng Ding, Liang Xiao, Jie Jiang, Donglin Wu, Qiufan Lin, Yin Li, Yang Wang, Shuai Liu, Zhixin Chen, Xiangru Li, Jinqu Zhang, Le Zhang, Xiao-Dong Li
   2404.09483
2. **SimBIG: Cosmological Constraints using Simulation-Based Inference of Galaxy Clustering with Marked Power Spectra**
   Elena Massara, ChangHoon Hahn, Michael Eickenberg, Shirley Ho, Jiamin Hou, Pablo Lemos, Chirag Modi, Azadeh Moradinezhad Dizgah, Liam Parker, Bruno Régaldo-Saint Blancard
   2404.04228
3. **Neural network reconstruction of density and velocity fields from the 2MASS Redshift Survey**
   Robert Lilow, Punyakoti Ganeshaiah Veena, Adi Nusser
   2404.02278
4. **Constraining Primordial Non-Gaussianity from Large Scale Structure with the Wavelet Scattering Transform**
   Matteo Peron, Gabriel Jung, Michele Liguori, Massimo Pietroni
   2403.17657
5. **Cosmology with Persistent Homology: a Fisher Forecast**
   Jacky H. T. Yip, Matteo Biagetti, Alex Cole, Karthik Viswanathan, Gary Shiu
   2403.13985
6. **Displacement Field Analysis via Optimal Transport: Multi-Tracer Approach to Cosmological Reconstruction**
   Farnik Nikakhtar, Ravi K. Sheth, Nikhil Padmanabhan, Bruno Lévy, Roya Mohayaee
   2403.11951
7. **Quijote-PNG: Optimizing the summary statistics to measure Primordial non-Gaussianity**
   Gabriel Jung, Andrea Ravenni, Michele Liguori, Marco Baldi, William R. Coulton, Francisco Villaescusa-Navarro, Benjamin D. Wandelt
   2403.00490
8. **syren-halofit: A fast, interpretable, high-precision formula for the CDM nonlinear matter power spectrum**
   Deaglan J. Bartlett, Benjamin D. Wandelt, Matteo Zennaro, Pedro G. Ferreira, Harry Desmond
   2402.17492
9. **Cosmology at the Field Level with Probabilistic Machine Learning**
   Adam Rouhiainen
   PhD thesis

10. **LtU-ILI: An All-in-One Framework for Implicit Inference in Astrophysics and Cosmology**
    Matthew Ho, Deaglan J. Bartlett, Nicolas Chartier, Carolina Cuesta-Lazaro, Simon Ding, Axel Lapel, Pablo Lemos, Christopher C. Lovell, T. Lucas Makinen, Chirag Modi, Viraj Pandya, Shivam Pandey, Lucia A. Perez, Benjamin Wandelt, Greg L. Bryan
    2402.05137
11. **SIMBIG: Cosmological Constraints from the Redshift-Space Galaxy Skew Spectra**
    Jiamin Hou, Azadeh Moradinezhad Dizgah, ChangHoon Hahn, Michael Eickenberg, Shirley Ho, Pablo Lemos, Elena Massara, Chirag Modi, Liam Parker, Bruno Régaldo-Saint Blancard
    2401.15074
12. **Bayesian Inference of Initial Conditions from Non-Linear Cosmic Structures using Field-Level Emulators**
    Ludvig Doeser, Drew Jamieson, Stephen Stopyra, Guilhem Lavaux, Florent Leclercq, Jens Jasche
    2312.09271
13. **A point cloud approach to generative modeling for galaxy surveys at the field level**
    Carolina Cuesta-Lazaro, Siddharth Mishra-Sharma
    2311.17141
14. **Constraining Neutrino Cosmologies with Nonlinear Reconstruction**
    Shi-Hui Zang, Hong-Ming Zhu
    2311.16439
15. **Self-calibrating BAO measurements in the presence of Small Displacement Interlopers**
    Alan B. H. Nguyen, Elena Massara, Will J. Percival
    2311.14210
16. **Imprint of massive neutrinos on Persistent Homology of large-scale structure**
    M. H. Jalali Kanafi, S. Ansarifard, S. M. S. Movahed
    2311.13520
17. **Taming assembly bias for primordial non-Gaussianity**
    Emanuele Fondi, Licia Verde, Francisco Villaescusa-Navarro, Marco Baldi, William R. Coulton, Gabriel Jung, Dionysios Karagiannis, Michele Liguori, Andrea Ravenni, Benjamin D. Wandelt
    2311.10088
18. **Analysis of an iterative reconstruction method in comparison of the standard reconstruction method**
    Xinyi Chen, Nikhil Padmanabhan
    2311.09531
19. **Elucidating the impact of massive neutrinos on halo assembly bias**
    Yunjia Song, Ying Zu
    2311.07650
20. **On the range of validity of perturbative models for galaxy clustering and its uncertainty**
    Giosuè Gambardella, Matteo Biagetti, Chiara Moretti, Emiliano Sefusatti
    2311.04608
21. **Evaluating the reconstruction of individual haloes in constrained cosmological simulations**
    Richard Stiskalek, Harry Desmond, Julien Devriendt, Adrianne Slyz
    2310.20672
22. **SimBIG: Field-level Simulation-Based Inference of Galaxy Clustering**
    Pablo Lemos, Liam Parker, ChangHoon Hahn, Shirley Ho, Michael Eickenberg, Jiamin Hou, Elena Massara, Chirag Modi, Azadeh Moradinezhad Dizgah, Bruno Regaldo-Saint Blancard, David Spergel

2310.15256

23. **SIMBIG: Galaxy Clustering Analysis with the Wavelet Scattering Transform**
Bruno Régaldo-Saint Blancard, ChangHoon Hahn, Shirley Ho, Jiamin Hou, Pablo Lemos, Elena Massara, Chirag Modi, Azadeh Moradinezhad Dizgah, Liam Parker, Yuling Yao, Michael Eickenberg
2310.15250

24. **SIMBIG: The First Cosmological Constraints from Non-Gaussian and Non-Linear Galaxy Clustering**
ChangHoon Hahn, Pablo Lemos, Liam Parker, Bruno Régaldo-Saint Blancard, Michael Eickenberg, Shirley Ho, Jiamin Hou, Elena Massara, Chirag Modi, Azadeh Moradinezhad Dizgah, David Spergel
2310.15246

25. **SIMBIG: The First Cosmological Constraints from the Non-Linear Galaxy Bispectrum**
ChangHoon Hahn, Michael Eickenberg, Shirley Ho, Jiamin Hou, Pablo Lemos, Elena Massara, Chirag Modi, Azadeh Moradinezhad Dizgah, Liam Parker, Bruno Régaldo-Saint Blancard
2310.15243

26. **A theoretical view on the T-web statistical description of the cosmic web**
Emma Ayçoberry, Alexandre Barthelemy, Sandrine Codis
2310.03548

27. **Primordial non-Gaussianities with weak lensing: Information on non-linear scales in the Ulagam full-sky simulations**
Dhayaa Anbajagane, Chihway Chang, Hayden Lee, Marco Gatti
2310.02349

28. **Small-scale signatures of primordial non-Gaussianity in k-Nearest Neighbour cumulative distribution functions**
William R. Coulton, Tom Abel, Arka Banerjee
2309.15151

29. **Sensitivity Analysis of Simulation-Based Inference for Galaxy Clustering**
Chirag Modi, Shivam Pandey, Matthew Ho, ChangHoon Hahn, Bruno R'egaldo-Saint Blancard, Benjamin Wandelt
2309.15071

30. **Towards an Optimal Cosmological Detection of Neutrino Mass with Bayesian Inference**
Adrian Bayer
PhD thesis

31. **The effects of non-linearity on the growth rate constraint from velocity correlation functions**
Motonari Tonegawa, Stephen Appleby, Changbom Park, Sungwook E. Hong, Juhan Kim
2309.14457

32. **Hybrid SBI or How I Learned to Stop Worrying and Learn the Likelihood**
Chirag Modi, Oliver H.E. Philcox
2309.10270

33. **Predicting Interloper Fraction with Graph Neural Networks**
Elena Massara, Francisco Villaescusa-Navarro, Will J. Percival
2309.05850

34. **The two-loop power spectrum in redshift space**
Petter Taule, Mathias Garny
2308.07379

35. **Beyond the 3rd moment: A practical study of using lensing convergence CDFs for cosmology with DES Y3**
    D. Anbajagane, C. Chang, A. Banerjee, T. Abel, M. Gatti, V. Ajani, A. Alarcon et al.
    2308.03863

36. **Precision cosmology using voids**
    Alex Woodfinden
    PhD thesis

37. **Probing the anisotropy and non-Gaussianity in redshift space through the derivative of excursion set moments**
    M. H. Jalali Kanafi, S. M. S. Movahed
    2308.03086

38. **Hybrid-bias and displacement emulators for field-level modelling of galaxy clustering in real and redshift space**
    Marcos Pellejero Ibanez, Raul E. Angulo, Drew Jamieson, Yin Li
    2307.09134

39. **Neutrino mass constraint from an Implicit Likelihood Analysis of BOSS voids**
    Leander Thiele, Elena Massara, Alice Pisani, ChangHoon Hahn, David N. Spergel, Shirley Ho, Benjamin Wandelt
    2307.07555

40. **Optimal Transport Reconstruction of Biased Tracers in Redshift Space**
    Farnik Nikakhtar, Nikhil Padmanabhan, Bruno Lévy, Ravi K. Sheth, Roya Mohayaee
    2307.03671

41. **Numerical Studies in Rarefied Gas Dynamics, Cosmological Summary Statistics, and Scalar Field Dark Matter**
    Alvaro Zamora
    PhD thesis

42. **Scattering Spectra Models for Physics**
    Sihao Cheng, Rudy Morel, Erwan Allys, Brice Menard, Stephane Mallat
    2306.17210

43. **Statistical Component Separation for Targeted Signal Recovery in Noisy Mixtures**
    Bruno Regaldo-Saint Blancard, Michael Eickenberg
    2306.15012

44. **Whispers from the Big Bang: cosmological constraints from galaxy power spectra**
    Aaron Glanville
    PhD thesis

45. **Signatures of a Parity-Violating Universe**
    William R. Coulton, Oliver H. E. Philcox, Francisco Villaescusa-Navarro
    2306.11782

46. **Effective cosmic density field reconstruction with convolutional neural network**
    Xinyi Chen, Fangzhou Zhu, Sasha Gaines, Nikhil Padmanabhan
    2306.10538

47. **On approximations of the redshift-space bispectrum and power spectrum multipoles covariance matrix**
    Sergi Novell-Masot, Héctor Gil-Marín, Licia Verde
    2306.03137

48. **Clustering of binary black hole mergers: a detailed analysis of the EAGLE+MOBSE simulation**
Matteo Peron, Sarah Libanore, Andrea Ravenni, Michele Liguori, Maria Celeste Artale
2305.18003

49. **Non-Linearity-Free prediction of the growth-rate f8 using Convolutional Neural Networks**
Koya Murakami, Indira Ocampo, Savvas Nesseris, Atsushi J. Nishizawa, Sachiko Kuroyanagi
2305.12812

50. **Quijote-PNG: The Information Content of the Halo Mass Function**
Gabriel Jung, Andrea Ravenni, Marco Baldi, William R. Coulton, Drew Jamieson, Dionysios Karagiannis, Michele Liguori, Helen Shao, Licia Verde, Francisco Villaescusa-Navarro, Benjamin D. Wandelt
2305.10597

51. **How to estimate Fisher matrices from simulations**
William R. Coulton, Benjamin D. Wandelt
2305.08994

52. **Improving constraints on primordial non-Gaussianity using neural network based reconstruction**
Thomas Flöss, P. Daniel Meerburg
2305.07018

53. **Constraining fNL using the Large-Scale Modulation of Small-Scale Statistics**
Utkarsh Giri, Moritz Münchmeyer, Kendrick M. Smith
2305.03070

54. **Posterior Sampling of the Initial Conditions of the Universe from Non-linear Large Scale Structures using Score-Based Generative Models**
Ronan Legin, Matthew Ho, Pablo Lemos, Laurence Perreault-Levasseur, Shirley Ho, Yashar Hezaveh, Benjamin Wandelt
2304.03788

55. **On the impact of f(Q) gravity on the Large Scale Structure**
Oleksii Sokoliuk, Simran Arora, Subhrat Praharaj, Alexander Baransky, P.K. Sahoo
2303.17341

56. **GEO-FPT: a model of the galaxy bispectrum at mildly non-linear scales**
Sergi Novell-Masot, Davide Gualdi, Héctor Gil-Marín, Licia Verde
2303.15510

57. **Predicting the Initial Conditions of the Universe using Deep Learning**
Vaibhav Jindal, Drew Jamieson, Albert Liang, Aarti Singh, Shirley Ho
2303.13056

58. **Probing massive neutrinos with the Minkowski functionals of the galaxy distribution**
Wei Liu, Aoxiang Jiang, Wenjuan Fang
2302.08162

59. **Cosmological Properties of the Cosmic Web**
Majd Shalak, Jean-Michel Alimi
Phys. Sci. Forum 2023

60. **Perturbation-theory informed integrators for cosmological simulations**
Florian List, Oliver Hahn
2301.09655

61. **Signature of Massive Neutrinos from the Clustering of Critical Points. I. Density-threshold-based Analysis in Configuration Space**
    Jeongin Moon, Graziano Rossi, Hogyun Yu
    ApJS 264 26 (2023)

62. **Constraining cosmological parameters from N-body simulations with Variational Bayesian Neural Networks**
    Héctor J. Hortúa, Luz Ángela García, Leonardo Castañeda C
    2301.03991

63. **Window function convolution with deep neural network models**
    Davit Alkhanishvili, Cristiano Porciani, Emiliano Sefusatti
    2212.09742

64. **Machine learning cosmology from void properties**
    Bonny Y. Wang, Alice Pisani, Francisco Villaescusa-Navarro, Benjamin D. Wandelt
    2212.06860

65. **Cosmology with cosmic web environments II. Redshift-space auto and cross power spectra**
    Tony Bonnaire, Joseph Kuruvilla, Nabila Aghanim, Aurélien Decelle
    2212.06338

66. **Quijote-PNG: Quasi-maximum likelihood estimation of Primordial Non-Gaussianity in the non-linear halo density field**
    Gabriel Jung, Dionysios Karagiannis, Michele Liguori, Marco Baldi, William R Coulton, Drew Jamieson, Licia Verde, Francisco Villaescusa-Navarro, Benjamin D. Wandelt
    2211.07565

67. **SIMBIG: A Forward Modeling Approach To Analyzing Galaxy Clustering**
    ChangHoon Hahn, Michael Eickenberg, Shirley Ho, Jiamin Hou, Pablo Lemos, Elena Massara, Chirag Modi, Azadeh Moradinezhad Dizgah, Bruno Régaldo-Saint Blancard, Muntazir M. Abidi
    2211.00723

68. **SIMBIG: Mock Challenge for a Forward Modeling Approach to Galaxy Clustering**
    ChangHoon Hahn, Michael Eickenberg, Shirley Ho, Jiamin Hou, Pablo Lemos, Elena Massara, Chirag Modi, Azadeh Moradinezhad Dizgah, Bruno Régaldo-Saint Blancard, Muntazir M. Abidi
    2211.00660

69. **Cosmological Information in Skew Spectra of Biased Tracers in Redshift Space**
    Jiamin Hou, Azadeh Moradinezhad Dizgah, ChangHoon Hahn, Elena Massara
    2210.12743

70. **New applications of Graph Neural Networks in Cosmology**
    Farida Farsian, Federico Marulli, Lauro Moscardini, Carlo Giocoli
    2210.11487

71. **Tracer-Field Cross-Correlations with k-Nearest Neighbor Distributions**
    Arka Banerjee, Tom Abel
    2210.05140

72. **Squeezing $f_{\rm NL}$ out of the matter bispectrum with consistency relations**
    Samuel Goldstein, Angelo Esposito, Oliver H. E. Philcox, Lam Hui, J. Colin Hill, Roman Scoccimarro, Maximilian H. Abitbol
    2209.06228

73. **Constraining CDM with density-split clustering**

Enrique Paillas, Carolina Cuesta-Lazaro, Pauline Zarrouk, Yan-Chuan Cai, Will J. Percival, Seshadri Nadathur, Mathilde Pinon, Arnaud de Mattia, Florian Beutler
2209.04310

74. **Bayesian evidence comparison for distance scale estimates**
Aseem Paranjape, Ravi K. Sheth
2209.00668

75. **Minkowski Tensors in Redshift Space – Beyond the Plane Parallel Approximation**
Stephen Appleby, Joby P. Kochappan, Pravabati Chingangbam, Changbom Park
2208.10164

76. **Correcting for small-displacement interlopers in BAO analyses**
Setareh Foroozan, Elena Massara, Will J. Percival
2208.05001

77. **Fast computation of non-linear power spectrum in cosmologies with massive neutrinos**
Hernán E. Noriega, Alejandro Aviles, Sebastien Fromenteau, Mariana Vargas-Magaña
2208.02791

78. **Estimating Cosmological Constraints from Galaxy Cluster Abundance using Simulation-Based Inference**
Moonzarin Reza, Yuanyuan Zhang, Brian Nord, Jason Poh, Aleksandra Ciprijanovic, Louis Strigari
2208.00134

79. **The Cosmic Graph: Optimal Information Extraction from Large-Scale Structure using Catalogues**
T. Lucas Makinen, Tom Charnock, Pablo Lemos, Natalia Porqueres, Alan Heavens, Benjamin D. Wandelt
2207.05202

80. **The Disordered Heterogeneous Universe: Galaxy Distribution and Clustering Across Length Scales**
Oliver H. E. Philcox, Salvatore Torquato
2207.00519

81. **Quijote PNG: The information content of the halo power spectrum and bispectrum**
William R Coulton, Francisco Villaescusa-Navarro, Drew Jamieson, Marco Baldi, Gabriel Jung, Dionysios Karagiannis, Michele Liguori, Licia Verde, Benjamin D. Wandelt
2206.15450

82. **Velocity profiles of matter and biased tracers around voids**
Elena Massara, Will J. Percival, Neal Dalal, Seshadri Nadathur, Slađana Radinović, Hans A. Winther, Alex Woodfinden
2206.14120

83. **Primordial non-Gaussianity and non-Gaussian Covariance**
Thomas Floss, Matteo Biagetti, P. Daniel Meerburg
2206.10458

84. **Field Level Neural Network Emulator for Cosmological N-body Simulations**
Drew Jamieson, Yin Li, Renan Alves de Oliveira, Francisco Villaescusa-Navarro, Shirley Ho, David N. Spergel
2206.04594

85. **Simple lessons from complex learning: what a neural network model learns about cosmic structure formation**
Drew Jamieson, Yin Li, Siyu He, Francisco Villaescusa-Navarro, Shirley Ho, Renan Alves de Oliveira, David N. Spergel
2206.04573

86. **Cosmological Information in the Marked Power Spectrum of the Galaxy Field**

Elena Massara, Francisco Villaescusa-Navarro, ChangHoon Hahn, Muntazir M. Abidi, Michael Eickenberg, Shirley Ho, Pablo Lemos, Azadeh Moradinezhad Dizgah, Bruno Regaldo-Saint Blancard
2206.01709

87. **Quijote-PNG: Quasi-maximum likelihood estimation of Primordial Non-Gaussianity in the non-linear dark matter density field**
Gabriel Jung, Dionysios Karagiannis, Michele Liguori, Marco Baldi, William R Coulton, Drew Jamieson, Licia Verde, Francisco Villaescusa-Navarro, Benjamin D. Wandelt
2206.01624

88. **Quijote-PNG: Simulations of primordial non-Gaussianity and the information content of the matter field power spectrum and bispectrum**
William R Coulton, Francisco Villaescusa-Navarro, Drew Jamieson, Marco Baldi, Gabriel Jung, Dionysios Karagiannis, Michele Liguori, Licia Verde, Benjamin D. Wandelt
2206.01619

89. **Accurate predictions from small boxes: variance suppression via the Zel'dovich approximation**
Nickolas Kokron, Shi-Fan Chen, Martin White, Joseph DeRose, Mark Maus
2205.15327

90. **Robust Neural Network-Enhanced Estimation of Local Primordial Non-Gaussianity**
Utkarsh Giri, Moritz Münchmeyer, Kendrick M. Smith
2205.12964

91. **Two-loop power spectrum with full time- and scale-dependence and EFT corrections: impact of massive neutrinos and going beyond EdS**
Mathias Garny, Petter Taule
2205.11533

92. **Improving cosmological covariance matrices with machine learning**
Natali S.M. de Santi, L. Raul Abramo
2205.10881

93. **Fast and realistic large-scale structure from machine-learning-augmented random field simulations**
Davide Piras, Benjamin Joachimi, Francisco Villaescusa-Navarro
2205.07898

94. **Distinguishing Dirac vs. Majorana Neutrinos: a Cosmological Probe**
Beatriz Hernandez-Molinero, Raul Jimenez, Carlos Pena-Garay
2205.00808

95. **Accurate Model of the Projected Velocity Distribution of Galaxies in Dark Matter Halos**
Han Aung, Daisuke Nagai, Eduardo Rozo, Brandon Wolfe, Susmita Adhikari
2204.13131

96. **Wavelet Moments for Cosmological Parameter Estimation**
Michael Eickenberg, Erwan Allys, Azadeh Moradinezhad Dizgah, Pablo Lemos, Elena Massara, Muntazir Abidi, ChangHoon Hahn, Sultan Hassan, Bruno Regaldo-Saint Blancard, Shirley Ho, Stephane Mallat, Joakim Andén, Francisco Villaescusa-Navarro
2204.07646

97. **Quantification of high dimensional non-Gaussianities and its implication to Fisher analysis in cosmology**
Core Francisco Park, Erwan Allys, Francisco Villaescusa-Navarro, Douglas P. Finkbeiner
2204.05435

98. **Bayesian Control Variates for optimal covariance estimation with pairs of simulations and surrogates**

Nicolas Chartier, Benjamin D. Wandelt
2204.03070

99. **Probing massive neutrinos with the Minkowski functionals of large-scale structure**
Wei Liu, Aoxiang Jiang, Wenjuan Fang
2204.02945

100. **Perturbation Theory vs Simulation: Quasi-linear Scale, Binning Effect, and Visualization of Bispectrum**
Joseph Tomlinson, Donghui Jeong
2204.00668

101. **The effect of local universe constraints on halo abundance and clustering**
Maxwell L. Hutt, Harry Desmond, Julien Devriendt, Adrianne Slyz
2203.14724

102. **Extracting high-order cosmological information in galaxy surveys with power spectra**
Yuting Wang, Gong-Bo Zhao, Kazuya Koyama, Will J. Percival, Ryuichi Takahashi, Chiaki Hikage, Héctor Gil-Marín, ChangHoon Hahn, Ruiyang Zhao, Weibing Zhang, Xiaoyong Mu, Yu Yu, Hong-Ming Zhu, Fei Ge
2202.05248

103. **Constraining cosmological parameters from N-body simulations with Bayesian Neural Networks**
Hector J. Hortua
2112.11865

104. **Detection of spatial clustering in the 1000 richest SDSS DR8 redMaPPer clusters with Nearest Neighbor distributions**
Yunchong Wang, Arka Banerjee, Tom Abel
2112.04502

105. **One-point statistics matter in extended cosmologies**
Alex Gough, Cora Uhlemann
2112.04428

106. **Cosmology with cosmic web environments I. Real-space power spectra**
Tony Bonnaire, Nabila Aghanim, Joseph Kuruvilla, Aurélien Decelle
2112.03926

107. **The Information Content of Projected Galaxy Fields**
Lucas Porth, Gary M. Bernstein, Robert E. Smith, Abigail J. Lee
2111.13702

108. **Cosmology and neutrino mass with the Minimum Spanning Tree**
Krishna Naidoo, Elena Massara, Ofer Lahav
2111.12088

109. **The Covariance of Squeezed Bispectrum Configurations**
Matteo Biagetti, Lina Castiblanco, Jorge Noreña, Emiliano Sefusatti
2111.05887

110. **NECOLA: Towards a Universal Field-level Cosmological Emulator**
Neerav Kaushal, Francisco Villaescusa-Navarro, Elena Giusarma, Yin Li, Conner Hawry, Mauricio Reyes
2111.02441

111. **The smearing scale in Laguerre reconstructions of the correlation function**
Farnik Nikakhtar, Ravi K. Sheth, Idit Zehavi
2110.03591

112. **Cosmology with the kinetic Sunyaev-Zeldovich effect: Independent of the optical depth and σ_8**
Joseph Kuruvilla
2109.13938

113. **Creating Jackknife and Bootstrap estimates of the covariance matrix for the two-point correlation function**
Faizan G. Mohammad, Will J. Percival
2109.07071

114. **The matter density PDF for modified gravity and dark energy with Large Deviations Theory**
Matteo Cataneo, Cora Uhlemann, Christian Arnold, Alex Gough, Baojiu Li, Catherine Heymans
2109.02636

115. **Towards an Optimal Estimation of Cosmological Parameters with the Wavelet Scattering Transform**
Georgios Valogiannis, Cora Dvorkin
2108.07821

116. **Beware of Fake νs : The Effect of Massive Neutrinos on the Non-Linear Evolution of Cosmic Structure**
Adrian E. Bayer, Arka Banerjee, Uros Seljak
2108.04215

117. **The effects of peculiar velocities on the morphological properties of large scale structures**
Aoxiang Jiang, Wei Liu, Wenjuan Fang, Wen Zhao
2108.03851

118. **Analytic Gaussian Covariance Matrices for Galaxy N-Point Correlation Functions**
Jiamin Hou, Robert N. Cahn, Oliver H.E. Philcox, Zachary Slepian
2108.01714

119. **Modeling Nearest Neighbor distributions of biased tracers using Hybrid Effective Field Theory**
Arka Banerjee, Nickolas Kokron, Tom Abel
2107.10287

120. **The reach of next-to-leading-order perturbation theory for the matter bispectrum**
Davit Alkhanishvili, Cristiano Porciani, Emiliano Sefusatti, Matteo Biagetti, Andrei Lazanu, Andrea Oddo, and Victoria Yankelevich
2107.08054

121. **The GIGANTES dataset: precision cosmology from voids in the machine learning era**
Christina D. Kreisch, Alice Pisani, Francisco Villaescusa-Navarro, David N. Spergel, Benjamin D. Wandelt, Nico Hamaus, Adrian E. Bayer
2107.02304

122. **The PDF perspective on the tracer-matter connection: Lagrangian bias and non-Poissonian shot noise**
Oliver Friedrich, Anik Halder, Aoife Boyle, Cora Uhlemann, Dylan Britt, Sandrine Codis, Daniel Gruen, ChangHoon Hahn
2107.02300

123. **Clustering in Massive Neutrino Cosmologies via Eulerian Perturbation Theory**
Alejandro Aviles, Arka Banerjee, Gustavo Niz, Zachary Slepian
2106.13771

124. **CARPool Covariance: Fast, unbiased covariance estimation for large-scale structure observables**
Nicolas Chartier, Benjamin D. Wandelt
2106.11718

125. **Extracting cosmological parameters from N-body simulations using machine learning techniques**
Andrei Lazanu
2106.11061

126. **Unsupervised Resource Allocation with Graph Neural Networks**
Miles Cranmer, Peter Melchior, Brian Nord
2106.09761

127. **Normalizing flows for random fields in cosmology**
Adam Rouhiainen, Utkarsh Giri, Moritz Münchmeyer
2105.12024

128. **Joint analysis of anisotropic power spectrum, bispectrum and trispectrum: application to N-body simulations**
Davide Gualdi, Hector Gil-Marin, Licia Verde
2104.03976

129. **Clustering and halo abundances in early dark energy cosmological models**
Anatoly Klypin, Vivian Poulin, Francisco Prada, Joel Primack, Marc Kamionkowski, Vladimir Avila-Reese, Aldo Rodriguez-Puebla, Peter Behroozi, Doug Hellinger, Tristan L Smith
MNRAS article

130. **Detecting the radiative decay of the cosmic neutrino background with line-intensity mapping**
Jose Luis Bernal, Andrea Caputo, Francisco Villaescusa-Navarro, Marc Kamionkowski
2103.12099

131. **Information content in mean pairwise velocity and mean relative velocity between pairs in a triplet**
Joseph Kuruvilla, Nabila Aghanim
2102.06709

132. **Detecting neutrino mass by combining matter clustering, halos, and voids**
Adrian E. Bayer, Francisco Villaescusa-Navarro, Elena Massara, Jia Liu, David N. Spergel, Licia Verde, Benjamin Wandelt, Matteo Viel, Shirley Ho
2102.05049

133. **Information Content of Higher-Order Galaxy Correlation Functions**
Lado Samushia, Zachary Slepian, Francisco Villaescusa-Navarro
2102.01696

134. **Cosmological cross-correlations and nearest neighbor distributions**
Arka Banerjee, Tom Abel
2102.01184

135. **Learning the Evolution of the Universe in N-body Simulations**
Chang Chen, Yin Li, Francisco Villaescusa-Navarro, Shirley Ho, Anthony Pullen
2012.05472

136. **Constraining M_ν with the Bispectrum II: The Total Information Content of the Galaxy Bispectrum**
ChangHoon Hahn, Francisco Villaescusa-Navarro
2012.02200

137. **Fast and Accurate Non-Linear Predictions of Universes with Deep Learning**
Renan Alves de Oliveira, Yin Li, Francisco Villaescusa-Navarro, Shirley Ho, David N. Spergel
2012.00240

138. **Minkowski functionals and the nonlinear perturbation theory in the large-scale structure: second-order effects**
Takahiko Matsubara, Chiaki Hikage, Satoshi Kuriki
2012.00203

139. **The unequal-time matter power spectrum: impact on weak lensing observables**
Lucia F. de la Bella, Nicolas Tessore, Sarah Bridle
2011.06185

140. **Exploring KSZ velocity reconstruction with N-body simulations and the halo model**
Utkarsh Giri, Kendrick M. Smith
2010.07193

141. **Modeling the Marked Spectrum of Matter and Biased Tracers in Real- and Redshift-Space**
Oliver H.E. Philcox, Alejandro Aviles, Elena Massara
2010.05914

142. **CARPool: fast, accurate computation of large-scale structure statistics by pairing costly and cheap cosmological simulations**
Nicolas Chartier, Benjamin Wandelt, Yashar Akrami, Francisco Villaescusa-Navarro
2009.08970

143. **Matter trispectrum: theoretical modelling and comparison to N-body simulations**
Davide Gualdi, Sergi Novell, Héctor Gil-Marín, Licia Verde
2009.02290

144. **The impact of massive neutrinos on halo assembly bias**
Titouan Lazeyras, Francisco Villaescusa-Navarro, Matteo Viel
2008.12265

145. **Capturing the Cosmic Web for Cosmology**
Krishna Naidoo
1829731

146. **Nearest Neighbor distributions: new statistical measures for cosmological clustering**
Arka Banerjee, Tom Abel
2007.13342

147. **The effects of massive neutrinos on the linear point of the correlation function**
G. Parimbelli, S. Anselmi, M. Viel, C. Carbone, F. Villaescusa-Navarro, P.S. Corasaniti, Y. Rasera, R. Sheth, G.D. Starkman, I. Zehavi
2007.10345

148. **A Lagrangian Perturbation Theory in the presence of massive neutrinos**
Alejandro Aviles, Arka Banerjee
2007.06508

149. **Discovering Symbolic Models from Deep Learning with Inductive Biases**
Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, Shirley Ho
2006.11287

150. **What does the marked power spectrum measure? Insights from perturbation theory**
Oliver H.E. Philcox, Elena Massara, David N. Spergel
2006.10055

151. **New Interpretable Statistics for Large Scale Structure Analysis and Generation**

E. Allys, T. Marchand, J.-F. Cardoso, F. Villaescusa-Navarro, S. Ho, S. Mallat
2006.06298

152. **A Faster Fourier Transform? Computing Small-Scale Power Spectra and Bispectra for Cosmological Simulations in $\mathcal{O}(N^2)$ Time**
Oliver H.E. Philcox
2005.01739

153. **Effective halo model: Creating a physical and accurate model of the matter power spectrum and cluster counts**
Oliver H.E. Philcox, David N. Spergel, Francisco Villaescusa-Navarro
2004.09515

154. **What Can We Learn by Combining the Skew Spectrum and the Power Spectrum?**
Ji-Ping Dai, Licia Verde, Jun-Qing Xia
2002.09904

155. **Using the Marked Power Spectrum to Detect the Signature of Neutrinos in Large-Scale Structure**
Elena Massara, Francisco Villaescusa-Navarro, Shirley Ho, Neal Dalal, David N. Spergel
2001.11024

156. **Super-resolution emulator of cosmological simulations using deep physical models**
Doogesh Kodi Ramanah, Tom Charnock, Francisco Villaescusa-Navarro, Benjamin D. Wandelt
2001.05519

157. **Primordial non-Gaussianity without tails – how to measure fNL with the bulk of the density PDF**
Oliver Friedrich, Cora Uhlemann, Francisco Villaescusa-Navarro, Tobias Baldauf, Marc Manera, Takahiro Nishimichi
1912.06621

158. **Fisher for complements: Extracting cosmology and neutrino mass from the counts-in-cells PDF**
Cora Uhlemann, Oliver Friedrich, Francisco Villaescusa-Navarro, Arka Banerjee, Sandrine Codis
1911.11158

159. **Learning neutrino effects in Cosmology with Convolutional Neural Networks**
Elena Giusarma, Mauricio Reyes Hurtado, Francisco Villaescusa-Navarro, Siyu He, Shirley Ho, ChangHoon Hahn
1910.04255

160. **Constraining M_ν with the bispectrum. Part I. Breaking parameter degeneracies**
ChangHoon Hahn, Francisco Villaescusa-Navarro, Emanuele Castorina, Roman Scoccimarro
1909.11107

161. **Weighing neutrinos with the halo environment**
Arka Banerjee, Emanuele Castorina, Francisco Villaescusa-Navarro, Travis Court, Matteo Viel
1907.06598

162. **Anisotropic halo assembly bias and redshift-space distortions**
Andrej Obuljen, Neal Dalal, Will J. Percival
1906.11823

163. **The Quijote simulations**
Francisco Villaescusa-Navarro, ChangHoon Hahn, Elena Massara, Arka Banerjee, Ana Maria Delgado, Doogesh Kodi Ramanah, Tom Charnock, Elena Giusarma, Yin Li, Erwan Allys, Antoine Brochard, Cora Uhlemann, Chi-Ting Chiang, Siyu He, Alice Pisani, Andrej Obuljen, Yu Feng, Emanuele Castorina, Gabriella

Contardo, Christina D. Kreisch, Andrina Nicola, Justin Alsing, Roman Scoccimarro, Licia Verde, Matteo Viel, Shirley Ho, Stephane Mallat, Benjamin Wandelt, David N. Spergel
1909.05273

CHAPTER

EIGHT

TEAM

- Francisco Villaescusa-Navarro (Flatiron/Princeton)
- ChangHoon Hahn (Berkeley)
- Elena Massara (Flatiron/Waterloo)
- Arka Banerjee (Stanford)
- Ana Maria Delgado (CUNY/Flatiron)
- Doogesh K. Ramanah (Sorbonne, Paris)
- Tom Charnock (Sorbonne, Paris)
- Elena Giusarma (Flatiron/MTech)
- Yin Li (Berkeley/IPMU)
- Erwan Allys (ENS, Paris)
- Antoine Brochard (ENS, Paris)
- Cora Uhlemann (Cambridge)
- Chi-Ting Chiang (BNL)
- Siyu He (Flatiron)
- Alice Pisani (Princeton)
- Andrej Obuljen (Waterloo)
- Yu Feng (Berkeley)
- Emanuele Castorina (Berkeley)
- Gabriella Contardo (Flatiron)
- Christina D. Kreisch (Princeton)
- Andrina Nicola (Princeton)
- Justin Alsing (Oskar Klein/Flatiron)
- William Coulton (Flatiron)
- Drew Jamieson (MPA)
- Gabriel Jung (Padova, Italy)
- Dionysios Karagiannis (Cape Town, South Africa)
- Michele Liguori (Padova, Italy)

- Marco Baldi (Bologna)
- Oliver Philcox (Columbia/Simons)
- Roman Scoccimarro (NYU)
- Licia Verde (Barcelona)
- Matteo Viel (SISSA)
- Shirley Ho (Flatiron/Princeton)
- Stephane Mallat (ENS/College de France)
- Ben Wandelt (IAP, Paris)
- David Spergel (Flatiron/Princeton)

CHAPTER

NINE

CITATION

If you have used data from the Quijote simulations you may consider citing the Quijote paper.

```
@ARTICLE{Quijote_sims,
         author = {{Villaescusa-Navarro}, Francisco and {Hahn}, ChangHoon and {Massara},␣
↪Elena and {Banerjee}, Arka and {Delgado}, Ana Maria and {Ramanah}, Doogesh Kodi and
↪{Charnock}, Tom and {Giusarma}, Elena and {Li}, Yin and {Allys}, Erwan and {Brochard},␣
↪Antoine and {Uhlemann}, Cora and {Chiang}, Chi-Ting and {He}, Siyu and {Pisani}, Alice␣
↪and {Obuljen}, Andrej and {Feng}, Yu and {Castorina}, Emanuele and {Contardo},␣
↪Gabriella and {Kreisch}, Christina D. and {Nicola}, Andrina and {Alsing}, Justin and
↪{Scoccimarro}, Roman and {Verde}, Licia and {Viel}, Matteo and {Ho}, Shirley and
↪{Mallat}, Stephane and {Wandelt}, Benjamin and {Spergel}, David N.},
         title = "{The Quijote Simulations}",
         journal = {\apjs},
         keywords = {N-body simulations, Cosmological parameters, Astrostatistics, Large-
↪scale structure of the universe, Cosmological neutrinos, 1083, 339, 1882, 902, 338,␣
↪Astrophysics - Cosmology and Nongalactic Astrophysics, Astrophysics - Instrumentation␣
↪and Methods for Astrophysics},
         year = 2020,
         month = sep,
         volume = {250},
         number = {1},
         eid = {2},
         pages = {2},
         doi = {10.3847/1538-4365/ab9d82},
         archivePrefix = {arXiv},
         eprint = {1909.05273},
         primaryClass = {astro-ph.CO},
         adsurl = {https://ui.adsabs.harvard.edu/abs/2020ApJS..250....2V},
         adsnote = {Provided by the SAO/NASA Astrophysics Data System}
}
```

If you use data from Molino, consider citing the Molino paper

```
@ARTICLE{Molino,
    author = {{Hahn}, ChangHoon and {Villaescusa-Navarro}, Francisco},
    title = "{Constraining M$_{{\ensuremath{\nu}}}$ with the bispectrum. Part II. The␣
↪information content of the galaxy bispectrum monopole}",
    journal = {\jcap},
    keywords = {cosmological parameters from LSS, cosmological simulations, neutrino␣
↪masses from cosmology, redshift surveys, Astrophysics - Cosmology and Nongalactic␣
↪Astrophysics},
```

(continues on next page)

(continued from previous page)

```
    year = 2021,
    month = apr,
    volume = {2021},
    number = {4},
    eid = {029},
    pages = {029},
    doi = {10.1088/1475-7516/2021/04/029},
    archivePrefix = {arXiv},
    eprint = {2012.02200},
    primaryClass = {astro-ph.CO},
    adsurl = {https://ui.adsabs.harvard.edu/abs/2021JCAP...04..029H},
    adsnote = {Provided by the SAO/NASA Astrophysics Data System}
}
```

If you use data from Gigantes, consider citing the Gigantes paper

```
@ARTICLE{Gigantes,
    author = {{Kreisch}, Christina D. and {Pisani}, Alice and {Villaescusa-Navarro},␣
↪Francisco and {Spergel}, David N. and {Wandelt}, Benjamin D. and {Hamaus}, Nico and
↪{Bayer}, Adrian E.},
    title = "{The GIGANTES dataset: precision cosmology from voids in the machine␣
↪learning era}",
    journal = {arXiv e-prints},
    keywords = {Astrophysics - Cosmology and Nongalactic Astrophysics, Astrophysics -␣
↪Instrumentation and Methods for Astrophysics},
    year = 2021,
    month = jul,
    eid = {arXiv:2107.02304},
    pages = {arXiv:2107.02304},
    archivePrefix = {arXiv},
    eprint = {2107.02304},
    primaryClass = {astro-ph.CO},
    adsurl = {https://ui.adsabs.harvard.edu/abs/2021arXiv210702304K},
    adsnote = {Provided by the SAO/NASA Astrophysics Data System}
}
```

If you use data from Quijote-PNG (see *Primordial non-Gaussianities*), consider citing the Quijote-PNG papers: 2206.01619 and 2206.01624.

```
@ARTICLE{Quijote-PNG,
    author = {{Coulton}, William R and {Villaescusa-Navarro}, Francisco and {Jamieson},␣
↪Drew and {Baldi}, Marco and {Jung}, Gabriel and {Karagiannis}, Dionysios and {Liguori},
↪ Michele and {Verde}, Licia and {Wandelt}, Benjamin D.},
    title = "{Quijote-PNG: Simulations of primordial non-Gaussianity and the information␣
↪content of the matter field power spectrum and bispectrum}",
    journal = {arXiv e-prints},
    keywords = {Astrophysics - Cosmology and Nongalactic Astrophysics},
    year = 2022,
    month = jun,
    eid = {arXiv:2206.01619},
    pages = {arXiv:2206.01619},
    archivePrefix = {arXiv},
```

(continues on next page)

(continued from previous page)

```
    eprint = {2206.01619},
    primaryClass = {astro-ph.CO},
    adsurl = {https://ui.adsabs.harvard.edu/abs/2022arXiv220601619C},
    adsnote = {Provided by the SAO/NASA Astrophysics Data System}
}

@ARTICLE{2022ApJ...940...71J,
   author = {{Jung}, Gabriel and {Karagiannis}, Dionysios and {Liguori}, Michele and
↪{Baldi}, Marco and {Coulton}, William R. and {Jamieson}, Drew and {Verde}, Licia and
↪{Villaescusa-Navarro}, Francisco and {Wandelt}, Benjamin D.},
   title = "{Quijote-PNG: Quasi-maximum Likelihood Estimation of Primordial Non-
↪Gaussianity in the Nonlinear Dark Matter Density Field}",
   journal = {\apj},
   keywords = {Non-Gaussianity, Cosmological parameters from large-scale structure,␣
↪Fisher's Information, 1116, 340, 1922, Astrophysics - Cosmology and Nongalactic␣
↪Astrophysics},
   year = 2022,
   month = nov,
   volume = {940},
   number = {1},
   eid = {71},
   pages = {71},
   doi = {10.3847/1538-4357/ac9837},
   archivePrefix = {arXiv},
   eprint = {2206.01624},
   primaryClass = {astro-ph.CO},
   adsurl = {https://ui.adsabs.harvard.edu/abs/2022ApJ...940...71J},
   adsnote = {Provided by the SAO/NASA Astrophysics Data System}
}
```

If you use data from Quijote-ODD (see *Parity-violation*), consider citing the Quijote-ODD paper:

```
@ARTICLE{Quijote-ODD,
    author = {{Coulton}, William R. and {Philcox}, Oliver H.~E. and {Villaescusa-Navarro}
↪, Francisco},
    title = "{Signatures of a Parity-Violating Universe}",
    journal = {arXiv e-prints},
    keywords = {Astrophysics - Cosmology and Nongalactic Astrophysics, Astrophysics -␣
↪Astrophysics of Galaxies, General Relativity and Quantum Cosmology, High Energy␣
↪Physics - Phenomenology, High Energy Physics - Theory},
    year = 2023,
    month = jun,
    eid = {arXiv:2306.11782},
    pages = {arXiv:2306.11782},
    doi = {10.48550/arXiv.2306.11782},
    archivePrefix = {arXiv},
    eprint = {2306.11782},
    primaryClass = {astro-ph.CO},
    adsurl = {https://ui.adsabs.harvard.edu/abs/2023arXiv230611782C},
    adsnote = {Provided by the SAO/NASA Astrophysics Data System}
}
```

CHAPTER

TEN

STRUCTURE AND TYPES

We now describe the way the Quijote simulations are organized and the different cosmological models present on it.

10.1 Classes and types

The Quijote simulations can be classified into three broad **classes**:

- **Fiducial simulations.** Those are simulations with a fiducial cosmology consistent with Planck. They only vary the initial random seed.
- **Individual parameter variations.** Those are simulations where the value of a single parameter is varied with respect to the fiducial simulations. The initial random seed of those simulations are taken to match those of the fiducial model. Those are designed for Fisher matrix calculations.
- **Multiple parameter variations.** Those are simulations that vary simultaneously the value of several parameters and the initial random seed. Those simulations are designed for machine learning applications.

The Quijote simulations can also be classified into different **types**:

- **LCDM**. Standard simulations with different values of $\Omega_{\rm m}$, $\Omega_{\rm b}$, h, n_s, σ_8. See *LCDM* for more details.
- **Dark Energy**. These are simulations were we vary the expansion rate of the Universe through the w parameter. See *Dark energy* for more details.
- **Massive neutrinos**. These are simulations that include massive neutrinos as additional particles. See *Massive neutrinos* for more details.
- **Separate Universe**. These simulations incorporate and overall over(under)density or an amplitude of the DC mode different to zero. See *Separate Universe* for more details.
- **Primordial non-Gaussianities**. These simulations include primordial non-Gaussianities of different types. See *Primordial non-Gaussianities* for more details.
- **Parity violating**. These simulations include parity violating features. See *Parity-violation* for more details.
- **Modified gravity**. These simulations are run with a modified gravity model: $f(R)$. See *Modified Gravity* for more details.

Note that these types are not exclusive, i.e. there are simulations that vary LCDM parameters plus the dark energy parameter plus the neutrino masses. The scheme bell shows the different classes of simulations in Quijote:

Fiducial model (17,100 simulations)

$\Omega_m = 0.3175, \Omega_b = 0.049, h = 0.6711, n_s = 0.9624, \sigma_8 = 0.834,$
$w = -1, M_\nu = 0, \delta_b = 0, f_{NL} = 0, p_{NL} = 0, f(R) = 0$

Individual parameter variations (23,000 simulations)

ΛCDM	Dark energy	Massive neutrinos	Separate Universe	Primordial non-Gaussianities	Parity violation	Modified gravity
$\Omega_m, \Omega_b, h, n_s, \sigma_8$	w	M_ν	δ_b	$f_{\rm NL}^{\rm local}, f_{\rm NL}^{\rm equil}, f_{\rm NL}^{\rm ortho}$	$p_{\rm NL}$	$f(R)$

Multiple parameter variations (42,816 simulations)

3 latin-hypercubes
$\Omega_m, \Omega_b, h, n_s, \sigma_8$
6,000 simulations

1 latin-hypercube
$\Omega_m, \Omega_b, h, n_s, \sigma_8, M_\nu, w$
2,000 simulations

Big Sobol Sequence
$\Omega_m, \Omega_b, h, n_s, \sigma_8$
32,768 simulations

SB7
$\Omega_m, \Omega_b, h, n_s, \sigma_8, M_\nu, f_R$
2,048 simulations

A brief description of the different cosmologies is provided in the below table. The standard and paired fixed snapshots or data products will be located inside the same folder. The paired fixed (or fixed) will be located inside folders starting with NCV (from No Cosmic Variance). Further details can be found in the Quijote paper and Quijote-PNG paper.

Name	Ω_m	Ω_b	h	n_s	σ_8	M_ν	w	δ_b	$f_{\rm NL}^{\rm loc}$
fiducial	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
fiducial	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
fiducial_ZA	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
fiducial_LR	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
fiducial_HR	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
Om_p	0.3275	0.049	0.6711	0.9624	0.834	0	-1	0	0
Om_p	0.3275	0.049	0.6711	0.9624	0.834	0	-1	0	0
Om_m	0.3075	0.049	0.6711	0.9624	0.834	0	-1	0	0
Om_m	0.3075	0.049	0.6711	0.9624	0.834	0	-1	0	0
Ob2_p	0.3175	0.051	0.6711	0.9624	0.834	0	-1	0	0
Ob2_p	0.3175	0.051	0.6711	0.9624	0.834	0	-1	0	0
Ob2_m	0.3175	0.047	0.6711	0.9624	0.834	0	-1	0	0
Ob2_m	0.3175	0.047	0.6711	0.9624	0.834	0	-1	0	0
Ob_p	0.3175	0.050	0.6711	0.9624	0.834	0	-1	0	0
Ob_m	0.3175	0.048	0.6711	0.9624	0.834	0	-1	0	0
h_p	0.3175	0.049	0.6911	0.9624	0.834	0	-1	0	0
h_p	0.3175	0.049	0.6911	0.9624	0.834	0	-1	0	0
h_m	0.3175	0.049	0.6511	0.9624	0.834	0	-1	0	0
h_m	0.3175	0.049	0.6511	0.9624	0.834	0	-1	0	0
ns_p	0.3175	0.049	0.6711	0.9824	0.834	0	-1	0	0
ns_p	0.3175	0.049	0.6711	0.9824	0.834	0	-1	0	0
ns_m	0.3175	0.049	0.6711	0.9424	0.834	0	-1	0	0

Table 1 – continued from previou

Name	Ω_m	Ω_b	h	n_s	σ_8	M_ν	w	δ_b	$f_{\rm NL}^{\rm loc}$
ns_m	0.3175	0.049	0.6711	0.9424	0.834	0	-1	0	0
s8_p	0.3175	0.049	0.6711	0.9624	0.849	0	-1	0	0
s8_p	0.3175	0.049	0.6711	0.9624	0.849	0	-1	0	0
s8_m	0.3175	0.049	0.6711	0.9624	0.819	0	-1	0	0
s8_m	0.3175	0.049	0.6711	0.9624	0.819	0	-1	0	0
Mnu_p	0.3175	0.049	0.6711	0.9624	0.834	0.1	-1	0	0
Mnu_p	0.3175	0.049	0.6711	0.9624	0.834	0.1	-1	0	0
Mnu_pp	0.3175	0.049	0.6711	0.9624	0.834	0.2	-1	0	0
Mnu_pp	0.3175	0.049	0.6711	0.9624	0.834	0.2	-1	0	0
Mnu_ppp	0.3175	0.049	0.6711	0.9624	0.834	0.4	-1	0	0
Mnu_ppp	0.3175	0.049	0.6711	0.9624	0.834	0.4	-1	0	0
w_p	0.3175	0.049	0.6711	0.9624	0.834	0	-1.05	0	0
w_m	0.3175	0.049	0.6711	0.9624	0.834	0	-0.95	0	0
DC_p	0.3175	0.049	0.6711	0.9624	0.834	0	-1	+0.035	0
DC_m	0.3175	0.049	0.6711	0.9624	0.834	0	-1	-0.035	0
LC_p	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	+100
LC_m	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	-100
EQ_p	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
EQ_m	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
OR_CMB_p	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
OR_CMB_m	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
OR_LSS_p	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
OR_LSS_m	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
ODD_p	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
ODD_m	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
fR_p	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
fR_pp	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
fR_ppp	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
fR_pppp	0.3175	0.049	0.6711	0.9624	0.834	0	-1	0	0
latin_hypercube	[0.1 - 0.5]	[0.03 - 0.07]	[0.5 - 0.9]	[0.8 - 1.2]	[0.6 - 1.0]	0	-1	0	0
latin_hypercube	[0.1 - 0.5]	[0.03 - 0.07]	[0.5 - 0.9]	[0.8 - 1.2]	[0.6 - 1.0]	0	-1	0	0
latin_hypercube	[0.1 - 0.5]	[0.03 - 0.07]	[0.5 - 0.9]	[0.8 - 1.2]	[0.6 - 1.0]	0	-1	0	0
nwLH	[0.1 - 0.5]	[0.03 - 0.07]	[0.5 - 0.9]	[0.8 - 1.2]	[0.6 - 1.0]	[0.01 - 1.0]	[-1.3 - -0.7]	0	0
SB7	[0.1 - 0.5]	[0.03 - 0.07]	[0.5 - 0.9]	[0.8 - 1.2]	[0.6 - 1.0]	[0.01 - 1.0]	-1	0	0
BSQ	[0.1 - 0.5]	[0.02 - 0.08]	[0.5 - 0.9]	[0.8 - 1.2]	[0.6 - 1.0]	0	-1	0	0

- Simulations with $\delta_b \neq 0$ correspond to separate universe simulations and therefore have an amplitude of the DC mode different than 0 (or equivalently, a curvature different than 0). See *Separate Universe* for further details on these simulatons.
- Simulations with $f_{\rm NL} \neq 0$ correspond to simulations with primordial non-Gaussianities (Quijote-PNG). See *Primordial non-Gaussianities* for further details on these simulations.
- Simulations with $p_{\rm NL} \neq 0$ correspond to simulations with parity-violating initial conditions (Qujjote-ODD). See *Parity-violation* for further details on these simulations.
- Simulations with $f_{R_0} \neq 0$ correspond to simulations with modified gravity (Quijote-MG). See *Modified Gravity* for further details on these simulations.
- Simulations with parameters in brackets correspond to the latin-hypercubes and sobol sequences simulations. See *Latin-hypercubes* and *Big Sobol Sequence* for further details on these simulations.

CHAPTER

ELEVEN

LCDM

Quijote contains standard N-body simulations varying the five vanilla ΛCDM parameters: $\Omega_{\rm m}$, $\Omega_{\rm b}$, h, n_s, σ_8. In these simulations $w = -1$, $M_\nu = 0$ eV, $\Omega_K = 0$ and the initial conditions are generated with 2LPT.

These simulations include `Om_p`, `Om_m`, `Ob_p`, `Ob_m`, `h_p`, `h_m`, `ns_p`, `ns_m`, `s8_p`, `s8_m`, `Ob2_p`, `Ob2_m`, `fidcial`, `fiducial_HR`, `fiducial_LR`, `fiducial_ZA`, and the three standard latin-hypercubes.

Note: The initial conditions of the `fiducial_ZA` simulations have been generated with the Zel'dovich approximation and not 2LPT. This is because these simulations are designed to be used with other Zel'dovich generated ICs simulation such as `Mnu_p`, `Mnu_pp`, and `Mnu_ppp`.

These simulations are designed to explore and quantify the impact of vanilla cosmological parameters of the spatial distribution of matter, halos, and galaxies.

CHAPTER

TWELVE

DARK ENERGY

Quijote contains simulations where the dark energy equation of state is $w =\neq -1$. These are standard N-body simulations run with a different Hubble function, $H(z)$, that contains the changes introduced in the evolution of the background by the dark energy equation of state.

The initial conditions of these simulations are generated using the Zel'dovich approximation and the inital matter power spectrum and $H(z)$ function is computed using reps. The simulations `w_p` and `w_m` (designed to compute partial derivatives for Fisher matrix calculations) together with the `nwLH` latin-hypercube are examples of simulations where $w \neq -1$.

These simulations are designed to explore and quantify the impact of the dark energy equation of state on the large-scale structure of the Universe.

CHAPTER

THIRTEEN

MASSIVE NEUTRINOS

Quijote include N-body simulations that model massive neutrinos. In these simulations, neutrinos as modelled as a separate cold and pressureless fluid represented by neutrino particles. The main difference between these particles and those of dark matter is that neutrino particles have thermal velocities that are draw in the initial conditions from the underlying Fermi-Dirac distribution.

The initial conditions of these simulations are generated using the Zel'dovich approximation taking into account the scale-dependent growth factor and growth rate induced by neutrinos. For details on this we refer the reader 1605.05283.

Currently, the simulations including massive neutrinos are `Mnu_p`, `Mnu_pp`, `Mnu_ppp` (designed to compute derivatives for Fisher matrix calculations), and the `nwLH` latin-hypercube (designed for machine learning applications). See *Structure and types* for further details.

These simulations are designed to explore and quantify the impact of massive neutrinos on the different elements of the cosmic web.

CHAPTER

FOURTEEN

SEPARATE UNIVERSE

In standard N-body simulations, the mean matter matter density in the box is $\Omega_{\rm m}\rho_{\rm crit}(1+z)^3$, where $\rho_{\rm crit}(z)$ is the critical density at redshift 0. In other words, the mean matter overdensity (with respect to the global one), δ_b, is zero. However, in the real Universe, regions of finite volume will exhibit fluctuations around $\delta_b = 0$ due to perturbation on scales larger than the considered regions. Separate Universe simulations will follow the evolution of dark matter particles under the influence of an overdensity different to zero; or equivalently under the impact of a fluctuation that is larger than the size of the box. These simulations will thus have one extra parameter, δ_b that represents the mean overdensity over the entire box.

The way to incorporate the global overdensity is to change the cosmology of it, introducing curvature. Thus, in these simulations $\Omega_K \neq 1$. Currently, the only Quijote simulations with $\delta_b \neq 0$ are `DC_p` and `DC_m` that are designed to compute partial derivatives to quantify supersample covariance effects. See section 2.3 of the Quijote paper.

These simulation are designed to explore and quatify the impact of super-sample covariance on cosmological ohservables. Many thanks to Yin Li for setting up the initial conditions and cosmology of these simulations.

CHAPTER

FIFTEEN

PRIMORDIAL NON-GAUSSIANITIES

Quijote contains 4,000 N-body simulations with primordial non-Gaussianities: **Quijote-PNG**. All these simulations contain 512^3 dark matter particles in a periodic volume of $(1\ h^{-1}\mathrm{Gpc})^3$ and share the same cosmology as the fiducial model: $\Omega_\mathrm{m} = 0.3175$, $\Omega_\mathrm{b} = 0.049$, $h = 0.6711$, $n_s = 0.9624$, $\sigma_8 = 0.834$, $w = -1$, $M_\nu = 0.0$ eV. These are standard N-body simulations run with initial conditions generated in a particular way.

The video below shows an example of two N-body simulations with Gaussian initial conditions (left) and local primordial non-Gaussianities initial conditions (right). As can be seen, differences are very small even for a value as large as $f_\mathrm{NL} = 200$ as the one we use.

The simulations in Quijote-PNG can be classified into four different sets: 1) local, 2) equilateral, 3) orthogonal CMB, and 4) orthogonal LSS (see *Bispectrum shapes*). Each set contains 1,000 simulations: 500 with $f_\mathrm{NL} = +100$ and 500 with $f_\mathrm{NL} = -100$. Quijote-PNG is thus organized into eight different folders, depending on the non-Gaussianity shape and the value of f_NL:

- **LC_p**: contains data from 500 simulations with local type and $f_\mathrm{NL} = +100$
- **LC_m**: contains data from 500 simulations with local type and $f_\mathrm{NL} = -100$
- **EQ_p**: contains data from 500 simulations with equilateral type and $f_\mathrm{NL} = +100$
- **EQ_m**: contains data from 500 simulations with equilateral type and $f_\mathrm{NL} = -100$
- **OR_CMB_p**: contains data from 500 simulations with orthogonal CMB type and $f_\mathrm{NL} = +100$
- **OR_CMB_m**: contains data from 500 simulations with orthogonal CMB type and $f_\mathrm{NL} = -100$
- **OR_LSS_p**: contains data from 500 simulations with orthogonal LSS type and $f_\mathrm{NL} = +100$
- **OR_LSS_m**: contains data from 500 simulations with orthogonal LSS type and $f_\mathrm{NL} = -100$

Each of the above folders contains 500 sub-folders, each of them hosting the result of a different simulation. For instance, the folder `EQ_p/72/` contains the results of the 72th simulation run with $f_\mathrm{NL} = +100$ for the equilateral shape. Depending on the location, these folder will contain the snapshots, halo catalogues, or other data products.

15.1 Bispectrum shapes

In Quijote-PNG we only consider models that have a primordial bispectrum, defined as

$$\langle \Phi(\mathbf{k}_1)\Phi(\mathbf{k}_2)\Phi(\mathbf{k}_3)\rangle = (2\pi)^3\delta^{(3)}(\mathbf{k}_1+\mathbf{k}_2+\mathbf{k}_3)B_\Phi(k_1,k_2,k_3)\ ,$$

where $\Phi(\mathbf{k})$ is the primordial potential. We consider four different shapes for the primordial bispectrum:

1) **Local**. The local shape can be characterized by

$$B_\Phi^\mathrm{local}(k_1,k_2,k_3) = 2f_\mathrm{NL}^\mathrm{local}P_\Phi(k_1)P_\Phi(k_2) + \text{ 2 perm.}$$

2) **Equilateral**. The equilaterial shape is described by

$$B_\Phi^{\rm equil.}(k_1,k_2,k_3) = 6f_{\rm NL}^{\rm equil.}\Big[-P_\Phi(k_1)P_\Phi(k_2) + \text{ 2 perm.}$$
$$-2\left(P_\Phi(k_1)P_\Phi(k_2)P_\Phi(k_3)\right)^{\frac{2}{3}} + P_\Phi(k_1)^{\frac{1}{3}}P_\Phi(k_2)^{\frac{2}{3}}P_\Phi(k_3) + \text{5 perm.}\Big]$$

3) **Orthogonal CMB**. The orthogonal CMB template is given by

$$B_\Phi^{\rm ortho-CMB}(k_1,k_2,k_3) = 6f_{\rm NL}^{\rm ortho-CMB}\Big[-3P_\Phi(k_1)P_\Phi(k_2)$$
$$+\text{ 2 perm.} - 8\left(P_\Phi(k_1)P_\Phi(k_2)P_\Phi(k_3)\right)^{\frac{2}{3}} + 3P_\Phi(k_1)^{\frac{1}{3}}P_\Phi(k_2)^{\frac{2}{3}}P_\Phi(k_3) + \text{5 perm.}\Big]$$

4) **Orthogonal LSS**. The orthogonal LSS template is given by

$$B_\Phi^{\rm ortho-LSS}(k_1,k_2,k_3) =$$
$$6f_{\rm NL}^{\rm ortho-CMB}\left(P_\Phi(k_1)P_\Phi(k_2)P_\Phi(k_3)\right)^{\frac{2}{3}}\Bigg[$$
$$-\left(1+\frac{9p}{27}\right)\frac{k_3^2}{k_1k_2} + \text{2 perms} + \left(1+\frac{15p}{27}\right)\frac{k_1}{k_3}$$
$$+\text{5 perms} - \left(2+\frac{60p}{27}\right)$$
$$+\frac{p}{27}\frac{k_1^4}{k_2^2k_3^2} + \text{2 perms} - \frac{20p}{27}\frac{k_1k_2}{k_3^2} + \text{2 perms}$$
$$-\frac{6p}{27}\frac{k_1^3}{k_2k_3^2} + \text{5 perms} + \frac{15p}{27}\frac{k_1^2}{k_3^2} + \text{5 perms}\Bigg]$$

15.2 Initial conditions

The initial conditions of the Quijote-PNG simulations have been generated using a modified version of the code described in Scoccimarro et al. 2012. Our modified version of the code is publicly available here.

The initial conditions of a given simulation can be found in a folder called `ICs`, that contains:

- `ics.X`. These are the initial conditions that contain the particle positions, velocities, and IDs. These are Gadget format-II snapshots and can be read as described in *Snapshots*. `X` can go from 0 to 127.
- `2LPT.params`. This is the parameter file used to generate the initial conditions.
- `logIC`. The output of the initial conditions generator code.

The value of initial random seed for the simulation i is $10 \times i + 5$ (this can be found in the `2LPT.params` file) independently of the shape and $f_{\rm NL}$ value. For instance, the value of the initial random seed for `OR_CMB_p/100` and `OR_CMB_m/100` is 1005. This choice enables the calculation of partial derivatives, needed for Fisher matrix calculations.

For the details about the linear matter power spectrum used for these simulations see *Linear power spectra*.

15.3 Snapshots

We keep snapshots at redshifts 0, 0.5, 1, 2, and 3. The snapshots are saved as HDF5 files, and they can be read in the standard way (see *Snapshots* for details on this).

15.4 Halo catalogues

We store Friends-of-Friends (FoF) halo catalogues for each snapshot of each simulation in Quijote-PNG. We refer the user to *Halo catalogs* for details on how to read these files.

15.5 Density fields

To facilitate the post-processing of the data we also provide 3D grids containing the overdensity, $\delta(x) = \rho(x)/\bar{\rho} - 1$, for each redshift of all PNG simulations. We refer the user to *Density fields* for details on how to read these files.

15.6 Team

Quijote-PNG was developed in 2022 by:

- William Coulton (CCA, USA)
- Gabriel Jung (Padova, Italy)
- Francisco Villaescusa-Navarro (CCA/Princeton, USA)
- Dionysios Karagiannis (Cape Town, South Africa)
- Drew Jamieson (MPA, Germany)
- Michele Liguori (Padova, Italy)
- Marco Baldi (Bologna, Italy)
- Licia Verde (Barcelona, Spain)
- Benjamin Wandelt (IAP, France)

CHAPTER

SIXTEEN

PARITY-VIOLATION

Quijote contains N-body simulations whose intial conditions were generated with parity-violation properties: Quijote-ODD. All these simulations contain 512^3 dark matter particles in a periodic volume of $(1\ h^{-1}\mathrm{Gpc})^3$ and share the same cosmology as the fiducial model: $\Omega_\mathrm{m} = 0.3175$, $\Omega_\mathrm{b} = 0.049$, $h = 0.6711$, $n_s = 0.9624$, $\sigma_8 = 0.834$, $w = -1$, $M_\nu = 0.0$ eV. As the Quijote-PNG simulations (see *Primordial non-Gaussianities*), these are standard N-body simulations run with initial conditions generated in a particular way.

The video below shows an example of two N-body simulations with Gaussian initial conditions (top-left) and parity-violating initial conditions (bottom-left). The parity-violating simulation has been flipped along the x-axis to mimick the effect of a mirror in the center. The panels on the right show the differences between both.

Currently, the simulations in Quijote-ODD can be classified into two sets depending on the sign of p_NL:

- **ODD_p**: 500 simulations with $p_\mathrm{NL} = +10^6$.
- **ODD_m**: 500 simulations with $p_\mathrm{NL} = -10^6$.

The simulations in the above sets can be directly compared (the share the same underlying Gaussian density field) with the first 500 simulations of the fiducial model.

16.1 Initial conditions

The initial conditions of the Quijote-ODD simulations have been generated using a modified version of the code described in Scoccimarro et al. 2012. Our modified version of the code is publicly available here.

The initial conditions of a given simulation can be found in a folder called `ICs`, that contains:

- `ics.X`. These are the initial conditions that contain the particle positions, velocities, and IDs. These are Gadget format-II snapshots and can be read as described in *Snapshots*. `X` can go from 0 to 127.
- `2LPT.params`. This is the parameter file used to generate the initial conditions.
- `logIC`. The output of the initial conditions generator code.

The value of initial random seed for the simulation i is $10 \times i + 5$ (this can be found in the `2LPT.params` file) independently of the shape and f_NL value. For instance, the value of the initial random seed for `ODD_p/100` and `ODD_m/100` is 1005. This choice enables the calculation of partial derivatives, needed for Fisher matrix calculations.

16.2 Snapshots

We keep snapshots at redshifts 0, and 1. The snapshots are saved as compressed HDF5 files, and they can be read in the standard way (see *Snapshots* for details on this).

16.3 Halo catalogs

We store both Friends-of-Friends (FoF) and Rockstar halo catalogs for each snapshot of each simulation in Quijote-ODD. We refer the user to *Halo catalogs* for details on how to read the FoF files. The Rockstar catalogs are ASCII files and the header contains information about the structure of the data.

16.4 Density fields

To facilitate the post-processing of the data we also provide 3D grids containing the overdensity, $\delta(x) = \rho(x)/\bar{\rho} - 1$, for each redshift of all PNG simulations. We refer the user to *Density fields* for details on how to read these files.

16.5 Team

Quijote-ODD was developed in 2023 by:

- William Coulton (CCA, USA)
- Oliver Philcox (Columbia/Simons, USA)
- Francisco Villaescusa-Navarro (Simons/Princeton, USA)

CHAPTER

SEVENTEEN

MODIFIED GRAVITY

Quijote contains N-body simulations with modified gravity: **Quijote-MG**. The movie below shows one of these simulations together wits ΛCDM counterpart:

If you are interested in using these simulations, please contact us at marco.baldi5@unibo.it or villaescusa.francisco@gmail.com.

17.1 General description

Quijote-MG contains 4,048 N-body simulations run with MG-Gadget and using the Hu & Sawicki f(R) model as the modified gravity model. Each simulation follows the evolution of 512^3 dark matter plus 512^3 neutrinos in a periodic cosmological volume of $(1000\ \mathrm{Mpc}/h)^3$. The initial conditions have been generated using the Zel'dovich approximation at $z = 127$ and the simulations have been run with the appropiate Hubble function $H(z)$. We have saved 5 snapshots, at redshifts 0, 0.5, 1, 2, and 3. For each simulation we have saved FoF catalogs, Rockstar catalogs, and different power spectra (see below).

The simulations can be classified into two different groups:

- Simulations designed for Fisher matrix calculations
- Simulations designed for machine learning calculations

17.1.1 Simulations for Fisher matrix

For the first category we have 2,000 simulations. In this category there are four different types:

- 500 simulations run with $f_{R_0} = -5 \times 10^{-7}$
- 500 simulations run with $f_{R_0} = -5 \times 10^{-6}$
- 500 simulations run with $f_{R_0} = -5 \times 10^{-5}$
- 500 simulations run with $f_{R_0} = -5 \times 10^{-4}$

Note: We refer the reader to *Structure and types* for details on the value of the cosmological parameters, the initial conditions...etc.

These simulations are designed for Fisher matrix calculations, and therefore, they have matching IDs between themselves and among other Quijote simulations. We note that to compute generic partial derivatives:

$$\frac{\partial \vec{S}}{\partial f_R}$$

where $\vec{S}$ is a generic summary statistics and f_R is the modified gravity parameter, we can use methods like this:

$$\frac{\partial \vec{S}}{\partial f_R} \simeq \frac{\vec{S}(f_R+\delta f_R)-\vec{S}(f_R)}{\delta f_R}$$

$$\frac{\partial \vec{S}}{\partial f_R} \simeq \frac{-3\vec{S}(f_R)+4\vec{S}(f_R+\delta f_R)-\vec{S}(f_R+2\delta f_R)}{2\delta f_R}$$

$$\frac{\partial \vec{S}}{\partial f_R} \simeq \frac{-21\vec{S}(f_R)+32\vec{S}(f_R+\delta f_R)-12\vec{S}(f_R+2\delta f_R)+\vec{S}(4\delta f_R)}{12\delta f_R}$$

$$\frac{\partial \vec{S}}{\partial f_R} \simeq \frac{-315\vec{S}(f_R)+512\vec{S}(f_R+\delta f_R)-224\vec{S}(f_R+2\delta f_R)+28\vec{S}(4\delta f_R)-\vec{S}(8\delta f_R)}{168\delta f_R}$$

where the fiducial value of f_R is set to zero.

Important: Note that the chosen values of f_{R_0} are not distributed equally in both linear and log considering that the fiducial value is $f_{R_0}=0$. Thus, when performing Fisher matrix calculations, we recommend perform the following change of variables: $Y=(f_{R_0})^{\log_{10}(2)}$. In that way, the values of f_{R_0} equal to 0, -5×10^{-7}, -5×10^{-6}, -5×10^{-5}, -5×10^{-4}, map to Y equal to 0, -0.0127, -0.0254, -0.0507, -0.101, and the above formulae can easily be used to evaluate $\partial\vec{S}/\partial Y$.

17.1.2 Simulations for machine learning

In this category we have 2,048 simulations. Each simulation has a different value of the initial random seed and of the parameters $\Omega_{\rm m}$, $\Omega_{\rm b}$, h, n_s, σ_8, M_ν, f_{R0}. The value of those parameters in the simulations are organized in a Sobol sequence with boundaries:

$$0.1 \leq \Omega_{\rm m} \leq 0.5$$
$$0.03 \leq \Omega_{\rm b} \leq 0.07$$
$$0.5 \leq h \leq 0.9$$
$$0.8 \leq n_s \leq 1.2$$
$$0.6 \leq \sigma_8 \leq 1.0$$
$$0.01 \leq M_\nu[{\rm eV}] \leq 1.0$$
$$-3\times10^{-4} \leq f_{R0} \leq 0$$

Note: The actual value of these parameters for the different simulations can be found here.

17.2 Organization

The data is split into different folders:

- `Snapshots`. This folder contains 2,048 subfolders, one for each simulation. Inside these subfolders, the user can find the initial conditions, snapshots, simulation parameters, and additional files produced by MG-Gadget.
- `Halos`. This folder contains 2 folders: `FoF` and `Rockstar`. Each of those folders contains 2,048 folders, inside which the halo catalogs at different redshifts are located.
- `Pk`. This folder contains 2,048 subfolders, one for each simulation. Inside these subfolders, the user can find the different power spectra.

17.3 Snapshots

Every simulation contains 5 snapshots. Each snapshot is stored in a folder called `snapdir_00X`, where `X=0` is $z = 3$, `X=1` is $z = 2$, `X=2` is $z = 1$, `X=3` is $z = 0.5$, `X=4` is $z = 0$. The snapshots are stored in hdf5 format, and can be read using Pylians (see details in *Snapshots*). Note that the snapshots have been compressed to save space, so please take a look at *FAQ* if you encounter problems reading them.

Note: The initial conditions are located inside a folder called `ICs`. The initial conditions are also stored as hdf5 files, and can be read in the same way as the simulation snapshots.

The MG-Gadget snapshots contains more blocks than traditional Gadget N-body simulations. The fields stored in the snapshots are:

```
/CompressionInfo
/Header
/PartType1
/PartType1/Acceleration
/PartType1/Coordinates
/PartType1/ModifiedGravityAcceleration Dataset
/PartType1/ModifiedGravityGradPhi Dataset
/PartType1/ModifiedGravityPhi Dataset
/PartType1/ParticleIDs
/PartType1/Velocities
/PartType2
/PartType2/Acceleration
/PartType2/Coordinates
/PartType2/ModifiedGravityAcceleration Dataset
/PartType2/ModifiedGravityGradPhi Dataset
/PartType2/ModifiedGravityPhi Dataset
/PartType2/ParticleIDs
/PartType2/Velocities
```

where `PartType1` represent cold dark matter and `PartType2` correspond to neutrinos.

17.4 Halo catalogs

Quijote-MG contains both FoF and Rockstar halo catalogs for every snapshot of each simulation. You can find details about how to read these files in *Halo catalogs*.

17.5 Power spectra

For every snapshot of each Quijote-MG simulation we have computed the following power spectra:

- cold dark matter auto-Pk in real-space: `Pk_CDM_z=X.XXX.dat`
- cold dark matter auto-Pk in redshift-space: `Pk_CDM_RS_axis=Y_z=X.XXX.dat`
- neutrino auto-Pk in real-space: `Pk_NU_z=X.XXX.dat`
- neutrino auto-Pk in redshift-space: `Pk_NU_RS_axis=Y_z=X.XXX.dat`
- total matter auto-Pk in real-space: `Pk_CDM+NU_z=X.XXX.dat`
- total matter auto-Pk in redshift-space: `Pk_CDM+NU_RS_axis=Y_z=X.XXX.dat`
- CDM-neutrino cross-Pk in real-space: `Pk_CDMNU_z=X.XXX.dat`
- CDM-neutrino cross-Pk in redshift-space: `Pk_CDMNU_RS_axis=Y_z=X.XXX.dat`

Where `X.XXX` is the redshift and `Y` (0, 1, or 2) is the axis along which the redshift-space distortions have been placed.

17.6 Bispectra

For every snapshot of each Quijote-MG simulation we have computed the full matter bispectrum. We use a grid with 384^3 voxels and we measure the bispectrum in more than 7,000 different triangle configurations. The name of the files is `Bk_m_z=X.X.txt`, where `X.X` represents the redshift.

CHAPTER

EIGHTEEN

LATIN-HYPERCUBES

Quijote provides several latin-hypercubes that can be classified into two main categories depending on whether they include massive neutrinos:

18.1 LH

The simulations in this category only consider massless neutrinos. There are three latin-hypercubes in this category, each containing 2,000 simulations that vary the value of $\Omega_{\rm m}$, $\Omega_{\rm b}$, h, n_s, σ_8. The limits of the latin-hypercubes are set by:

$$\begin{aligned}\Omega_{\rm m} &\in [0.1; 0.5]\\ \Omega_{\rm b} &\in [0.03; 0.07]\\ h &\in [0.5; 0.9]\\ n_s &\in [0.8; 1.2]\\ \sigma_8 &\in [0.6; 1.0]\end{aligned}$$

The value of the cosmological parameters for each simulation of a latin-hypercube of this category can be found here. Alternatively, inside each snapshot folder, there is a file called `Cosmo_params.dat` that contains the value of the cosmological parameters of that simulation. Each simulation of the latin-hypercube has a different value of the initial random seed. The value of the initial random seed of each simulation is written in the file `ICs/2LPT.param` inside each simulation folder.

The differences between the three latin-hypercubes are these:

- **standard**: This latin-hypercube contains 2,000 standard simulations with 512^3 particles each. The snapshots, halo catalogues...etc of this latin-hypercube are located in a folder called `latin_hypercube`. The folder names are `X`, where `X` goes from 0 to 1999.
- **fixed**: This latin-hypercube contains 2,000 fixed simulations with 512^3 particles each. The snapshots, halo catalogues...etc of this latin-hypercube are located in a folder called `latin_hypercube`. The folder names are `NCV_X` where `X` goes from 0 to 1999.
- **high-resolution**. This latin-hypercube contains 2,000 standard simulations with 1024^3 particles each. The snapshots, halo catalogues...etc of this latin-hypercube are located in a folder called `latin_hypercube_HR`. The folder names are `X`, where `X` goes from 0 to 1999.

Note: The simulations in the standard and high-resolution latin-hypercubes share the same initial random seed. E.g. the simulation 723 of the standard latin-hypercube has the same initial random seed as the simulation 723 of the high-resolution latin-hypercube. The only difference is the maximum k sampled in each.

18.2 nwLH

The simulations in this category include massive neutrinos. There is one single latin-hypercube in this category, and it contains 2,000 simulations that vary the value of $\Omega_{\rm m}$, $\Omega_{\rm b}$, h, n_s, σ_8, M_ν, and w. The limits of this latin-hypercube are set by

$$\begin{aligned}
\Omega_{\rm m} &\in [0.1; 0.5] \\
\Omega_{\rm b} &\in [0.03; 0.07] \\
h &\in [0.5; 0.9] \\
n_s &\in [0.8; 1.2] \\
\sigma_8 &\in [0.6; 1.0] \\
M_\nu &\in [0.01; 1.0]\ {\rm eV} \\
w &\in [-1.3; -0.7]
\end{aligned}$$

The value of the cosmological parameters of each simulation of the latin-hypercube can be found here. Alternatively, inside each snapshot folder, there is a file called `Cosmo_params.dat` that contains the value of the cosmological parameters of that simulation. Each simulation of the latin-hypercube has a different value of the initial random seed. The value of the initial random seed of each simulation is written in the file `ICs/NGenIC.param` inside each simulation folder.

Note: Note that the initial conditions of these simulations have been generated using the Zel'dovich approximation, while the initial conditions of latin-hypercubes that do not include neutrinos were generated using 2LPT.

The snapshots, halo catalogues...etc of this latin-hypercube are located in a folder called `latin_hypercube_nwLH`. The folder names are `X`, where `X` goes from 0 to 1999.

CHAPTER

NINETEEN

BIG SOBOL SEQUENCE

The Big Sobol Sequence (BSQ) is a collection of 32,768 N-body simulations designed for machine learning applications. Each simulation follows the evolution of 512^3 dark matter particles in a periodic comoving volume of $(1000\ h^{-1}\mathrm{Mpc})^3$. Each of these simulations have a different initial random seed and a value of the cosmological parameters Ω_m, Ω_b, h, n_s, σ_8 that are arranged in a Sobol sequence with boundaries (the value of the cosmological parameters for each BSQ simulation can be found here):

$$\begin{aligned}\Omega_\mathrm{m} &\in [0.10; 0.50]\\ \Omega_\mathrm{b} &\in [0.02; 0.08]\\ h &\in [0.50; 0.90]\\ n_s &\in [0.80; 1.20]\\ \sigma_8 &\in [0.60; 1.00]\end{aligned}$$

The value of the other cosmological parameters is the same in all simulations: $M_\nu = 0.0$ eV, $w = -1$, $\Omega_\mathrm{K} = 0$. The initial conditions were generated at $z = 127$ using 2LPT, and the simulations have been run using Gadget-III with a slightly more stringent force accuracy parameters than the other Quijote simulations.

Warning: As of January 7th 2024, 16,384 simulations have been run and are publicly available in both globus and binder (see *Data access*). The remaining simulations are being run and they are made publicly available inmediatly. The expected time to have the full set run is summer 2024.

For each simulation we dump 11 snapshots at redshifts 6, 5, 4, 3, 2, 1.5, 1, 0.7, 0.5, 0.2, and 0. We then post-process that data and saved halo catalogs, power spectra, bispectra, and density fields. We now describe the different data we store:

19.1 Snapshots

We have saved full snapshots for the initial conditions (ICs) and at redshifts 1 (`snap_006.hdf5`) and 0 (`snap_010.hdf5`). Note that that snapshots at redshifts 0 and 1 only contain a single file, in contrast with standard Quijote ones that have 8. This data can be read in the standard way (see *Snapshots*).

19.2 Halo catalogs

For each of the 11 snapshots per simulation we have generated both FoF and Rockstar halo catalogs. We have also run consistent trees on the Rockstar catalogs and we have saved the generated merger tree. For FoF, the convention is this:

- 000: redshift 6
- 001: redshift 5
- 002: redshift 4
- 003: redshift 3
- 004: redshift 2
- 005: redshift 1.5
- 006: redshift 1
- 007: redshift 0.7
- 008: redshift 0.5
- 009: redshift 0.2
- 010: redshift 0

We refer the reader to *Halo catalogs* for details on how to read these files.

19.3 Power spectra

For each snapshot of each simulation we have computed the matter power spectrum in real- and redshift-space and saved the results.

19.4 Bispectra

For each snapshot of each simulation we have computed the matter bispectrum in real- and redshift-space and saved the results. The bispectrum is computed on grids with 256^3 voxels and it contains ~2000 triangles down to $k \sim 0.5\, h\mathrm{Mpc}^{-1}$.

19.5 Density fields

We have generated density fields with the matter field with 256^3 voxels in real- and redshift-space for all 11 available redshifts. The density fields have been generated using the Cloud-in-Cell (CIC) mass assignments scheme. The files are stored as hdf5 files, and can be read as this

```
import numpy as np
import h5py

f = h5py.File('df_m_CIC_z=0.00.hdf5', 'r')
df = f['df'][:]  #df contains the number of particles in each voxel
f.close()
```

CHAPTER

TWENTY

HADES

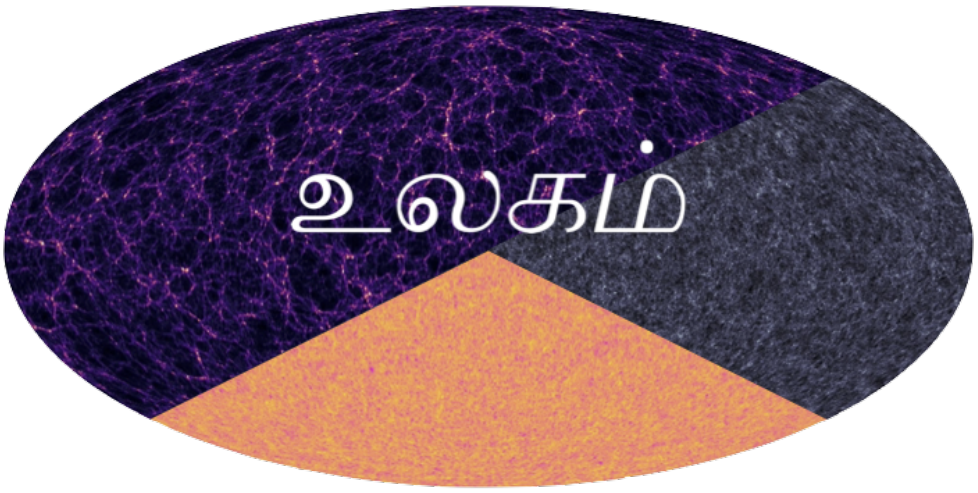
The HADES simulations were the precursor of Quijote, and they contained around 1,000 N-body and hydrodynamic simulations run with different neutrino masses. HADES was designed to study neutrino effects of cosmological observables, while Quijote philosophy is more generic and not just focused on neutrinos.

Quijote now contains all HADES data. The data is however stored on tape, but can be retrieved back and placed in globus, url, and binder. If you need this please reach out to villaescusa.francisco@gmail.com

CHAPTER

TWENTYONE

ULAGAM SIMULATIONS



The Ulagam simulations are companion runs to Quijote, and in particular, are full-sky lightcone simulations that are consistent with the Quijote initial conditions. The Ulagam suite has around 4,000 N-body simulations that are primarily meant for (i) Fisher forecasting of cosmology parameters related to wCDM or to primordial non-gaussianity, and (ii) estimating covariances for various data vectors. There are 2,000 simulations at fiducial cosmology, and 100 each for the various +/- parameter values.

For details see the Ulagam website.

Team:

- Dhayaa Anbajagane (Chicago)
- Chihway Chang (Chicago)
- Hayden Lee (Chicago)
- Marco Gatti (Upenn)

CHAPTER

TWENTYTWO

SNAPSHOTS

The snapshots are stored in either Gadget-II format or HDF5. They can be read using the readgadget.py and readsnap.py scripts. If you have Pylians installed you already have them. The user can find an example on how to read and manipulate Quijote snapshots in *Tutorials*.

The snapshots only contain 4 blocks:

- Header: This block contains general information about the snapshot such as redshift, number of particles, box size, particle masses...etc.
- Positions: This block contains the positions of all particles. Stored as 32-floats
- Velocities: This block contains the velocities of all particles. Stored as 32-floats
- IDs: This block contains the IDs of all particles. Stored as 32-integers. (This block may be removed in the future to reduce the size of the snapshots)

An example on how to read a snapshot is this:

```
import numpy as np
import readgadget

# input files
snapshot = '/home/fvillaescusa/Quijote/Snapshots/h_p/snapdir_002/snap_002'
ptype    = [1] #[1](CDM), [2](neutrinos) or [1,2](CDM+neutrinos)

# read header
header   = readgadget.header(snapshot)
BoxSize  = header.boxsize/1e3  #Mpc/h
Nall     = header.nall         #Total number of particles
Masses   = header.massarr*1e10 #Masses of the particles in Msun/h
Omega_m  = header.omega_m      #value of Omega_m
Omega_l  = header.omega_l      #value of Omega_l
h        = header.hubble       #value of h
redshift = header.redshift     #redshift of the snapshot
Hubble   = 100.0*np.sqrt(Omega_m*(1.0+redshift)**3+Omega_l)#Value of H(z) in km/s/(Mpc/h)

# read positions, velocities and IDs of the particles
pos = readgadget.read_block(snapshot, "POS ", ptype)/1e3 #positions in Mpc/h
vel = readgadget.read_block(snapshot, "VEL ", ptype)     #peculiar velocities in km/s
ids = readgadget.read_block(snapshot, "ID  ", ptype)-1   #IDs starting from 0
```

In the simulations with massive neutrinos it is possible to read the positions, velocities and IDs of the neutrino particles. Notice that the field should contain exactly 4 characters, that can be blank: `"POS "`, `"VEL "`, `"ID "`. The number in the name of the snapshot represents its redshift:

- 000 ——> z=3
- 001 ——> z=2
- 002 ——> z=1
- 003 ——> z=0.5
- 004 ——> z=0

Warning: In February 2023 we compressed the Qujote snapshots due to storage limitations. While the format is exactly the same, you may enconter problems reading them if you don't use Pylians. In order to read them you will need both hdf5 and hdf5plugin.

For instance, if you are reading the snapshots using h5py directly, you will need to install hdf5plugin, `python -m pip install hdf5plugin`, and then import both h5py and hdf5plugin

```
import h5py
import hdf5plugin
```

Reach out if you experience problems.

22.1 Initial conditions

On top of the snapshots at redshifts 0, 0.5, 1, 2, and 3, we also provide the initial conditions for each simulation. Those can be stored as hdf5 files or as Gadget format I files. In both cases, you can read the positions, velocities, and IDs of the particles using the above example just using as snapshot the name of the initial conditions, for instance:

```
snapshot = '/home/fvillaescusa/Quijote/Snapshots/w_p/ICs/ics'
```

If you want to use the linear matter power spectrum used to create the initial conditions, take a look at *Linear power spectra*.

Note: We note that the particle IDs are unique across snapshots. For instance, particles with an ID equal to 43623 at redshifts 0, 0.5, and 127 represent the very same particle at different times. This can be used to track particles back/forward in time; for instance, can be used to identify the Lagrangian region of a halo or a void.

22.2 Compression

The particle positions, velocities, and PID, are stored in HDF5 files, using HDF5 compression filters to reduce the disk usage. Specifically, the files use the Blosc compression filter, as implemented in the hdf5plugin Python package. Blosc compression applies a transpose to the data then passes it to zstandard, all of which is lossless and transparent to the user. As a preconditioning step to increase the Blosc compression ratio, we manually null out some bits of the positions and velocities to increase the compression ratio. This step is lossy. The typical total compression ratio is 2.5x.

The positions are stored as absolute coordinates in float32 precision. The lossy preconditioning we apply is to set several of the low bits in the float32 significand to zero. The number of bits nulled out is B=6 for the 1024^3 simulations, B=7 for 512^3, and B=8 for 256^3. This introduces a fractional error of 2^(-24+B), which is 3.8e-6 for the 1024^3 simulations. Since these are 1 Gpc/h simulations, this is 3.8 kpc/h precision worst-case. The softening length in all cases is 1/40th of the interparticle spacing, or 24.4 kpc/h for 1024^3. Therefore, the lossiness is 6.4x smaller than the softening length and should have a minimal impact on science analyses.

Likewise, we null out 11 low bits of the velocities, for a fractional precision of 0.01%. The velocity rarely goes above 6000 km/s in LCDM simulations, so this is a worst case of 0.6 km/s precision.

No lossy compression is applied to the IC files, or to the PIDs.

Each HDF5 file also has a new group called `/CompressionInfo` whose attributes contain a JSON string describing the exact compression options used.

The scripts used to do the compression are here: https://github.com/lgarrison/quijote-compression

Check *FAQ* if you are having problems reading the snapshots.

CHAPTER

TWENTYTHREE

HALO CATALOGS

Quijote contains FoF and Rockstar halo catalogs. The `Halo` folder contains three folders:

23.1 FoF

The FoF halo catalogs can be read through the readfof.py script. If you have Pylians installed you already have it. An example on how to read a halo catalog is this (we provide further details on how to read and manipulate these catalogs in *Tutorials*):

```
import readfof

# input files
snapdir = '/home/fvillaescusa/Quijote/Halos/FoF/s8_p/145/' #folder hosting the catalogue
snapnum = 4                                                #redshift 0

# determine the redshift of the catalogue
z_dict = {4:0.0, 3:0.5, 2:1.0, 1:2.0, 0:3.0}
redshift = z_dict[snapnum]

# read the halo catalogue
FoF = readfof.FoF_catalog(snapdir, snapnum, long_ids=False,
                          swap=False, SFR=False, read_IDs=False)

# get the properties of the halos
pos_h = FoF.GroupPos/1e3            #Halo positions in Mpc/h
mass  = FoF.GroupMass*1e10          #Halo masses in Msun/h
vel_h = FoF.GroupVel*(1.0+redshift) #Halo peculiar velocities in km/s
Npart = FoF.GroupLen                #Number of CDM particles in the halo
```

The number in the name of the halo catalogue represents its redshift:

- 000 ——> z=3
- 001 ——> z=2
- 002 ——> z=1
- 003 ——> z=0.5
- 004 ——> z=0

Note: The above correspondence applies to the majority of the simulations but not to all of them. For instance, for Quijote-ODD, 000 represents redshift 1 while 001 corresponds to redshift 0. Thus, we always recommend reading the

redshift of the correspond snapshot.

23.2 FoF_id

This folder contains FoF halo catalogs. There are two differences with respect to the above FoF folder. First, these halo catalogs contain the IDs of the particles belonging to the halos and second, it has been run over the compressed snapshots. Thus, there may be some small (likely negligible) differences among with respect to the halo catalogs in the FoF folder. For this reason we keep both halo catalogs. These halo catalogs can be read in exactly the same way as above, but now you can also access the IDs of the particles in a given halo as

```
import readfof

# input files
snapdir = '/home/fvillaescusa/Quijote/Halos/FoF_id/s8_p/145/' #folder hosting the␣
↪catalogue
snapnum = 4                                                   #redshift 0

# determine the redshift of the catalogue
z_dict = {4:0.0, 3:0.5, 2:1.0, 1:2.0, 0:3.0}
redshift = z_dict[snapnum]

# read the halo catalogue
FoF = readfof.FoF_catalog(snapdir, snapnum, long_ids=False,
                          swap=False, SFR=False, read_IDs=True)

# get the properties of the halos
pos_h = FoF.GroupPos/1e3            #Halo positions in Mpc/h
mass  = FoF.GroupMass*1e10          #Halo masses in Msun/h
vel_h = FoF.GroupVel*(1.0+redshift) #Halo peculiar velocities in km/s
Npart = FoF.GroupLen                #Number of CDM particles in the halo

# get the IDs of the halos
IDs_h = FoF.GroupIDs

# To get the IDs of the particles belong to the first halo one would do
IDs_0  = IDs_h[0:Npart[0]]
pos_0  = pos_h[0]
mass_0 = mass_h[0]

# Similarly, to get the IDs of the particles in the second halo one would do
IDs_1  = IDs_h[Npart[0]:Npart[0]+Npart[1]]
pos_1  = pos_h[1]
mass_1 = mass_h[1]
```

23.3 Rockstar

Quijote also contain Rockstar halo catalogs. A typical Rockstar folder will contain the following files:

- `out_X.list`. These are the Rockstar-generated halo+subhalo catalogs. X usually goes from 0 to 4, and it represents the snapshot number. E.g. the rockstar catalog corresponding to the snapdir_004 would be out_4.list. Those are ASCII files where the header describes the content of the file.
- `out_X_pid.list`. These are the Rockstar-generated halo+subhalo catalogs. X usually goes from 0 to 4, and it represents the snapshot number. E.g. the rockstar catalog corresponding to the snapdir_004 would be out_4.list. Those are ASCII files where the header describes the content of the file. The main difference between these files and the `out_X.list` is that `out_X_pid.list` contains an additional column called PID that allow to distinguish between halos and halos. For halos PID $= -1$ while for subhalos PID is the parent halo ID.
- `rockstar_params.cfg`. This file contains the Rockstar parameter file.
- `rockstar.slurm`. This file contains the slurm submission script used to run Rockstar.
- `rockstar.slurm`. This file contains the output from the slurm script.
- `rockstar.cfg`. The Rockstar-generated configuration file. This is generated by Rockstar when running it.
- `output.dat`. The output generated by Rockstar when running it.

In general, we recommend using the `out_X_pid.list` files that can be read easily with something like this:

```
import numpy as np

# catalog file
f_catalog = '/home/fvillaescusa/Quijote/Halos/fiducial/0/out_4_pid.list'

# read the halo catalog
data = np.loadtxt('f_catalog')

# we can now get the different properties of the halos
Mvir = data[:,2]
Vmax = data[:,3]
PID  = data[:,41]
```

Important: In some cases, like in the BSQ simulations, there are some additional folders, like `hlists` and `trees`. These folders contains the halo/subhalo catalogs and merger trees generated after running consistent trees. We note that consistent trees needs multiple snapshots to run, so only some Quijote simulations have these folders. In the case these folders exists, we recommend the user to use them. E.g. it is better to read the `hlist/hlist_1.00000.list` file than the `out_4_pid.list` as the former contains more information.

CHAPTER

TWENTYFOUR

GALAXY CATALOGS

ChangHoon Hahn has created a set of tens of thousands of galaxy catalogues from the Quijote simulations called the Molino catalogues.

You can find all the information and how to access this data here.

CHAPTER

TWENTYFIVE

SANCHO SUITE

The Sancho Suite of galaxy mock catalogs consist of $240,000$ galaxy catalogs in redshift-space scanning across 11 cosmologies, 3 massive neutrino cosmologies, 6 non-Gaussian initial conditions and 11 Halo Occupation Distribution (HOD) model parameters. The average number density of each box is $n_g \sim 3 \times 10^{-4}\,(h/\mathrm{Gpc})$ at $z = 0.5$ and they are generated from the halo catalogs of Quijote simulations, each run with 512^3 particles on a $1\,(\mathrm{Gpc}/h)^3$ box. The fiducial HOD model is tailored such that the catalogs mock the CMASS galaxies observed by BOSS.

25.1 Organization

The **Sancho catalogs** are organized in the following way:

- $15,000$ mocks at the fiducial cosmological and HOD parameter values.
- $172,500$ mocks at 23 different cosmological/HOD parameter values and $45,000$ mocks at different amplitudes of different types of non-Gaussian initial conditions. For each cosmology and HOD set of values, there are 500 N-body realizations and, for each of these, 5 realizations of the HOD are run, making a total of 2500 mocks.
- Redshift space displacements are computed in the distant observer approximation along each of the axes. There are three different files for each realization/redshift.

25.2 Cosmologies and HOD implementation

We vary 5 cosmological parameters, Ω_m, Ω_b, h, n_s and σ_8 plus the sum of massive neutrinos $\sum m_\nu$ and 3 different types of primordial non-Gaussianities, $f_{\rm NL}^{\rm local}$, $f_{\rm NL}^{\rm equil}$ and $f_{\rm NL}^{\rm equil}$, following the same naming and step convention as the rest of the Quijote suite.

The HOD model adopted is as described in Zheng et al and it consists of 5 parameters:

- $M_{\rm min}$ is the lowest mass of a halo that can host any galaxy
- $\sigma_{\log M}$ regulates the central galaxy occupation, which is modeled as a Bernoulli distribution
- The satellite galaxy distribution is a Poisson distribution with a mean that depends on three parameters: M_0, M_1, and α.

> The code implementing this HOD algorithm has been developed by Juan Calles and is freely available here .

We summarize all cosmologies and HOD models in the following table:

	Cosmological Parameters									HOD parameters					
name	Ω_m	Ω_b	h	n_s	σ_8	$\sum m_\nu$	$f_{\rm NL}^{\rm local}$	$f_{\rm NL}^{\rm equil}$	$f_{\rm NL}^{\rm ortho}$	$\log M_{\rm min}$	$\sigma_{\log M}$	$\log M_0$	α	$\log M_1$	realizations
fiducial	0.3175	0.049	0.6711	0.9624	0.834	0	0	0	0	13.0	0.2	13.1	0.75	14.25	15,000
step	0.01	0.002	0.02	0.02	0.015	*	100	100	100	0.025	0.025	0.2	0.2	0.2	500

* As for the mass of neutrinos, M_ν, there are three steps corresponding to the total mass of neutrinos of 0.1, 0.2, and 0.4 eV.

25.3 Access to data

Sancho can be accessed through the dedicated folder in Globus. Folders are organized in the same way as the rest of the Quijote suite: */Cosmology/#realization/files*

Inside each of these directories you can find the following products:

- Catalogs in Python binary including position, velocity and type (central or galaxy),
- Power spectrum measurements including monopole, quadrupole and hexadecapole (see details in the next section),
- Bispectrum measurements, for now monopole only, quadrupole coming soon!

The file naming for catalogs have the following structure:

`A_HOD_B_NFW_C_1Gpc_z0.50_D_E.npz`

where

- A: cosmology with the same naming convention as Quijote products (fiducial,Om_p,Om_m,...)
- B: HOD set of parameters. 'fid' is the fiducial set, and the rest of parameters are named as in previous section, using '_m' and '_p' to indicate steps.
- C: ID of HOD realization (from 0 to 4)
- D: RSD direction (1:x, 2:y, 3:z)
- E: ID of run

Example: `Mnu_p_HOD_fid_NFW_sample0_1Gpc_z0.50_RSD3_run0.npz`

For power spectrum and bispectrum measurements, files have an additional string `ps_sancho` and `bisp_sancho`, respectively, at the beginning of the file name, and the file format is `.dat`.

25.4 Metadata

To access relevant metadata of each catalog, a simple code *metadata.py* is available at the parent folder of the Sancho suite. Usage:

```
python metadata.py --cat_file "catalog_file.npz"
```

The code will output a JSON file with metadata listing cosmology, simulation , HOD and measurements specifications.

25.5 How to read catalogs

After download, the catalogs can be used in Python with the following script:

```
import numpy as np
filename = 'filename.npz'
cat = np.load(filename)
pos = cat['pos']        # shape: (N_galaxies, 3) --> X,Y,Z position of each galaxy in␣
↪Mpc/h
vel = cat['vel']        # shape: (N_galaxies, 3) --> Vx, Vy, Vz velocity of the galaxy␣
↪in km/s
gtype = cat['gtype']    # shape: scalar --> Type of galaxy, central: 1, satellite: 0
```

25.6 Power spectrum and Bispectrum

We measure the galaxy redshift power spectrum using the public code PBI4. We use a fourth-order density interpolation and interlacing scheme described in Sefusatti et al. Bins have width of $\Delta k = 2k_f$, where $k_f = 0.006h/Mpc$ is the fundamental frequency of the box, and are computed up to $k_{\rm max} = 0.3h/Mpc$.

The structure of the Power spectrum files is:

$$k \;|\; k_{\rm avg} \;|\; P_0(k) \;|\; P_2(k) \;|\; P_4(k) \;|\; N_{\rm modes}$$

where $k_{\rm avg}$ is the value of k inside a bin averaged over the bin, in units of h/Mpc. P_0, P_2 and P_4 are the monopole, quadrupole and hexadecapole, respectively. The units of the power spectra are $({\rm Mpc}/h)^3$. On the third line of each file, you can find two numbers, corresponding to the number of galaxies for this catalog, and the related Poisson shot-noise.

In python, the files can be read as

```
import numpy as np
filename = 'ps_sancho_fiducial_HOD_fid_NFW_sample0_1Gpc_z0.50_RSD1_run1.dat'
k, avgk, pk,avgP2, avgP4, Nmodes= np.loadtxt(filename, unpack=True)
```

To get shot-noise:

```
def getSN(filename):
    f = open(filename)
    fline = f.readline()
    fline = f.readline()
    fline = f.readline()
    psn = float(fline.split(' ')[-1])
    f.close()
    return psn
```

Using the same binning, we also measure the bispectrum for each galaxy catalog, resulting in 1654 triangles up to $k_{\rm max} = 0.3h/{\rm Mpc}$.

The structure of the Bispectrum files are:

$$k_1/k_f \;|\; k_2/k_f \;|\; k_3/k_f \;|\; P(k_1) \;|\; P(k_2) \;|\; P(k_3) \;|\; B(k_1,k_2,k_3) \;|\; B(k_1,k_2,k_3) + {\rm SN} \;|\; N_{\rm tr}$$

where ${\rm SN} = 1/n^2 + (P(k_1) + P(k_2) + P(k_3))/n$ is the bispectrum Poisson shot-noise and $N_{\rm tr}$ is the number of triangles in a give triangle bin.

In python, the files can be read as

```
import numpy as np

k1, k2, k3, Pk1, Pk2, Pk3, B0, B0+BSN, N_tri= np.loadtxt(filename, unpack=True)
```

25.7 Team

The Sancho Suite of galaxy mock catalogs was developed in 2023 by:

- Matteo Biagetti (Area Science Park, Italy)
- Juan Calles (PUCV, Chile)
- Jacky Yip (UW–Madison, USA)
- Emilio Bellini (IFPU, Italy)

If you use data from Sancho, please cite the Yip, Biagetti, Cole, Bellini, Calles, Shiu (2023).

CHAPTER

TWENTYSIX

GIGANTES VOIDS

Gigantes is a collection of void catalogs created by running VIDE on top of Quijote data. It is a massive dataset aimed at investigated the properties and information content of the underdense regions of the cosmic web. We refer the reader to the Gigantes website for details on the data.

The video below shows examples of the Gigantes voids together with the positions of the galaxies used to identified them. Credit: Wang et al. 2022.

CHAPTER

TWENTYSEVEN

VOID CATALOGS

Beside Gigantes, Quijote also contain catalogs of voids identified using a spherical-overdensity algorithm (check this website for details). The catalogs are stored as hdf5 files and they contain the following blocks:

- pos: the positions of the void centers in Mpc/h
- radius: the sizes of the voids in in Mpc/h
- VSF: the void size function
- VSF_Rbins: the radii bins of the void size function
- parameters: the values of the void finder parameters used to generate the void catalogue

In python, the files can be read as

```
import h5py

f = h5py.File('/home/fvillaescusa/Quijote/Voids/fiducial/0/void_catalogue_m_z=0.hdf5', 'r
↪')
pos        = f['pos'][:]        #void center positions in Mpc/h
radius     = f['radius'][:]     #void radii in Mpc/h
VSF        = f['VSF'][:]        #VSF (#voids/dR/Volume)
VSF_Rbins  = f['VSF_Rbins'][:]  #VSF radii in Mpc/h
parameters = f['parameters'][:] #parameters used to run the void finder
f.close()
```

CHAPTER

TWENTYEIGHT

POWER SPECTRA

28.1 Linear power spectra

The different folders contain both the CAMB parameter files and the matter power spectrum at z=0. In some cases transfer functions and power spectra for neutrinos, CDM, baryons, and CDM+baryons are also present. The format of the power spectrum files is

- k | P(k)

where the units of k and P(k) are comoving h/Mpc and (Mpc/h)^3, respectively. For the `fiducial`, `0m_p`, `0m_m`, `0b_p`, `0b_m`, `0b2_p`, `0b2_m`, `h_p`, `h_m`, `ns_p`, `ns_m`, `s8_p`, `s8_m`, `LC_p`, `LC_m`, `EQ_p`, `EQ_m`, `OR_LSS_p`, `OR_LSS_m`, `OR_LSS_p`, `OR_LSS_m` the name of the matter power spectrum files at z=0 is `CAMB_matterpow_0.dat`. For `Mnu_p`, `Mnu_pp`, and `Mnu_ppp` the files are called instead `XeV_Pm_rescaled_z0.0000.txt`, where X = 0.1(Mnu_p), 0.2(Mnu_pp) and 0.4(Mnu_ppp). For the latin_hypercube simulations, the files are named `Pk_mm_z=0.000.txt`.

Note that the matter power spectra at $z = 0$ are not normalized (this is because the normalization is performed in the code that generates the initial conditions). The normalization factor is stored in the file `Normfac.txt`. One example on how to obtain the correct normalized matter power spectrum for a given cosmology is this:

```
import numpy as np

f_Pk   = '/home/fvillaescusa/Quijote/Linear_Pk/ns_p/CAMB_TABLES/CAMB_matterpow_0.dat'
f_norm = '/home/fvillaescusa/Quijote/Linear_Pk/ns_p/Normfac.txt'

k, Pk   = np.loadtxt(f_Pk, unpack=True)
Normfac = np.loadtxt(f_norm)

Pk_norm = Pk*Normfac
```

Caution: For the primordial non-Gaussianity simulations, `LC_p`, `LC_m`, `EQ_p`, `EQ_m`, `OR_LSS_p`, `OR_LSS_m`, `OR_LSS_p`, `OR_LSS_m`, the linear power spectra files contain the Gaussian linear matter power spectrum from CAMB. The code that generates the initial conditions will take this Gaussian power spectrum and generate the non-Gaussian initial conditions.

28.2 Non-linear power spectra

The format of the power spectra are:

- k | P(k) for power spectra in real-space
- k | P0(k) | P2(k) | P4(k) for power spectra in redshift-space

where P0(k), P2(k) and P4(k) are the monopole, quadrupole and hexadecapole, respectively. The units of k are h/Mpc, while for the power spectra are (Mpc/h)^3.

In redshift-space there are three different files for each realization/redshift. These have been computed by placing the redshift-space distortions along the three different axes.

In python, the files can be read as

```
import numpy as np

k, Pk = np.loadtxt('/home/fvillaescusa/Quijote/Pk/matter/fiducial/3/Pk_m_z=0.txt',␣
↪unpack=True)
k, Pk0, Pk2, Pk4 = np.loadtxt('/home/fvillaescusa/Quijote/Pk/matter/fiducial/3/Pk_m_RS1_
↪z=0.txt', unpack=True)
```

28.3 Marked power spectra

The files whose name starts with

- Mk_ contain marked power spectra M(k) evaluated at wavenumber k
- Xk_ contain the cross spectra between marked and standard density field X(k) evaluated at wavenumber k

The unit of k is h/Mpc, while the one of M(k) and X(k) is (Mpc/h)^3.

Files with measurements performed in the fiducial cosmology have name

- Mk_fiducial0-4999_....hdf5
- Xk_fiducial0-4999_....hdf5

where the first numbers (in the above case 0-4999) indicate the realizations saved in the file, and the dots specify the marked model considered.

The remaining files contain measurements performed in the other cosmologies and from 500 realization per cosmology. Their name is

- Mk_fTH_....hdf5
- Xk_fTH_....hdf5

Also in this case the dots specify the marked model considered.

In python, the files can be read as

```
import numpy as np

f = h5py.File(FILENAME, 'r')
k = f['k'][:]
# Fiducial cosmology
Mk = f['i'][:]
```

(continues on next page)

(continued from previous page)

```
# Massive neutrino cosmologies
Mk = f['cosmo/i_suffix'][:]
# Other cosmologies
Mk = f['cosmo/i'][:]
```

where i is the number of the realization, cosmo is the wanted cosmology and suffix can be

- 'm' for the total matter field
- 'cb' for the cold dar matter plus baryons

In order to see the name of each cosmology type

```
print(list(f.keys()))
```

CHAPTER

TWENTYNINE

BISPECTRA

The format of the individual bispectra files are:

- k1/kf | k2/kf | k3/kf | P0(k1) | P0(k2) | P0(k3) | B0(k1,k2,k3) | Q(k1,k2,k3) | B_SN(k1,k2,k3) | counts

where k1, k2, k3 specify the length of the triangle sides, P0(k) is the power spectrum monopole, B0(k1,k2,k3) is the bispectrum monopole, Q(k1,k2,k3) is the reduced bipsectrum, B_SN is the bispectrum shot noise correction, and counts is the number of triangles in the bin. B0 is already shot-noise corrected. The header specifies kf, the fundamental mode, and Nhalo, the number of halos.

The individual bispectra files can be read in python as follows,

```
import numpy as np

k1, k2, k3, p0k1, p0k2, p0k3, b123, q123, b_sn, cnts = np.loadtxt(FILENAME, skiprows=1,␣
↪unpack=True, usecols=range(10))

# read header to get Nhalo
hdr   = open(FILENAME).readline().rstrip()
Nhalo = int(hdr.split('Nhalo=')[-1])
```

Alternatively, sets of bispectra files for a specific redshift and cosmology can easily be accessed

```
import h5py

fbk    = h5py.File(FILENAME, 'r')
k1     = fbk['k1'][...]
k2     = fbk['k2'][...]
k3     = fbk['k3'][...]
p0k1   = fbk['p0k1'][...]
p0k2   = fbk['p0k2'][...]
p0k3   = fbk['p0k3'][...]
b123   = fbk['b123'][...]
q123   = fbk['q123'][...]
b_sn   = fbk['b_sn'][...]
cnts   = fbk['counts'][...] # triange counts
Nhalos = fbk['Nhalos'][...] # number of halos
files  = fbk['files'][...]  # names of individual files.
```

CHAPTER

THIRTY

CORRELATION FUNCTIONS

The format of the correlation functions are:

- R | xi(R) for correlation functions in real-space
- R | xi0(R) | xi2(R) | xi4(R) for correlation functions in redshift-space

where xi0(R), xi2(R) and xi4(R) are the monopole, quadrupole and hexadecapole, respectively. The units of R are Mpc/h, while the different xi are dimensionless.

In redshift-space there are three different files for each realization/redshift. These have been computed by placing the redshift-space distortions along the three different axes.

In python, the files can be read as

```
import numpy as np

R, xi = np.loadtxt('/home/fvillaescusa/Quijote/CF/matter/fiducial/0/CF_m_1024_z=0.txt',␣
↪unpack=True)
R, xi0, xi2, xi4 = np.loadtxt('/home/fvillaescusa/Quijote/CF/matter/fiducial/0/CF_m_RS0_
↪1024_z=0.txt', unpack=True)
```

CHAPTER

THIRTYONE

PDFS

The format of the PDF files is:

- delta | pdf

where delta is the density contrast (rho/< rho > - 1).

In python, the files can be read as

```
import numpy as np

delta, pdf = np.loadtxt('/home/fvillaescusa/Quijote/PDF/matter/latin_hypercube/0/PDF_m_5.
↪0_z=0.txt', unpack=True)
```

CHAPTER

THIRTYTWO

DENSITY FIELDS

32.1 3D fields

The 3D density fields are located in the New York cluster (see *Data access*) under the `3D_cubes` folder.

There are different folders for the different cosmologies. Inside each cosmology folder there are the folder containing the data for the different realizations. Inside each of those folders the 3D density fields can be found with names as `df_m_X_Y_z=Z.npy`, where X can be 64, 128, 256, or 512, and it represents the grid size of the cube. Y represents the mass assignment scheme used to construct the density field, and can be something like CIC (cloud-in-cell) or PCS (piece-wise spline). Z represents the redshift of the density field. For instance, `df_m_256_CIC_z=0.npy` contains the 3D density field on a grid with 256^3 voxels constructed using the CIC mass-assignment scheme at $z = 0$.

Note: These fields are constructed in real-space. Please reach us if you need them in redshift-space.

The files can be read simply as

```
import numpy as np

df = np.load('/home/fvillaescusa/Quijote/3D_cubes/Om_p/df_m_128_PCS_z=0.npy')
```

Warning: Density fields with a large number of voxels occupy a significant amount of disk space, so they may not be available. However, constructing these fields is straightforward and it can be done directly on binder; thus, there is no need to download and process the data. We have examples of how to create these density fields directly on binder in *Tutorials*.

32.2 2D fields

2D fields (say images) can be constructed from the above 3D fields by taking a slice and projected it into 2D. For instance:

```
import numpy as np

# read the 3D density field
df_3D = np.load('/home/fvillaescusa/Quijote/3D_cubes/Om_p/df_m_128_PCS_z=0.npy')

# take a slice of 4 voxels width, i.e. 1000/128*4 = 31.25 Mpc/h
```

(continues on next page)

(continued from previous page)

```
# along z-direction and project into 2D by computing the mean value
df_2D = np.mean(df_3D[:,:,0:4], axis=2)
```

CHAPTER

THIRTYTHREE

LOGO

The below gif image has been created by linearly interpolating two images: 1) an image of the large-scale structure from one Quijote simulation, and 2) an image created by using the DeepDream software on top of 1). The goal of the animation is to emphasize and highlight regions of the cosmic web in a novel way.

The images below show Don Quijote riding his horse with the sky showing the large-scale structure of the Universe from its *traditional* version (top-right) to its *machine learning* version (bottom-left).

CHAPTER

THIRTYFOUR

LICENSE

CHAPTER

THIRTYFIVE

FAQ

35.1 I am having problems reading the snapshots. What should I do?

Note that the snapshots have been compressed to reduce the storage needs. To read them, you need to install the hdf5plugin with, e.g. `python -m pip install hdf5plugin`.

If you are using python, you need to load that library as:

```
import h5py
import hdf5plugin
```

Note that if you are reading the snapshots with Pylians you dont need to do anything else. However, keep in mind that you may need to update Pylians to the latest version with `python -m pip install --upgrade Pylians`.

If you are using a non-python software (e.g. using running Rockstar in a snapshot or running a simulation from the initial conditions using Gadget) you need to set `HDF5_PLUGIN_PATH` environment variable to make use of the HDF5 compression filters. The path can be obtained as:

```
python -c "import hdf5plugin; print(hdf5plugin.PLUGIN_PATH)"
```

and therefore, one should set in a terminal:

```
export HDF5_PLUGIN_PATH=$(python -c "import hdf5plugin; print(hdf5plugin.PLUGIN_PATH)")
```

After doing this, the code can be run as normal. For further details check this. If you experience problems with this please reach out to us at villaescusa.francisco@gmail.com or lgarrison@flatironinstitute.org.

35.2 The documentation says that there are 15,000 realizations for the fiducial cosmology, but I can only find 8,000. Where is the rest?

The 15,000 realizations of the fiducial model are split among the New York and San Diego Cluster. The New York cluster contains the first 8,000 while the rest is in the San Diego cluster. Check *Data access* for details.

35.3 How many latin-hypercubes are there?

Currently, there are 4 latin-hypercubes, each of them having 2,000 simulations:

- Three of them only vary $\Omega_{\rm m}$, $\Omega_{\rm b}$, h, $n_{\rm s}$, σ_8.
- One of them vary $\Omega_{\rm m}$, $\Omega_{\rm b}$, h, $n_{\rm s}$, σ_8, M_ν, w.

CHAPTER

THIRTYSIX

HELP

For problems, questions and general help you can reach us at villaescusa.francisco@gmail.com