THE UNIVERSITY *of* EDINBURGH
School of Physics
and Astronomy

Senior Honours Project

Neutrino Properties From Large Scale Structure

Finlay Sime
October 2025

Abstract

This report sets out to investigate how neutrino wakes can be revealed, detected and what properties can be deduced from the results. Data from the N-body Quijote simulation is used to stack centred and velocity aligned halos to reduce noise and increase a dipole feature. The overdensity difference between CDM and neutrinos is determined to try and reveal a neutrino wake. No neutrino wake was found in this report likely due to the dipole density fluctuations found (confirming recent studies) dominating over any possible neutrino wake signal.

Date: 31/10/2021

Declaration

I declare that this project and report is my own work.

Supervisor: Dr. Yan-Chuan Cai 10 Weeks

Contents

1 Introduction

The neutrino was originally proposed in 1930 by Pauli in order to account for the missing momentum and energy in beta decay [1]. In the standard model (SM) there are three flavours of neutrinos. There is the electron neutrino, discovered in the Cowan-Reines neutrino experiment (actually detected the electron antineutrino, [2]), the muon neutrino, discovered in 1962 in [3], and the tau neutrino that was first observed in 2001 in [4]. Each of these neutrino flavours are produced through different means (ν_e produced in nuclear reactions, ν_μ produced in pion beams and ν_τ produced with a proton beam on a tungsten target) [5, 6, 4]. Given that ν_e are produced in nuclear reactions, they should be produced in large quantities in the Sun from the first step in the p-p chain (the neutrino only interacts with the weak force). Neutrinos produced in the Sun are typically referred to as "solar neutrinos". When two protons fuse together into a deuterium nucleus, they release a positron and an electron neutrino:

$$p + p \rightarrow d + e^+ + \nu_e, \tag{1}$$

which can be detected on Earth [7]. These were first detected in the Homestake Mine experiment by using the reaction

$$\nu_e + {}^{37}_{17}Cl \rightarrow {}^{37}_{18}Ar + e^-, \tag{2}$$

however only approximately a third of the expected neutrinos were detected [8]. This became the first piece of evidence for a phenomenon known as neutrino oscillations. Neutrino oscillations are caused by the mixing between mass and flavour eigenstates, a neutrino is produced with a definite or "pure" flavour, however, will then propagate as mass eigenstates mixing into other neutrino flavours [9]. This is why only a third of the expected neutrinos were detected in the Homestake Mine experiment, by the time the neutrinos had reached Earth, they had oscillated into other flavour states. Further solar neutrino experiments have taken place to provide more evidence for the existence of neutrino oscillations and flavour change (eg. [10, 11, 12]). Neutrinos can be produced in the atmosphere as well (i.e. atmospheric neutrinos) and when trying to detect these neutrinos leads to a similar conclusion. When cosmic rays strike the atmosphere they produce neutrinos (mostly ν_e and ν_μ) but, again, the ratio of detected muon to electron neutrinos was low and so provides evidence of these neutrinos oscillating into tau neutrinos [13].

A neutrino oscillation can be most easily understood in the simplified case of two neutrino flavours with two neutrino mass eigenstates. This derivation can be found in appendix A.1, [14]. The case of three neutrino flavours mixing can be described by the PMNS (Pontecorvo–Maki–Nakagawa–Sakata) matrix,

$$|\nu_l\rangle = \sum_i U^*_{li} |\nu_i\rangle \,, \tag{3}$$

where $|\nu_l\rangle$ is the neutrino field of left handed neutrinos (neutrinos with), U^*_{li} is the PMNS matrix and $|\nu_i\rangle$ is the neutrino mass eigenstate (Majorana or Dirac, another unknown property of neutrinos; a Majorana particle is its own antiparticle and a Dirac particle is not) with mass m_i [15].

Exact neutrino masses have not yet been determined and remains a very active and exciting field of research. Although bounds to neutrino masses have been determined. Because of the neutrino oscillations two of the three neutrino mass eigenstates must be significantly larger than the other one thus the neutrino mass eigenstates can be placed into a hierarchy. There is the normal hierarchy (NH, $m_1 < m_2 \ll m_3$) and the inverted hierarchy (IH, $m_3 \ll m_1 < m_2$). By inspecting the mass ordering in each hierarchy, the mass-squared difference ($\Delta m_{ij}^2 = m_i^2 - m_j^2$) between the masses is clearly $\Delta m_{21}^2 \ll \Delta m_{32}^2 \approx \Delta m_{31}^2 > 0$ for the NH and $\Delta m_{21}^2 \ll -(\Delta m_{31}^2 \approx \Delta m_{32}^2 < 0)$ for IH. The neutrino mass hierarchy therefore depends on the sign of Δm_{31}^2. Presently the most up to date values for the mass-squared difference is $\Delta m_{21}^2 \approx 7.6 \times 10^{-5}$ eV^2 and $|\Delta m_{31}^2| \approx 2.5 \times 10^{-3}$ eV^2[16]. An overview of the experimental techniques used to determine the squared-mass difference can be found in [17]. Different hierarchies prefer different theoretical models so determining mass ordering will eliminate large amounts of different models in particle physics [16, 18]. A way this can be done is by finding the total sum of the neutrino mass eigenstates. In doing so a mass hierarchy can be determined. The current bounds on the neutrino mass are in the region of 0.7 to 0.12 eV (at a 95% CL) [19, 20]. This was achieved through cosmological studies as cosmology is one of the possible ways the sum of neutrino masses can be constrained. This is the focus of this report, to attempt to find neutrino properties through studying the cosmological N-body simulation Quijote.

The structure of this report is as follows, Section 2 will discuss the thermal history and evolution of neutrino over time and the free-streaming property of neutrinos and Section 3 will follow this by explaining the observable impacts neutrinos can have on the universe and the wakes they form around dark matter halos. Section 5 will go through the methodology used in this report, the results are then shown and discussed in Section 6 and finally the report is concluded in Section 8.

2 Thermal History & Free-streaming

The cosmic neutrino background (CNB, sometimes called "relic neutrinos") is predicted by the SM but has not yet been detected although its existence has been established by the agreement of calculated and observed cosmological values. In the early universe, neutrinos were produced through weak interactions at very high temperatures and kept in thermal equilibrium with the primordial plasma (electrons, positrons and photons to name a few) [21]. These were reactions such as $e^- + e^+ \rightarrow \nu_e + \bar{\nu}_e$ but after a short period of time (approximately one second), the neutrinos decoupled from the plasma and stopped being produced. This happened when the weak interaction rate

$$\Gamma \approx \langle \sigma n_\nu \rangle \,, \tag{4}$$

fell below the Hubble expansion rate H, where σ_ν is the cross-section of the electron-neutrino weak interaction and n_ν is the neutrino number density. This occurred at an estimated temperature that can be derived by setting $\Gamma = H$. The expansion rate is given by the Hubble parameter

$$H = \sqrt{\left(\frac{8\pi\rho}{3M_P^2}\right)}, \tag{5}$$

for relativistic species the number density scales as $n \propto T^3$ and the cross-section of the weak interaction can be approximated as the Fermi constant times energy squared ($G_F^2 E^2$) allowing us to make the approximation

$$\Gamma \approx G_F^2 T^5, \tag{6}$$

(as E scales with T). The total energy density, $\rho \propto T^4$ and M_P is the Planck mass, these can be substituted into equation 5 to give

$$H \propto \frac{T^2}{G_F^{1/2}}. \tag{7}$$

By equating equations 6 and 7 the decoupling temperature is found as

$$T_{dec} \approx \left(\frac{1}{G_F^2 M_P}\right)^{1/3}, \tag{8}$$

which is approximately equal to a few MeV. More specifically the decoupling occurred at $T_{\nu_e} = 1.87$ MeV and $T_{\nu_\mu,\nu_\tau} = 3.12$ MeV [22, 21, 23]. These decoupled neutrinos were then "free-streaming" and travelling at relativistic speeds because the neutrino energies were dominated by momentum. As the universe continued to expand and cool, the momentum of the neutrinos (which follows a Fermi-Dirac distribution) redshifted. This is easily shown as the neutrino momentum and temperature are proportional to the inverse of the scale factor,

$$p \propto \frac{1}{a(t)}, T_\nu \propto \frac{1}{a(t)}, \tag{9}$$

so the neutrinos lose kinetic energy and slow down over time, eventually becoming non-relativistic [23]. The transition to the non-relativistic regime occurred when the thermal energy of the neutrino became comparable to its rest mass, and after such the neutrino energies were dominated by its rest mass [24]. When this happened exactly will depend on the neutrino mass and the neutrino flavour. This brings neutrinos to their present day state, free-streaming and non-relativistic.

The free-streaming scale is the distance neutrinos can travel freely due to their thermal velocities without clustering. It is defined as the comoving (factors out the expansion of the universe) distance a neutrino travels in one Hubble time (inverse of the Hubble constant)

$$\lambda_{fs} \approx \frac{\sigma_\nu}{H_0}, \tag{10}$$

where σ_ν is the velocity dispersion and $H_0 \approx 70$ km/s/Mpc is the Hubble constant. The velocity dispersion can be approximated as $\sigma_\nu \approx 3T_{\nu,0}/m_\nu$ where m_ν is the neutrino mass and $T_{\nu,0}$ is the temperature of neutrinos today (and the speed of light has been set as $c = 1$). For $T_{\nu,0} \approx 1.7 \times 10^{-4}$ eV and $m_{\nu,1,2,3} = 0.1, 0.2$ and 0.4 eV the velocity dispersion

is $\sigma_{\nu,1} = 1530$ km/s, $\sigma_{\nu,2} = 765$ km/s and $\sigma_{\nu,3} = 382.5$ km/s. Substituting these values into equation 10 the free-streaming scale for the three neutrino masses are $\lambda_{fs,1} \approx 21.9$ Mpc, $\lambda_{fs,2} \approx 10.9$ Mpc and $\lambda_{fs,3} \approx 5.46$ Mpc [25]. It is obvious that as neutrino mass increases, the free-streaming scale associated with said mass decreases. So as the neutrino mass becomes heavier it becomes easier for them to cluster like CDM does. Figure 1, from [26], clearly shows the correlation of CDM and neutrino density fields and that heavier neutrinos cluster more than lighter ones.

3 Neutrino's Impact on the Universe

The CNB affects the evolution of the homogeneous and the inhomogeneous universe. The homogeneous universe is at the background level and is described in the Friedmann equations. The expansion of the universe is described through the Friedmann equation for the Hubble parameter

$$H(a) = \frac{8\pi G}{3}\rho(a) - \frac{K}{a^2} \tag{11}$$

where a is the cosmic scale factor, G is the gravitational constant, K parametrises the curvature of space and ρ is the total energy density. In the early universe ($T \leq 1$ MeV), neutrinos were relativistic meaning they contributed to the radiation energy density like photons do, $\rho_r = \rho_\gamma + \rho_\nu$ (once non-relativistic they contribute to the matter density). This means neutrinos increased the energy density in the early universe which increased H in equation 11 [23, 27]. This has a consequential cosmological impact on the epoch of matter-radiation equality. The epoch of matter-radiation equality occurred when the matter density, ρ_m, was equal to the radiation density, ρ_r. Given that heavier neutrinos, as will be discussed in the next Section, become non-relativistic earlier they contribute more to the matter density in the universe. This leads to an observable impact in the universe as one looks back into the cosmic microwave background (CMB) that can be used to constrain the neutrino mass [17]. This can be measured through the matter power spectrum, $P(k)$ which is a Fourier transform of the two point correlation function of matter in the universe. The correlation function is explained more in Section 4 and it is a very powerful tool in cosmology.

Shown in Figure 2 are the matter power spectrums illustrated at different redshift as presented in [23]. Measuring the matter power spectrum can therefore be used to constrain the total neutrino mass and even the individual neutrino masses. The power spectra in Figure 2 show that the IH ordering of neutrino masses has stronger clustering at the scale of approximately $10^{-2}k$, where a large k corresponds to short scale distances and a large k corresponds to larger scale distances. Simulations, such as in [28], show that the amplitude of the matter power spectrum is reduced at large k in the presence of massive neutrinos.

The CNB has a similar effect on the halo mass function (HMF). The HMF quantifies the abundance of dark matter halos at different masses, it is often written as $\frac{dn}{dM}$ where n is the comoving number density of halos. To begin with the HMF is dependent on the initial density perturbations and growth of structure [29]. Since the HMF is dependent on the growth of structure, and the growth of structure is suppressed on small scale beneath the free-streaming scale, this can lead to fewer low-mass halos [30].

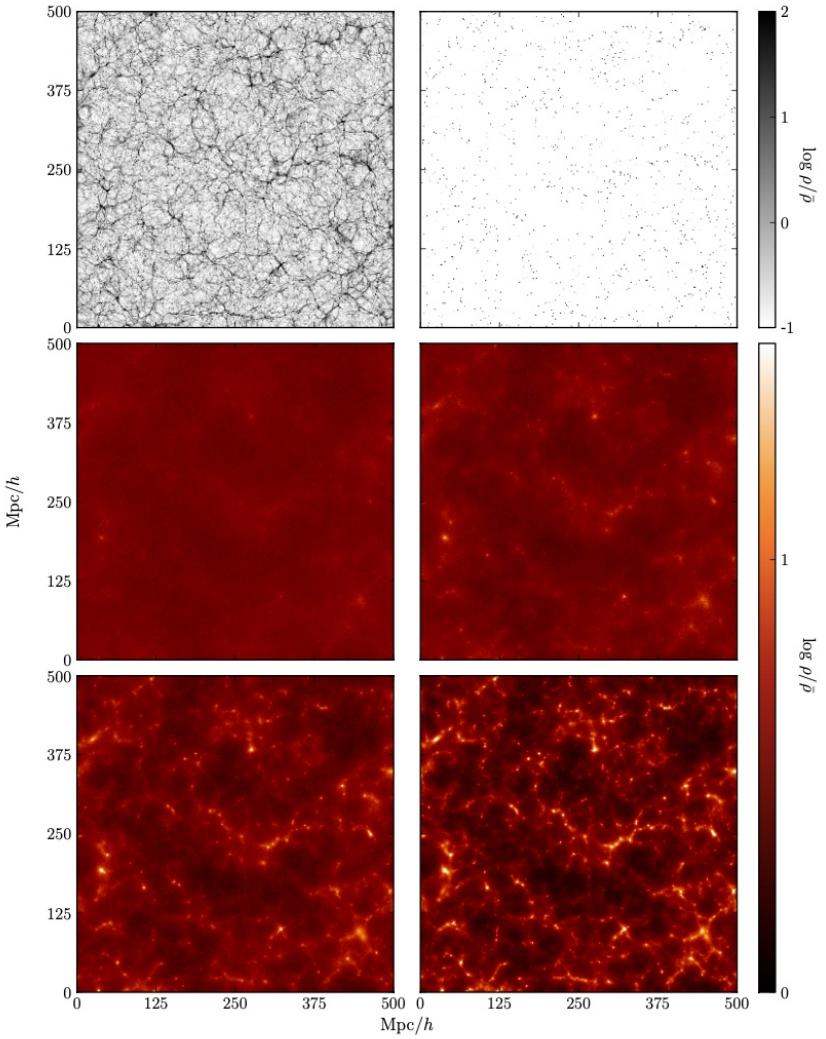

Figure 1: Along the top row are dark matter (left) and halo (right) density slices from 0.2 eV neutrino simulations. Starting from the middle left and finishing in the bottom right are neutrino density slices from 0.05, 0.1, 0.2 and 0.4 eV neutrino simulations, respectively.

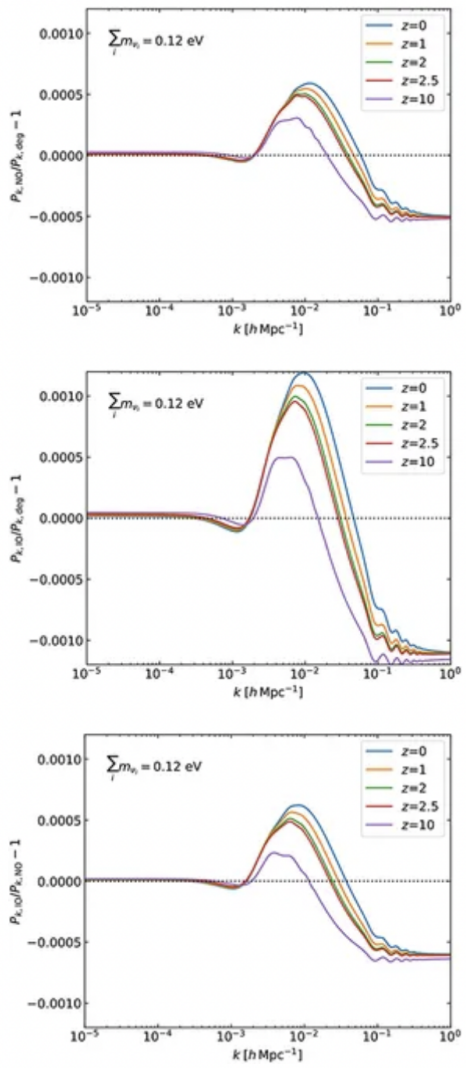

Figure 2: The top row shows the matter power spectrum for normal mass ordering, the middle row shows the matter power spectrum for inverted ordering and the bottom row shoes the ratio of the two hierarchies.

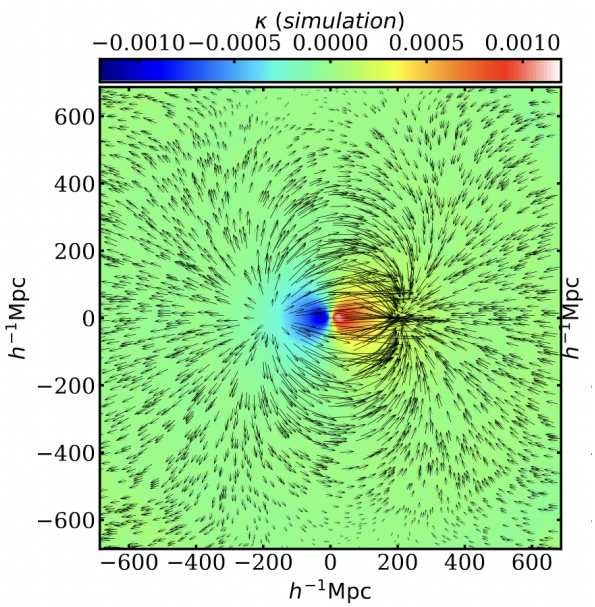

Figure 3: Sourced from [33]. This is the projected matter density fluctuation corresponding to the large-scale velocity field.

4 Neutrino Wakes

There are two types of dipole in the direction of the moving dark matter halo, there is the large scale density fluctuations caused by the large gravitational potential of the matter and the second is the neutrino wake. Both shall be discussed here. The "bulk velocity field" is the average coherent motion of particles (in this case for the CDM and CNB) at any given point in space and time and the "relative bulk velocity field" is the difference between two bulk velocity fields. Simulations show that the neutrino free-streaming induces a relative velocity field between the CNB and the CDM [26]. The relative velocity field has been proposed as a method to measuring the sum of neutrino masses [31]. Due to the relative velocity between the neutrinos and the CDM, a dipole distortion in the cross-correlation function can be observed over large linear scales[31, 32]. The cross-correlation function,

$$\xi_{CDM,\nu}(r) = \langle \delta_{CDM}(x)\delta_{\nu}(x+r)\rangle \tag{12}$$

defines the excess probability of finding a neutrino at a distance $x + r$ from an overdensity of CDM at position x. A dipole in this function is a front-back asymmetry. This dipole is expected to produce a high density region in front of the moving halo, as the halo accelerates towards the source of gravity, and a low density behind the halo. This effect is expected to happen over tens or hundreds of Mpc and has been simulated in [33] as shown in Figure 3. The dipole also emerges in the relative velocity field of the CDM and neutrinos. The local peculiar velocity is expected to point in this direction as it is parallel to the direction of gravity. This effectively pulls the halo along the up the gravity gradient and a neutrino density plot should appear equivalently, with a high density region ahead of the halo and low density behind it [33].

Shown in Figure 3 is a stacked velocity field between of pure CDM. It produces a

dipole distortion as predicted by [31] and a similar method to that produced this plot will be used in this report. Dark matter halos travelling through the CNB experience a force causing them to decelerate, this is known as dynamical friction and is due to interactions with the neutrinos. The dynamical friction causes a displacement in the dark matter halo's position over one Hubble time of

$$\Delta x \propto v_{\nu,c} m_\nu^4 \tag{13}$$

where $v_{\nu,c}$ is the rms relative velocity, so different neutrino masses will have different velocities and displacements that can be calculated [34]. This displacement of the halo relative to the neutrinos leads to neutrinos accumulating anisotropically behind the halo and forming an overdensity - the neutrino wake. The neutrino wake is a dipole phenomenon taking place on much smaller scales in comparison to the large-scale dipole feature, making it much harder to simulate. This can be thought of as equivalent to a wake behind a boat or a stone skimming on water as the dark matter halo moves through the CNB it gravitationally focuses neutrinos into a small overdensity downstream of the halo [35, 25]. A derivation of the clustering of a neutrino wake behind a single halo can be found in [25] and the Figure from such can be found in appendix A.2, Figure 7. [25] also completes a derivation for the neutrino density contrast for wakes in a general CDM distribution in an expanding universe. The neutrino density contrast in this case is shown to be

$$\delta_\nu(\vec{k}) = \left(\frac{k_{fs}}{k}\right)^2 \left[1 + i\gamma(\hat{k}\cdot\hat{z})\frac{\sigma_{rel}}{\sigma_\nu}\right]\delta(\vec{k}) \tag{14}$$

where $p_{rel}(\vec{k}) = i(\hat{k}\cdot\hat{z})\sigma_{rel}\delta(\hat{k})$ is the relative momentum term of the neutrinos to the CDM in Fourier space, k_{fs} is the free-streaming scale (in Fourier space), γ is a numerical factor, σ is the velocity dispersion and $\delta(\vec{k})$ is the background CDM density contrast [25, 34, 35]. It is easy to spot in this equation that the neutrino density contrast will indeed be strongest close to the halo (i.e. below the free-streaming scale), parallel to the halos direction of motion (here this is the $\hat{z}$ direction) and downstream of the halo, where you would expect to find a neutrino wake. Given that the neutrino density contrast of the wake is proportional to the free-streaming scale and the neutrinos relative momentum to the CDM the neutrino density contrast is proportional to the neutrino mass and measuring the density contrast of the neutrino wake could be used to constrain the neutrino mass [35]. Since the neutrino wake is much weaker than large scale dipole and potentially easily washed out by the velocity dispersion of the neutrinos, detecting it in a simulation will require a low noise environment with lots of halos for data. This makes halo stacking essential with velocity aligned to average out the background noise and constructively add the neutrino wake.

The primary method that is theorised to detect a neutrino wake or density dipole is gravitational lensing. Gravitational lensing is a phenomenon where light from distant sources on its way to Earth passes by a large object with strong gravity (e.g. clusters of dark matter). The gravity will bend the path of the light as a lens does and hence the name, gravitational lensing. The amount of bending is described by Einstein's theory of general relativity and can therefore be used to measure the mass of the astrophysical

object causing the lensing. This method has already been put forward by several papers [33, 35].

5 Method

The data used in this paper was processed in python and originally produced in a Quijote simulation. The Quijote simulation is an N-body simulation that modelled massive neutrinos of three different masses, these being the $M_\nu^+ = 0.1$ eV, $M_\nu^{++} = 0.2$ eV and $M_\nu^{+++} = 0.4$ eV [1] [36]. The information describing the set-up and initial parameters of the simulation can be read about in [37]. The python modules used were the python libraries for the analysis of numerical simulations (aka. Pylians3) modules [2]

The first results in Figure 4 were made in the code found in Section A.3. The data was imported using the `readgadget` module and sorted through by reading of the headers (particle's positions, velocities, etc). Information about the size of the box in the simulation is then printed for the user's convenience and a grid size to average the particles positions into is chosen. The grid here is 3 dimensional, so the data will be contained into a grid made up of the value of L cubed (L being the length of the grid, for the results in Figure 4 $L = 512$ for the CDM and $L = 228$ for the neutrinos). The density fields are computed into this grid using the `MASL.density_field_gadget()` function and stored as NumPy arrays. In order to display the data on a 2 dimensional plot, the simulation data must be sliced and averaged. A slice (in the Y-Z plane) of an inputted depth is created, such that the new 3 dimensional array is of $L \times L\times$ slice depth, where L is the length of the simulation box. This density values of this slice are then averaged over the slice depth (in the X direction) to create a 2 dimensional array ($L \times L$ in size) ready for plotting. The location of the slice within the box is irrelevant as it is statistically the same. Following this the slice is easily plotted using `matplotlib` functions and these results are shown in Figure 4.

Following on from this, in order to reveal a density of neutrinos and CDM around a moving halo as shown in Figure 5, the images must be stacked. In the code found in appendix A.4, the halo of mass less than $10^{14} M_\odot$ are filtered out of the grid. Next a slice is taken at the edge of the box, as was done in the previous paragraph, and each halo is centred into the slice (without losing the periodicity of the box) and then rotated such that their velocity is parallel to the positive x-axis. This halo centred and aligned slice is then averaged over the X direction and the 2D plot is stored in a list. This process is repeated for every halo in the slide. In order to artificially increase the number of halos available and average out the noise, this process is repeated in all 3 dimension. After taking a slice in the Y-Z plane, centring, aligning and storing each halo averaged over the X direction, a slice is taken in the X-Z plane and the X-Y plane and the process is repeated over the Y and Z directions respectively. Since each direction is orthogonal to one another it is independent and this can reliably improve the construction of the dipole and neutrino wake features. This process of taking a slice along each direction is then repeated through a python loop, where each iteration moves the slice along until

[1] More information can be read about the Quijote simulation for massive neutrinos at `https://quijote-simulations.readthedocs.io/en/latest/Mnu.html`

[2] This can be found at `https://github.com/franciscovillaescusa/Pylians3`.

the entire box has been covered such that every halo has been used 3 times. This leads to just over $80,000$ halos being used. All the stored 2D plots of centred and aligned halos are then stacked together and averaged, maximising the data that can be extracted from a single simulation. This is similar to the method found in [33]. The halos were centred using the `numpy.roll` function to maintain the periodicity of the box and rotated using the `scipy.ndimage.rotate` function. This results in the plot being a square with a circle of radius 1000 Mpc/h and the corners being a consequence of the rotation that can be ignored. The plot produced was be changed by modifying the dataset in the `ax.imshow()` function in line 348. The neutrino wake is revealed with the equation

$$\delta_{wake} = f_{opt}\delta_{\nu} - \delta_{CDM} \tag{15}$$

where the f_{opt} is the optimum factor determined by chi-squared, this derivation can be found in appendix A.5. The calculated values for f_{opt} can be found in table 1 in appendix A.6.

This method of stacking the halos is very computationally heavy and slow, taking hours for a $500 \times 500 \times 500$ grid so a faster method implementing NumPy broadcasting was tested however, while the computational time was significantly decreased in order to achieve the same grid size or higher thousands of gigabytes of memory were required. This memory-speed trade-off was not worth it in the end as getting access to over a thousand gigabytes of memory at once was not possible and so this method proved to be the best.

6 Results & Discussions

The results show in Figure 4 clearly show a clustering of CDM and neutrinos around the dark matter halos. There is also a visible increase in clustering of neutrinos around the dark matter halos as the neutrino mass increases. While the neutrino overdensity for M_{ν}^{+++} doesn't appear to reach values as high as it does in the M_{ν}^{+} case, the space between the clusters of dark matter halos is more void of neutrinos in the M_{ν}^{+++} case that in the M_{ν}^{+} case. This is expected as the the free-streaming scale of neutrinos, defined in equation 10, is inversely proportional to the neutrino mass. As neutrino mass increases the neutrinos are expected to cluster more and this can be visually seen Figure 10. These results are also in agreement with the results from [26] in Figure 1. Figure 1 also clearly shows the correlation between the decrease of the neutrino free-streaming scale with increasing neutrino mass.

Shown in Figure 5 are the stacked plots for centred and aligned halos. A high overdensity of neutrinos in front of the halos can clearly be seen, along with a low overdensity almost immediately behind the halos. This is the large scale dipole described in Section 4. The difference between the different neutrino masses is stark. For the M_{ν}^{+} neutrino mass the transition in the overdensity across the dipole is very smooth and washed out. The dipole feature is also weaker with a maximum overdensity value of 0.00875 and a minimum value of -0.00640. In comparison the M_{ν}^{+++} neutrino mass has a very clear dipole feature in the overdensity with a maximum overdensity of 0.0185 and a minimum overdensity of -0.00771. The change in the strength of the (approximately 0.578% weaker in M_{ν}^{+} compared to M_{ν}^{+++}) dipole is most likely due to the significantly larger

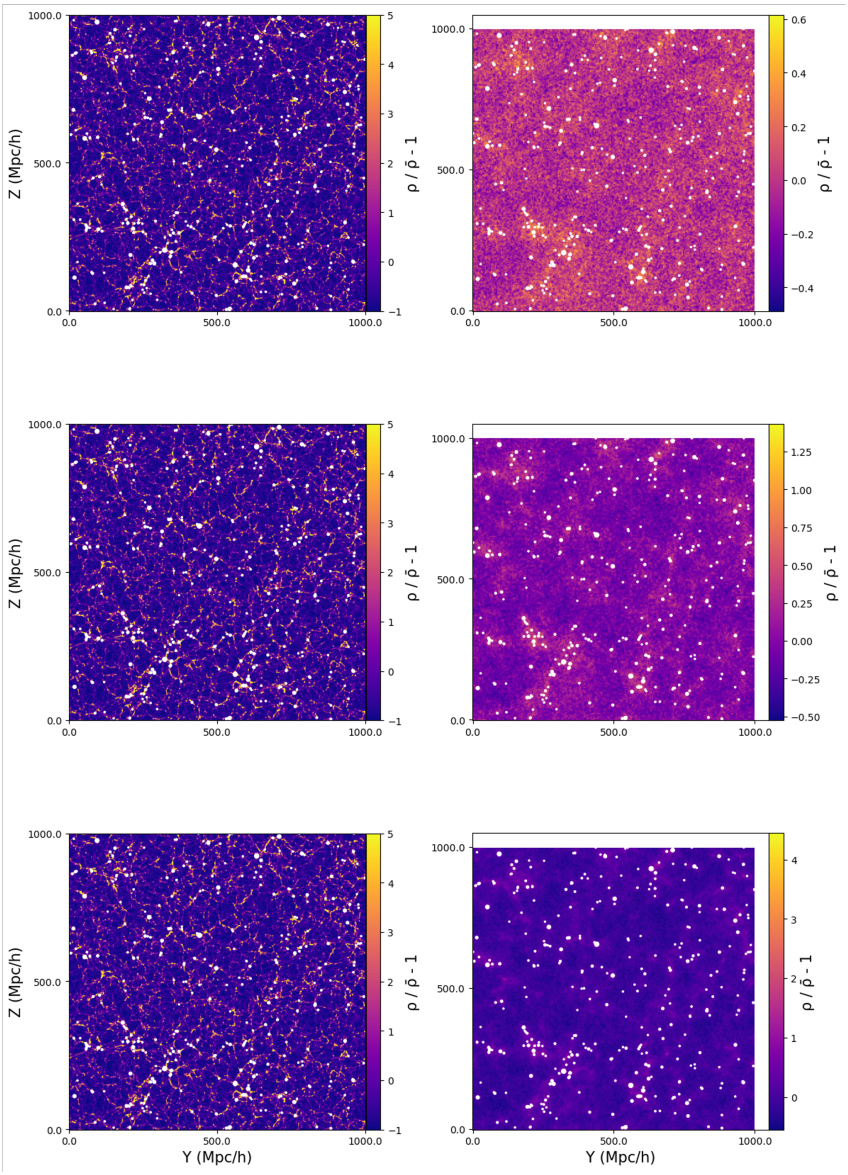

Figure 4: These are plots produced from the code found in appendix A.3. Along the top row are the results of simulations of mass M_ν^+, the middle row a mass of M_ν^{++} and the bottom for a mass of M_ν^{+++}. On the left are the CDM overdensity fields, and on the right are the neutrino overdensity fields, both have halos within their respective slice plotted over the top. The voxel size and slice depth for the CDM is 1.95 Mpc/h. The voxel size for the neutrinos is 4.39 Mpc/h and the depth of the slice is 13.16 Mpc/h.

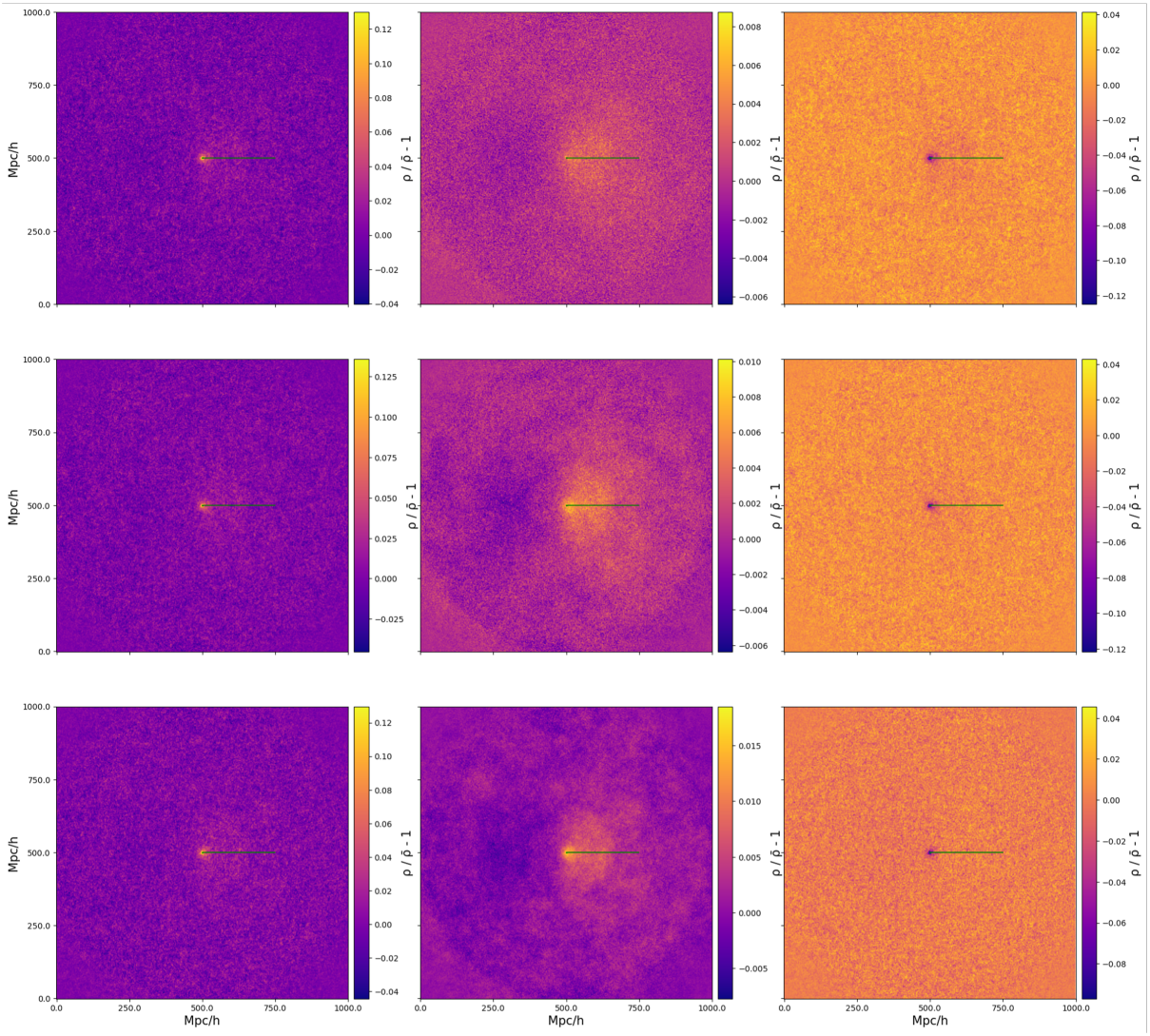

Figure 5: Shown here are the halo centred stacked plots produced from the code in Appendix A.4 where each cube of the grid has a side length of 2 Mpc/h. The top row contains M_ν^+, the middle contains M_ν^{++} and the bottom contains M_ν^{+++}. The left column is displays the CDM overdensity, the centre column displays the neutrino overdensity and the right column displays a subtraction of the CDM overdensity from the neutrino overdensity multiplied by the factor calculated in Appendix A.5. The green dot in the centre is the stacked halo position and the green arrow indicates the direction of the halo velocity.

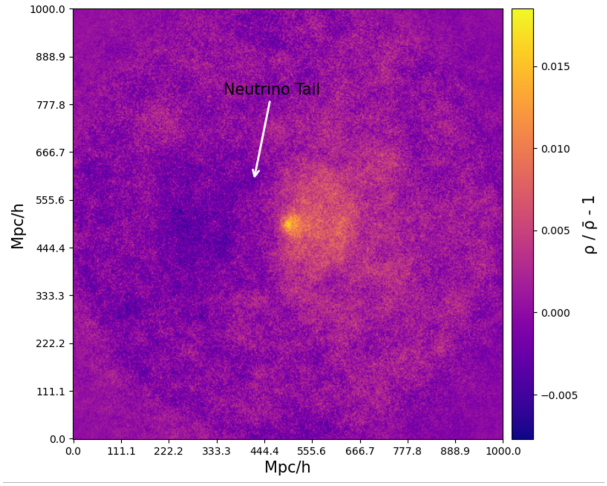

Figure 6: The stacked neutrino overdensity for the M_ν^{+++} neutrino mass. Annotated above is the neutrino "tail".

thermal dispersion and velocity of the M_ν^+ neutrinos. As shown in Section 2, the thermal dispersion for a neutrino of mass $M_\nu^+ = 0.1$ eV over three times greater than the thermal dispersion for a neutrino of mass M_ν^{+++}. For completeness the maximum overdensity for the M_ν^{++} dipole overdensity is 0.0101 and a minimum of -0.00634. A particularly interesting feature is the noticeable semi-circle of mild overdensity between the neutrino shadow (the low overdensity part of the dipole) and the halo itself, as pointed out in Figure 6. This feature is easy to spot for the M_ν^{+++} and M_ν^{++} neutrino masses but it is very faint for the M_ν^+ neutrino mass. This is again likely due to the difference in thermal dispersions. It could be a neutrino wake as it appears to be a clustering of neutrinos downstream of the halo velocity, however it is unlikely as other models have predicted the size of this wake to be much smaller and the wake should be present at non-linear scales. For example in Figure 7 the calculated neutrino wake is only about 1 Mpc in size.

This dipole feature should be produced by the CDM as well but at a smaller scale. While there is a very slight difference in the overdensity in front and behind the halo for the CDM, the difference is negligible. This is most likley due to the CDM clustering far more than the neutrinos as the thermal dispersion of CDM that is much smaller than the neutrino's. The free-streaming scale of the CDM is therefore much smaller that the neutrino free-streaming scale as well.

In the subtracted fields on the right of Figure 5 the large scale dipole feature on the neutrino plot is not clear and has been removed as expected by the methodology, however there is no apparent neutrino wake present as expected in the methodology. The high overdensity of neutrinos in front of the halo has vanished and the absence of the CDM in the centre is easy to see. Even with the optimization factor applied to the neutrino field the high density of the CDM around the dark matter halo appears to dominate the image, perhaps vanishing other features such as a neutrino wake. It is also possible that the neutrino wake isn't differentiable in the stacked neutrino plots and is "covered up" by noise i.e. the signal-to-noise ratio (SNR) is too high to produce a plot with a visible neutrino wake. This has been suggested by the paper [38] where the cumulative SNR is found to increase with neutrino mass. This would mean that the large-scale dipole

dominates over the neutrino wake signal regardless of neutrino mass. It is also possible the resolution of the plot simply isn't fine enough. In Figures 4 and 5 the voxels have a side length of 2 Mpc/h which is a good enough resolution to show the distributions and dipoles of neutrinos and CDM. If any neutrino wakes in Figure 5 are of this scale it is likely they will be hidden in the noise and the lack of resolution.

The existence of the neutrino sea and information about neutrino properties could potentially still be deduced from these plots. As described in Section 4 the presence of neutrinos could be observable through gravitational lensing. A dark matter halo could be identified through a gravitational lensing pattern and then through the fine structure of the lensing can be studied. The density dipole of neutrinos should be revealed in such along with the dipole feature produced by the CDM, even if the dipole is much weaker and smaller.

7 Future Work & Development

A major missing set of components of the neutrino and CDM behaviour around dark matter halos are the velocity fields. As stated in Section 4, [31, 33] shows how neutrino masses can be measured from the relative velocity field between neutrinos and CDM. These two fields also produce a relative dipole that could be used to infer the presence of a neutrino wake. Following a similar method to this report the relative velocity field could be constructed with the equation

$$v_{rel} = f_{opt} v_\nu - v_{CDM} \tag{16}$$

from the same method of centring and rotating halo positions and then stacking the velocity fields to produce an overall relative velocity field.

The subtraction of the CDM overdensity field from the neutrino overdensity field could also be developed further. Since the CDM produces a dipole too weak to be visible in Figure 5, the subtraction only reveals the significant difference between the clustering at small distances around the halo. This could be improved by amplifying the dipole signal in the CDM overdensity field before subtraction. This however would have to be matched with the neutrino dipole and if the physical lengths do not align this might become technically very difficult.

Improved methods of filtering the noise due to the neutrino's high velocity dispersion could also be added. Applying Gaussian filters and Fourier analysis to remove different signals could further enhance the dipole signal for the CDM, which can then be matched with the neutrino signal and removed to reveal a wake structure.

8 Conclusion

As shown neutrinos accumulate in front of dark matter halos and form a large scale dipole, confirming the results of other studies. This phenomenon is also seen very weakly for the CDM. Unfortunately no neutrino wakes were revealed by inspecting the difference between the neutrino and CDM overdensity fields. This is most likely due to the large SNR predicted in [38] and the large scale with low resolution to reveal structures such

as a neutrino wake. This means that the large scale dipole dominates over the neutrino wake regardless of neutrino mass which has not yet been shown in studies. Finding the neutrino wake signal in data such as this will therefore prove a challenge for future work. The CDM large scale dipole is negligible here as the clustering directly around the dark matter halo washes it out. Fixing this issue could produce a more balanced subtraction and reveal the weaker signal of a neutrino wake. There is also the interesting feature of the neutrino "tail" pointed out in Figure 6. This tail should be studied further as to determine whether it will become an issue in the dipole subtraction between the CDM and the neutrinos or is related to the neutrino wake. Properties of neutrinos can still be drawn from measuring the effects of this dipole and investigating the tail of the high neutrino overdensity perhaps through gravitational lensing.

As future surveys become able to measure the properties described in this report in more detail, it could be possible to measure an observable that proves the existence of neutrino wakes. More complex simulations with a more thorough analysis will also be conducted that can find potential affects caused by neutrino wakes.

The code used in this report can be found in full at the github link: `https://github.com/F1onnlagh03/Senior-Honours-Project` and is available for public use.

9 Acknowledgements

I would like to thank my supervisor Yan-Chuan Cai and Aritra Gon for their patience, explanations and introducing me to a fascinating topic.

References

[1] W. Pauli. *in letter to participants of the Conference in T¨ubingen*. 1930.

[2] C. L. Jr. Cowan et al. "Detection of the Free Neutrino: A Confirmation". In: *Science* 124.3212 (1956), pp. 103–104.

[3] G. Danby et al. "Observation of High-Energy Neutrino Reactions and the Existence of Two Kinds of Neutrinos". In: *Physical Review Letters* 9.1 (1962), pp. 36–44.

[4] K. Kodama et al. "Observation of Tau Neutrino Interactions". In: *Physics Letters B* 504.3 (2001), pp. 218–224.

[5] B. Xin et al. "Production of electron neutrinos at nuclear power reactors and the prospects for neutrino physics". In: *Phys. Rev. D* 72 (1 2005), p. 012006.

[6] Sacha E. Kopp. "Accelerator neutrino beams". In: *Physics Reports* 439.3 (2007), pp. 101–159.

[7] Anna M. Suliga et al. "A closer look at the *pp*-chain reaction in the Sun: Constraining the coupling of light mediators to protons". In: *arXiv preprint* (2020).

[8] Raymond Jr. Davis et al. "Search for Neutrinos from the Sun". In: *Physical Review Letters* 20.21 (1968), pp. 1205–1209.

[9] P.A. Zyla et al. "Neutrino Mixing". In: *Progress of Theoretical and Experimental Physics* 2020 (2020), p. 083C01.

[10] M. Altmann et al. "Complete results for five years of GNO solar neutrino observations". In: *Physics Letters B* 616 (2005), pp. 174–190.

[11] J. N. Abdurashitov et al. "Solar neutrino flux measurements by the Soviet-American gallium experiment (SAGE) for half the 22-year solar cycle". In: *Journal of Experimental and Theoretical Physics* 95 (2002), pp. 181–193.

[12] Q. R. Ahmad et al. "Direct Evidence for Neutrino Flavor Transformation from Neutral-Current Interactions in the Sudbury Neutrino Observatory". In: *Physical Review Letters* 89 (2002).

[13] Y. Fukuda et al. "Evidence for Oscillation of Atmospheric Neutrinos". In: *Physical Review Letters* 81 (1998), pp. 1562–1567.

[14] S. M. Bilenky et al. "Lepton Mixing and Neutrino Oscillations". In: *Physics Reports* 41.4 (1978), pp. 225–261.

[15] S. M. Bilenky. *Neutrino in Standard Model and beyond*. 2015.

[16] P. F. de Salas et al. "Neutrino Mass Ordering from Oscillations and Beyond". In: *Frontiers in Astronomy and Space Sciences* 5 (2018), p. 36.

[17] Particle Data Group. "14. Neutrino Masses, Mixing, and Oscillations". In: *Physical Review D* 110 (2024), p. 030001.

[18] W. Winter. "Neutrino Mass Hierarchy: Theory and Phenomenology". In: *AIP Conference Proceedings* 1666 (2015), p. 120001.

[19] Daniel Naredo-Tuero et al. *Living at the Edge: A Critical Look at the Cosmological Neutrino Mass Bound*. 2024. arXiv: `2407.13831 [astro-ph.CO]`. URL: `https://arxiv.org/abs/2407.13831`.

[20] Shouvik Roy Choudhury et al. “Updated Cosmological Constraints in Extended Parameter Space with Planck PR4, DESI Baryon Acoustic Oscillations, and Supernovae: Dynamical Dark Energy, Neutrino Masses, Lensing Anomaly, and the Hubble Tension”. In: *The Astrophysical Journal Letters* 976.1 (2024), p. L11.

[21] Julien Lesgourgues et al. “Neutrino Cosmology and Plank”. In: *New Journal of Physics* 16 (2014), p. 065002.

[22] A.D. Dolgov. “Neutrinos in cosmology”. In: *Physics Reports* 370.4–5 (2002), pp. 333–535.

[23] Martino Gerbino et al. “Status of Neutrino Properties and Future Prospects”. In: *Frontiers in Physics* 5 (2017), p. 70.

[24] Alex C. Hall et al. “Probing the neutrino mass hierarchy with cosmic microwave background weak lensing”. In: *Monthly Notices of the Royal Astronomical Society* 425.2 (2012), pp. 1170–1184.

[25] Caio Nascimento et al. “Neutrino winds on the sky”. In: *Journal of Cosmology and Astroparticle Physics* 11 (2023), p. 036.

[26] Derek Inman et al. “Precision reconstruction of the cold dark matter–neutrino relative velocity”. In: *Physical Review D* 92.2 (2015), p. 023502.

[27] S. M. Bilenky. “Neutrino in Cosmology”. In: *Physics of Particles and Nuclei* 52 (2021), pp. 337–356.

[28] Volker Springel Matteo Viel Martin G. Haehnelt. “The effect of neutrinos on the matter distribution as probed by the Intergalactic Medium”. In: *Journal of Cosmology and Astroparticle Physics* 06 (2010), p. 015.

[29] Darren S. Reed et al. “The halo mass function from the dark ages through the present day”. In: *Monthly Notices of the Royal Astronomical Society* 374 (2007), pp. 2–15.

[30] Chi-Ting Chiang et al. “Scale-dependent bias and bispectrum in neutrino separate universe simulations”. In: *Physical Review D* 97 (2018), p. 123526.

[31] Hong-Ming Zhu et al. “Measurement of Neutrino Masses from Relative Velocities”. In: *Physical Review Letters* 113 (2014).

[32] Dmitriy Tseliakhovich et al. “Relative velocity of dark matter and baryonic fluids and the formation of the first structures”. In: *Physical Review D* 82 (2010).

[33] Yan-Chuan Cai et al. *Detection of cosmological dipoles aligned with transverse peculiar velocities*. 2025.

[34] Chiamaka Okoli et al. “Dynamical friction in the primordial neutrino sea”. In: *Monthly Notices From the Royal Astronomical Society* 468 (2017), pp. 2164–2175.

[35] Hong-Ming Zhu et al. “Probing Neutrino Hierarchy and Chirality via Wakes”. In: *Physical Review Letters* 116 (2016), p. 141301.

[36] Francisco Villaescusa-Navarro et al. “The Quijote Simulations”. In: *The Astrophysical Journal* 250.1 (2020).

[37] M. Zennaro et al. “Initial conditions for accurate N-body simulations of massive neutrino cosmologies”. In: *Monthly Notices of the Royal Astronomical Society* 466.3 (2016), pp. 3244–3258.

[38] Caio B. de S. Nascimento et al. *Dark winds on the horizon: Prospects for detecting neutrino and hot dark matter wakes in large-scale structure*. 2025.

A Appendices

A.1 Appendix A

Consider ν_e and ν_μ being represented as two orthogonal linear superpositions of two different mass eigenstates, m1 and m2:

$$|\nu_e\rangle = cos(\theta)\,|\nu_1\rangle + sin(\theta)\,|\nu_2\rangle \tag{17}$$

$$|\nu_\mu\rangle = -sin(\theta)\,|\nu_1\rangle + cos(\theta)\,|\nu_2\rangle \tag{18}$$

In quantum mechanics, a wave function evolves according to $H\Psi = i\hbar\frac{d\Psi}{dt}$, so after time t the wave function ν_e will evolve to

$$|\nu_e\rangle_t = e^{-\frac{iE_1t}{\hbar}}cos(\theta)\,|\nu_1\rangle + e^{-\frac{iE_2t}{\hbar}}sin(\theta)\,|\nu_2\rangle \tag{19}$$

Equations 17, 18, 19 can be rearranged to give:

$$|\nu_e\rangle_t = e^{-\frac{iE_1t}{\hbar}}[(cos^2(\theta) + e^{\frac{i(E_1-E_2)t}{\hbar}}sin^2(\theta))\,|\nu_e\rangle - sin(\theta)cos(\theta)1 - e^{\frac{i(E_1-E_2)t}{\hbar}}\,|\nu_\mu\rangle] \tag{20}$$

The square of the amplitude of $|\nu_e\rangle$ is thus the probability of the electron neutrino after some time t being in the state ν_e is:

$$P_e(t) = |cos^2(\theta) + e^{\frac{i(E_1-E_2)t}{\hbar}}sin^2(\theta)|^2 \tag{21}$$

A.2 Appendix B

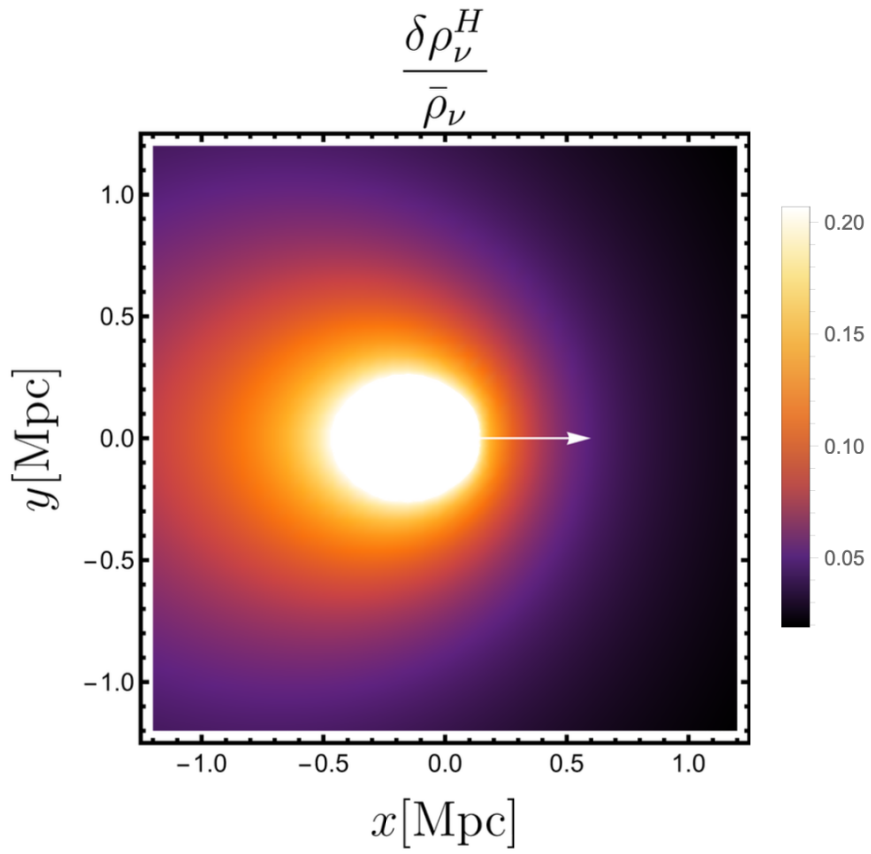

Figure 7: Anisotropic clustering of neutrinos behind a point mass halo. In this case $m_\nu = 0.1$ eV and $M_H = 10^{13} M_\odot$. Sourced from [25].

A.3 Appendix C

```
import readgadget
import numpy as np
import MAS_library as MASL

# input files
snapshot_Mn_p = '/ceph/cephfs/agon/quijote/Mn_p/0/snapdir_004/
    snap_004'
snapshot_Mn_pp = '/ceph/cephfs/agon/quijote/Mn_pp/0/snapdir_004/
    snap_004'
snapshot_Mn_ppp = '/ceph/cephfs/agon/quijote/Mn_ppp/0/snapdir_004
    /snap_004'

print("snapshot files imported")

#[1](CDM), [2](neutrinos) or [1,2](CDM+neutrinos)
```

```
13
14 ptype_CDM = [1]
15 ptype_Neutrino = [2]
16
17 # read header
18 header_p   = readgadget.header(snapshot_Mn_p)
19 BoxSize  = header_p.boxsize/1e3  #Mpc/h
20 Nall     = header_p.nall         #Total number of particles
21 Masses   = header_p.massarr*1e10 #Masses of the particles in Msun
      /h
22 Omega_m  = header_p.omega_m      #value of Omega_m
23 Omega_l  = header_p.omega_l      #value of Omega_l
24 h        = header_p.hubble       #value of h
25 redshift = header_p.redshift     #redshift of the snapshot
26 Hubble   = 100.0*np.sqrt(Omega_m*(1.0+redshift)**3+Omega_l)#Value
       of H(z) in km/s/(Mpc/h)
27
28 # read positions, velocities and IDs of the particles
29 pos_CDM_p = readgadget.read_block(snapshot_Mn_p, "POS␣",
      ptype_CDM)/1e3 #positions in Mpc/h
30 vel_CDM_p = readgadget.read_block(snapshot_Mn_p, "VEL␣",
      ptype_CDM)     #peculiar velocities in km/s
31 ids_CDM_p = readgadget.read_block(snapshot_Mn_p, "ID␣␣",
      ptype_CDM)-1
32
33
34 pos_Neutrino_p = readgadget.read_block(snapshot_Mn_p, "POS␣",
      ptype_Neutrino)/1e3
35 vel_Neutrino_p = readgadget.read_block(snapshot_Mn_p, "VEL␣",
      ptype_Neutrino)
36 ids_Neutrino_p = readgadget.read_block(snapshot_Mn_p, "ID␣␣",
      ptype_Neutrino)-1
37
38 # read header
39 header_pp   = readgadget.header(snapshot_Mn_pp)
40 BoxSize  = header_pp.boxsize/1e3  #Mpc/h
41 Nall     = header_pp.nall         #Total number of particles
42 Masses   = header_pp.massarr*1e10 #Masses of the particles in
      Msun/h
43 Omega_m  = header_pp.omega_m      #value of Omega_m
44 Omega_l  = header_pp.omega_l      #value of Omega_l
45 h        = header_pp.hubble       #value of h
46 redshift = header_pp.redshift     #redshift of the snapshot
47 Hubble   = 100.0*np.sqrt(Omega_m*(1.0+redshift)**3+Omega_l)#Value
       of H(z) in km/s/(Mpc/h)
48
49 # read positions, velocities and IDs of the particles
50 pos_CDM_pp = readgadget.read_block(snapshot_Mn_pp, "POS␣",
      ptype_CDM)/1e3 #positions in Mpc/h
```

```
vel_CDM_pp = readgadget.read_block(snapshot_Mn_pp, "VEL␣",
    ptype_CDM)     #peculiar velocities in km/s
ids_CDM_pp = readgadget.read_block(snapshot_Mn_pp, "ID␣␣",
    ptype_CDM)-1


pos_Neutrino_pp = readgadget.read_block(snapshot_Mn_pp, "POS␣",
    ptype_Neutrino)/1e3
vel_Neutrino_pp = readgadget.read_block(snapshot_Mn_pp, "VEL␣",
    ptype_Neutrino)
ids_Neutrino_pp = readgadget.read_block(snapshot_Mn_pp, "ID␣␣",
    ptype_Neutrino)-1

# read header
header_ppp   = readgadget.header(snapshot_Mn_ppp)
BoxSize   = header_ppp.boxsize/1e3  #Mpc/h
Nall      = header_ppp.nall         #Total number of particles
Masses    = header_ppp.massarr*1e10 #Masses of the particles in
    Msun/h
Omega_m   = header_ppp.omega_m      #value of Omega_m
Omega_l   = header_ppp.omega_l      #value of Omega_l
h         = header_ppp.hubble       #value of h
redshift  = header_ppp.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_CDM_ppp = readgadget.read_block(snapshot_Mn_ppp, "POS␣",
    ptype_CDM)/1e3 #positions in Mpc/h
vel_CDM_ppp = readgadget.read_block(snapshot_Mn_ppp, "VEL␣",
    ptype_CDM)     #peculiar velocities in km/s
ids_CDM_ppp = readgadget.read_block(snapshot_Mn_ppp, "ID␣␣",
    ptype_CDM)-1


pos_Neutrino_ppp = readgadget.read_block(snapshot_Mn_ppp, "POS␣",
     ptype_Neutrino)/1e3
vel_Neutrino_ppp = readgadget.read_block(snapshot_Mn_ppp, "VEL␣",
     ptype_Neutrino)
ids_Neutrino_ppp = readgadget.read_block(snapshot_Mn_ppp, "ID␣␣",
     ptype_Neutrino)-1

print("Positions␣&␣velocities␣acquired")

##############################To get the halo catalog
    ######################################
f_catalog_p = '/ceph/cephfs/agon/quijote/Mn_p/halos/0/out_4_pid.
    list'
```

```
f_catalog_pp = '/ceph/cephfs/agon/quijote/Mn_pp/halos/0/out_4_pid.list'
f_catalog_ppp = '/ceph/cephfs/agon/quijote/Mn_ppp/halos/0/out_4_pid.list'

# read the halo catalog
data_p = np.loadtxt(f_catalog_p)
data_pp = np.loadtxt(f_catalog_pp)
data_ppp = np.loadtxt(f_catalog_ppp)

print("Halo data imported")

pid_p = data_p[:,41]
idx_p = pid_p == -1  #ids of the halos
data_halo_p = data_p[idx_p]

pid_pp = data_pp[:,41]
idx_pp = pid_pp == -1
data_halo_pp = data_pp[idx_pp]

pid_ppp = data_ppp[:,41]
idx_ppp = pid_ppp == -1  #ids of the halos
data_halo_ppp = data_ppp[idx_ppp]

#Units: Masses in Msun / h
#Units: Positions in Mpc / h (comoving)
#Units: Velocities in km / s (physical, peculiar)
#BoxSize=1000.0

halo_pos_p = data_halo_p[:,8:11]
halo_velocity_p = data_halo_p[:,11:14]
halo_mass_p = data_halo_p[:,2]

halo_pos_pp = data_halo_pp[:,8:11]
halo_velocity_pp = data_halo_pp[:,11:14]
halo_mass_pp = data_halo_pp[:,2]

halo_pos_ppp = data_halo_ppp[:,8:11]
halo_velocity_ppp = data_halo_ppp[:,11:14]
halo_mass_ppp = data_halo_ppp[:,2]
print("Halo positions, velocites, & masses acquired")

halo_filter = 1e14 #Solar Masses

# Filter out halos with less mass than 1e14 solar masses
idx = np.where(halo_mass_p > halo_filter)
halo_mass_p = halo_mass_p[idx]
halo_pos_p = halo_pos_p[idx]
halo_vel_p = halo_velocity_p[idx]
```

```
131
132 idx = np.where(halo_mass_pp > halo_filter)
133 halo_mass_pp = halo_mass_pp[idx]
134 halo_pos_pp = halo_pos_pp[idx]
135 halo_vel_pp = halo_velocity_pp[idx]
136
137 idx = np.where(halo_mass_ppp > halo_filter)
138 halo_mass_ppp = halo_mass_ppp[idx]
139 halo_pos_ppp = halo_pos_ppp[idx]
140 halo_vel_ppp = halo_velocity_ppp[idx]
141
142 print(f"Halos␣with␣mass␣less␣than␣{halo_filter}␣solar␣masses␣
    removed!")
143
144 ##############################Build density field
    ########################################
145 grid_CDM = int(input("Grid␣size␣CDM:␣")) #the density field will
    have grid^3 voxels
146 grid_N = int(input("Grid␣size␣Neutrinos:␣"))
147 MAS = 'CIC' #Mass-assignment scheme:'NGP', 'CIC', 'TSC', 'PCS'
148 do_RSD = False # Dont do redshift-space distortions
149 axis = 0 #axis along which place RSD; not used here
150 verbose = True #whether to print information about progress
151
152 # print some information
153 print('BoxSize:␣%.3f␣Mpc/h'%BoxSize)
154 print('Redshift:␣%.3f'%redshift)
155 print('CDM:')
156 print('%.3f␣<␣X␣<␣%.3f'%(np.min(pos_CDM_p[:,0]), np.max(pos_CDM_p
    [:,0])))
157 print('%.3f␣<␣Y␣<␣%.3f'%(np.min(pos_CDM_p[:,1]), np.max(pos_CDM_p
    [:,1])))
158 print('%.3f␣<␣Z␣<␣%.3f'%(np.min(pos_CDM_p[:,2]), np.max(pos_CDM_p
    [:,2])))
159 print('NEUTRINOS:')
160 print('%.3f␣<␣X␣<␣%.3f'%(np.min(pos_Neutrino_p[:,0]), np.max(
    pos_Neutrino_p[:,0])))
161 print('%.3f␣<␣Y␣<␣%.3f'%(np.min(pos_Neutrino_p[:,1]), np.max(
    pos_Neutrino_p[:,1])))
162 print('%.3f␣<␣Z␣<␣%.3f'%(np.min(pos_Neutrino_p[:,2]), np.max(
    pos_Neutrino_p[:,2])))
163
164 # compute the effective number of particles in each voxel
165 delta_pos_CDM_p = MASL.density_field_gadget(snapshot_Mn_p,
    ptype_CDM, grid_CDM, MAS, do_RSD, axis, verbose)
166 delta_pos_Neutrino_p = MASL.density_field_gadget(snapshot_Mn_p,
    ptype_Neutrino, grid_N, MAS, do_RSD, axis, verbose)
167
```

```
delta_pos_CDM_pp = MASL.density_field_gadget(snapshot_Mn_pp, ptype_CDM, grid_CDM, MAS, do_RSD, axis, verbose)
delta_pos_Neutrino_pp = MASL.density_field_gadget(snapshot_Mn_pp, ptype_Neutrino, grid_N, MAS, do_RSD, axis, verbose)

delta_pos_CDM_ppp = MASL.density_field_gadget(snapshot_Mn_ppp, ptype_CDM, grid_CDM, MAS, do_RSD, axis, verbose)
delta_pos_Neutrino_ppp = MASL.density_field_gadget(snapshot_Mn_ppp, ptype_Neutrino, grid_N, MAS, do_RSD, axis, verbose)

# compute the density contrast (aka overdensity)
delta_pos_CDM_p /= np.mean(delta_pos_CDM_p, dtype=np.float64); delta_pos_CDM_p -= 1
delta_pos_Neutrino_p /= np.mean(delta_pos_Neutrino_p, dtype=np.float64); delta_pos_Neutrino_p -= 1

delta_pos_CDM_pp /= np.mean(delta_pos_CDM_pp, dtype=np.float64); delta_pos_CDM_pp -= 1
delta_pos_Neutrino_pp /= np.mean(delta_pos_Neutrino_pp, dtype=np.float64); delta_pos_Neutrino_pp -= 1

delta_pos_CDM_ppp /= np.mean(delta_pos_CDM_ppp, dtype=np.float64); delta_pos_CDM_ppp -= 1
delta_pos_Neutrino_ppp /= np.mean(delta_pos_Neutrino_ppp, dtype=np.float64); delta_pos_Neutrino_ppp -= 1

#print voxel sizes
size_CDM = BoxSize / grid_CDM
size_N = BoxSize / grid_N
print(f"Voxel size in CDM is {size_CDM}Mpc/h")
print(f"Voxel size in Neutrinos is {size_N}Mpc/h")

# create slice thickness
while True:
    Slice_CDM = int(input("No. of slices for CDM: "))
    Slice_N = int(input("No. of slices for Neutrinos: "))

    if isinstance(Slice_CDM, int) and isinstance(Slice_N, int):
        break
    else:
        print("Invalid input, please enter an integer")

slice_thickness_CDM = Slice_CDM * size_CDM
slice_thickness_N = Slice_N * size_N
print(f"CDM slice thickness: {slice_thickness_CDM} Mpc/h")
print(f"Neutrino slice thickness: {slice_thickness_N} Mpc/h")

#compute the mean along the x-axis for the slice
```

```
206 mean_density_CDM_p = np.mean(delta_pos_CDM_p[:Slice_CDM,:,:],axis
    =0)
207 mean_density_Neutrino_p = np.mean(delta_pos_Neutrino_p[:Slice_N
    ,:,:],axis=0)
208
209 mean_density_CDM_pp = np.mean(delta_pos_CDM_pp[:Slice_CDM,:,:],
    axis=0)
210 mean_density_Neutrino_pp = np.mean(delta_pos_Neutrino_pp[:Slice_N
    ,:,:],axis=0)
211
212 mean_density_CDM_ppp = np.mean(delta_pos_CDM_ppp[:Slice_CDM,:,:],
    axis=0)
213 mean_density_Neutrino_ppp = np.mean(delta_pos_Neutrino_ppp[:
    Slice_N,:,:],axis=0)
214
215 print('Image␣shape:',mean_density_CDM_p.shape)
216 print('%.3e␣<␣mass␣<␣%.3e'%(np.min(mean_density_CDM_p), np.max(
    mean_density_CDM_p)))
217
218 print("CDM:")
219 # filter out particles in slice
220 indexes_CDM_p = np.where((pos_CDM_p[:,0]<slice_thickness_CDM))
221 pos_slide_CDM_p = pos_CDM_p[indexes_CDM_p]
222 print('%.3f␣<␣X␣<␣%.3f'%(np.min(pos_slide_CDM_p[:,0]), np.max(
    pos_slide_CDM_p[:,0])))
223 print('%.3f␣<␣Y␣<␣%.3f'%(np.min(pos_slide_CDM_p[:,1]), np.max(
    pos_slide_CDM_p[:,1])))
224 print('%.3f␣<␣Z␣<␣%.3f'%(np.min(pos_slide_CDM_p[:,2]), np.max(
    pos_slide_CDM_p[:,2])))
225 print('')
226
227 print("NEUTRINOS:")
228 indexes_Neutrino_p = np.where((pos_Neutrino_p[:,0]<
    slice_thickness_N))
229 pos_slide_Neutrino_p = pos_Neutrino_p[indexes_Neutrino_p]
230 print('%.3f␣<␣X␣<␣%.3f'%(np.min(pos_slide_Neutrino_p[:,0]), np.
    max(pos_slide_Neutrino_p[:,0])))
231 print('%.3f␣<␣Y␣<␣%.3f'%(np.min(pos_slide_Neutrino_p[:,1]), np.
    max(pos_slide_Neutrino_p[:,1])))
232 print('%.3f␣<␣Z␣<␣%.3f'%(np.min(pos_slide_Neutrino_p[:,2]), np.
    max(pos_slide_Neutrino_p[:,2])))
233
234 # filter out halo positions for the slice
235 indexes_halo_p_N = np.where(halo_pos_p[:,0] < slice_thickness_N)
236 pos_slide_halo_p_N = halo_pos_p[indexes_halo_p_N]
237 indexes_halo_p_CDM = np.where(halo_pos_p[:,0] < slice_thickness_N
    )
238 pos_slide_halo_p_CDM = halo_pos_p[indexes_halo_p_CDM]
239
```

```
indexes_halo_pp_N = np.where(halo_pos_pp[:,0] < slice_thickness_N
    )
pos_slide_halo_pp_N = halo_pos_pp[indexes_halo_pp_N]
indexes_halo_pp_CDM = np.where(halo_pos_pp[:,0] <
    slice_thickness_N)
pos_slide_halo_pp_CDM = halo_pos_pp[indexes_halo_pp_CDM]

indexes_halo_ppp_N = np.where(halo_pos_ppp[:,0] <
    slice_thickness_N)
pos_slide_halo_ppp_N = halo_pos_ppp[indexes_halo_ppp_N]
indexes_halo_ppp_CDM = np.where(halo_pos_ppp[:,0] <
    slice_thickness_N)
pos_slide_halo_ppp_CDM = halo_pos_ppp[indexes_halo_ppp_CDM]

# scale the size of the halo point by mass
area_p_N = halo_mass_p[indexes_halo_p_N]/5e13
area_p_CDM = halo_mass_p[indexes_halo_p_CDM]/5e13

area_pp_N = halo_mass_pp[indexes_halo_pp_N]/5e13
area_pp_CDM = halo_mass_pp[indexes_halo_pp_CDM]/5e13

area_ppp_N = halo_mass_ppp[indexes_halo_ppp_N]/5e13
area_ppp_CDM = halo_mass_ppp[indexes_halo_ppp_CDM]/5e13

# set up voxel sizes for each particle type
L_CDM = BoxSize/grid_CDM
L_N = BoxSize/grid_N

##################################Draw Density Fields
    ####################################
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
from pylab import *


fig = plt.figure(figsize=(12, 16))

# Create layout of plots
mosaic = [
    ["CDM_p",   "Nu_p"],
    ["CDM_pp",  "Nu_pp"],
    ["CDM_ppp", "Nu_ppp"]
]

ax = fig.subplot_mosaic(mosaic, empty_sentinel=None)
fig.subplots_adjust(wspace=0.35, hspace=0.35)

# create function to organise each mosaic panel (one for cdm and
    one for neutrinos)
```

```
283 def plot_field_cdm(ax_handle, data, grid, L, halo, mass):
284
285     # plot data
286     im = ax_handle.imshow(data.T, vmin=-1, vmax=5, cmap='plasma',
        origin='lower')
287     ax_handle.scatter(halo[:,1]/L, halo[:,2]/L, c='g', s=mass)
288
289     # set up colorbars
290     divider = make_axes_locatable(ax_handle)
291     cax = divider.append_axes("right", size="5%", pad="2%")
292     fig.colorbar(im, cax=cax, label='\u03C1␣/␣\u03C1\u0304')
293
294     # relable ticks from voxel co-ords to Mpc/h
295     ticks = np.linspace(0, grid, 3)
296     tick_labels = [f"{i␣*␣L:.1f}" for i in ticks]
297
298     ax_handle.set_xticks(ticks)
299     ax_handle.set_yticks(ticks)
300     ax_handle.set_xticklabels(tick_labels)
301     ax_handle.set_yticklabels(tick_labels)
302
303     ax_handle.set_xlabel("Y␣(Mpc/h)")
304     ax_handle.set_ylabel("Z␣(Mpc/h)")
305
306     return im
307
308 def plot_field_n(ax_handle, data, grid, L, halo, mass):
309
310     im = ax_handle.imshow(data.T, cmap='plasma', origin='lower')
311     ax_handle.scatter(halo[:,1]/L, halo[:,2]/L, c='g', s=mass)
312
313     # set up colorbars
314     divider = make_axes_locatable(ax_handle)
315     cax = divider.append_axes("right", size="5%", pad="2%")
316     fig.colorbar(im, cax=cax, label='\u03C1␣/␣\u03C1\u0304')
317
318     ticks = np.linspace(0, grid, 3)
319     tick_labels = [f"{i␣*␣L:.1f}" for i in ticks]
320
321     ax_handle.set_xticks(ticks)
322     ax_handle.set_yticks(ticks)
323     ax_handle.set_xticklabels(tick_labels)
324     ax_handle.set_yticklabels(tick_labels)
325
326     ax_handle.set_xlabel("Y␣(Mpc/h)")
327     ax_handle.set_ylabel("Z␣(Mpc/h)")
328
329     return im
330
```

```
#Mn_p
plot_field_cdm(ax["CDM_p"], mean_density_CDM_p, grid_CDM, L_CDM,
   pos_slide_halo_p_CDM, area_p_CDM)
plot_field_n(ax["Nu_p"], mean_density_Neutrino_p, grid_N, L_N,
   pos_slide_halo_p_N, area_p_CDM)

#Mn_pp
plot_field_cdm(ax["CDM_pp"], mean_density_CDM_pp, grid_CDM, L_CDM
   , pos_slide_halo_pp_CDM, area_pp_CDM)
plot_field_n(ax["Nu_pp"], mean_density_Neutrino_pp,  grid_N, L_N,
    pos_slide_halo_pp_N, area_pp_N)

#Mn_ppp
plot_field_cdm(ax["CDM_ppp"], mean_density_CDM_ppp, grid_CDM,
   L_CDM, pos_slide_halo_ppp_CDM, area_ppp_CDM)
plot_field_n(ax["Nu_ppp"], mean_density_Neutrino_ppp, grid_N, L_N
   , pos_slide_halo_ppp_N, area_ppp_N)

plt.show()
```

A.4 Appendix D

```
import readgadget
import numpy as np
import MAS_library as MASL
import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable
from pylab import *
from scipy.ndimage import rotate

# input files
snapshot = '/Users/finlaysime/Desktop/Senior␣Honour␣Project/Mn_p/
   snapdir_004/snap_004'
print("snapshot␣files␣imported")

#[1](CDM), [2](neutrinos) or [1,2](CDM+neutrinos)

ptype_CDM = [1]
ptype_Neutrino = [2]

# 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_CDM = readgadget.read_block(snapshot, "POS ", ptype_CDM)/1e3
   #positions in Mpc/h
vel_CDM = readgadget.read_block(snapshot, "VEL ", ptype_CDM)
   #peculiar velocities in km/s
ids_CDM = readgadget.read_block(snapshot, "ID  ", ptype_CDM)-1

pos_Neutrino = readgadget.read_block(snapshot, "POS ",
   ptype_Neutrino)/1e3
vel_Neutrino = readgadget.read_block(snapshot, "VEL ",
   ptype_Neutrino)
ids_Neutrino = readgadget.read_block(snapshot, "ID  ",
   ptype_Neutrino)-1

print("Positions & velocities acquired")

##############################To get the halo catalog
   ######################################

f_catalog = '/Users/finlaysime/Desktop/Senior Honour Project/
   halos/1/out_4_pid.list'

# read the halo catalog
data = np.loadtxt(f_catalog)
print("Halo data imported")

pid  = data[:,41]
idx = pid == -1  #ids of the halos
data_halo = data[idx]

#Units: Masses in Msun / h
#Units: Positions in Mpc / h (comoving)
#Units: Velocities in km / s (physical, peculiar)

# retrieve halo properties
halo_pos = data_halo[:,8:11]
halo_vel = data_halo[:,11:14]
halo_mass = data_halo[:,2]

print("Halo positions, velocites, & masses acquired")

halo_filter = 1e14 #Solar Masses
 # Filter out halos with less mass than 1e14 solar masses
idx = np.where(halo_mass > halo_filter)
halo_mass = halo_mass[idx]
```

```
halo_pos = halo_pos[idx]
halo_vel = halo_vel[idx]

print(f"Halos with mass less than {halo_filter} solar masses removed!")

############################Build density field ####################################

Grid = 200 #int(input("Grid size: ")) #density field with have grid^3 voxels
MAS = 'CIC' #Mass-assignment scheme:'NGP', 'CIC', 'TSC', 'PCS'
do_RSD = False # Dont do redshift-space distortions
axis = 0 #axis along which place RSD; not used here
verbose = True #whether to print information about progress

#Print some information
print('BoxSize: %.3f Mpc/h'%BoxSize)
print('Redshift: %.3f'%redshift)


# OVERDENSITY FIELD #

# compute the effective number of particles/mass in each voxel
delta_pos_Neutrino = MASL.density_field_gadget(snapshot, ptype_Neutrino, Grid, MAS, do_RSD, axis, verbose)
delta_pos_CDM = MASL.density_field_gadget(snapshot, ptype_CDM, Grid, MAS, do_RSD, axis, verbose)

# compute the density contrast
delta_pos_Neutrino /= np.mean(delta_pos_Neutrino, dtype=np.float64); delta_pos_Neutrino -= 1
delta_pos_CDM /= np.mean(delta_pos_CDM, dtype=np.float64); delta_pos_CDM -= 1


# Define voxel size
L = BoxSize / Grid

################################Draw Density Fields ####################################

# prepare stacked arrays
stacked_n_pos = np.zeros((Grid, Grid))
stacked_cdm_pos = np.zeros((Grid, Grid))
stacked_h_pos_x = []
stacked_h_pos_y = []
num = 0
num_vel = 0
```

```
108
109 # Number of slices
110 print(f"Voxel size in Neutrinos is {L}Mpc/h")
111
112 L = L.astype(int)
113
114 # Loop through each 'slice' of the box, the slice will be one
    voxel thick.  Each iteration will
115 # include the slice along each dimension-direction and then
    within one j-iteration a loop is formed for
116 # each dimension.  Within this inner loop the halos are then
    centred and stacked into a single point.
117
118
119 for j in range(Grid):
120     #Take the first 5 components along the first axis and compute
        the mean value
121
122     # Along X axis
123     mean_density_Neutrino_1 = np.mean(delta_pos_Neutrino[j:j+L
        ,:,:],axis=0)
124     #mean_vel_Neutrino_1 = np.mean(delta_vel[j:j+L:,:,:],axis=0)
125
126     mean_density_CDM_1 = np.mean(delta_pos_CDM[j:j+L,:,:],axis=0)
127
128     # Along Y axis
129     mean_density_Neutrino_2 = np.mean(delta_pos_Neutrino[:,j:j+L
        ,:],axis=1)
130     #mean_vel_Neutrino_2 = np.mean(delta_vel[:,j:j+L,:],axis=1)
131
132     mean_density_CDM_2 = np.mean(delta_pos_CDM[:,j:j+L,:],axis=1)
133
134     # Along Z axis
135     mean_density_Neutrino_3 = np.mean(delta_pos_Neutrino[:,:,j:j+
        L],axis=2)
136     #mean_vel_Neutrino_3 = np.mean(delta_vel[:,:,j:j+L],axis=2)
137
138     mean_density_CDM_3 = np.mean(delta_pos_CDM[:,:,j:j+L],axis=2)
139
140
141     # Halo slice
142     l_min = j*L
143     l_max = (j+1)*L
144
145     indexes_halo_1 = np.where((halo_pos[:,0] >= l_min) & (
        halo_pos[:,0] <= l_max))
146     pos_slide_halo_1 = halo_pos[indexes_halo_1]
147     vel_slide_halo_1 = halo_vel[indexes_halo_1]
148
```

```
149     indexes_halo_2 = np.where((halo_pos[:,1] >= l_min) & (
            halo_pos[:,1] <= l_max))
150     pos_slide_halo_2 = halo_pos[indexes_halo_2]
151     vel_slide_halo_2 = halo_vel[indexes_halo_2]
152 
153     indexes_halo_3 = np.where((halo_pos[:,2] >= l_min) & (
            halo_pos[:,2] <= l_max))
154     pos_slide_halo_3 = halo_pos[indexes_halo_3]
155     vel_slide_halo_3 = halo_vel[indexes_halo_3]
156 
157     # Get the angle between the halo velocity to the positive x-
            axis
158     angles_1 = np.degrees(np.arctan2(vel_slide_halo_1[:,1],
            vel_slide_halo_1[:,2]))
159     angles_1 = angles_1 % 360 # To wrap the angle back around the
             clockwise direction
160 
161     angles_2 = np.degrees(np.arctan2(vel_slide_halo_2[:,0],
            vel_slide_halo_2[:,2]))
162     angles_2 = angles_2 % 360
163 
164     angles_3 = np.degrees(np.arctan2(vel_slide_halo_3[:,0],
            vel_slide_halo_3[:,1]))
165     angles_3 = angles_3 % 360
166 
167 
168     # Loop over every halo in the slide, extracting its position
            as a reference
169     # ALONG X-AXIS
170     for i, (hx, hy, hz) in enumerate(pos_slide_halo_1):
171         num += 1
172         # change to voxel co-ords
173         cx = int(np.floor(hy/L)%Grid)
174         cy = int(np.floor(hz/L)%Grid)
175 
176         # density field shift of neutrinos to centre halo
177         shifted_density_field_n = np.roll(np.roll(
                mean_density_Neutrino_1.T, shift=-cx+Grid//2, axis=0),
178                                           shift=-cy+Grid//2, axis
                                              =1)
179 
180         shifted_density_field_cdm = np.roll(np.roll(
                mean_density_CDM_1.T, shift=-cx+Grid//2, axis=0),
181                                             shift=-cy+Grid//2,
                                                axis=1)
182 
183 
184         # rotate the neutrino density field with rotate function
                from scipy
```

```
185         shifted_density_field_n = rotate(shifted_density_field_n,
                angles_1[i], axes=(1,0), reshape=False)
186         shifted_density_field_cdm = rotate(
               shifted_density_field_cdm, angles_1[i], axes=(1,0),
               reshape=False)
187
188         # Form a rotation matrix
189         angle_rad = np.deg2rad(angles_1[i])
190
191         R = np.array([
192     [ np.cos(angle_rad), -np.sin(angle_rad)],
193     [ np.sin(angle_rad),  np.cos(angle_rad)]
194     ])
195
196         stacked_n_pos += shifted_density_field_n
197
198         stacked_cdm_pos += shifted_density_field_cdm
199
200         # Extract Y,Z positions relative to the current halo
               center
201         pos_yz = pos_slide_halo_1[:, [1,2]] - np.array([hy, hz])
202
203         # Apply rotation in  Y Z  plane
204         pos_rotated_yz = pos_yz @ R
205
206         # Reinsert X coordinate unchanged
207         h_pos_rotate = np.copy(pos_slide_halo_1)
208         h_pos_rotate[:, [1,2]] = pos_rotated_yz + np.array([hy,
               hz])
209
210         # shift halo position such that it is in the centre and
               halos that are moved off
211         # one edge are located appropriately on the other side
212         hy_shift = (h_pos_rotate[:,1]-hy+BoxSize/2)%BoxSize
213         hz_shift = (h_pos_rotate[:,2]-hz+BoxSize/2)%BoxSize
214         hy_shift = hy_shift//L
215         hz_shift = hz_shift//L
216
217         stacked_h_pos_x.append(hy_shift[i])
218         stacked_h_pos_y.append(hz_shift[i])
219
220         print(f"Stacked␣image␣{num}", end='\r')
221
222
223     # ALONG Y-AXIS
224     for k, (hx, hy, hz) in enumerate(pos_slide_halo_2):
225         num += 1
226         # change to voxel co-ords
227         cx = int(np.floor(hx/L)%Grid)
```

```
228         cy = int(np.floor(hz/L)%Grid)
229
230         # density and velocity field shift of neutrinos to centre
                halo
231         shifted_density_field_n = np.roll(np.roll(
               mean_density_Neutrino_2.T, shift=-cx+Grid//2, axis=0),
232                                           shift=-cy+Grid//2, axis
                                              =1)
233
234         shifted_density_field_cdm = np.roll(np.roll(
               mean_density_CDM_2.T, shift=-cx+Grid//2, axis=0),
235                                           shift=-cy+Grid//2, axis
                                              =1)
236
237
238         # rotate the neutrino density and velocity fields with
               rotate function from scipy
239         shifted_density_field_n = rotate(shifted_density_field_n,
                angles_2[k], axes=(1,0), reshape=False)
240         shifted_density_field_cdm = rotate(
               shifted_density_field_cdm, angles_2[k], axes=(1,0),
               reshape=False)
241
242
243         # Form a rotation matrix
244         angle_rad = np.deg2rad(angles_2[k])
245
246         R = np.array([
247     [ np.cos(angle_rad), -np.sin(angle_rad)],
248     [ np.sin(angle_rad),  np.cos(angle_rad)]
249     ])
250
251
252         stacked_n_pos += shifted_density_field_n
253         stacked_cdm_pos += shifted_density_field_cdm
254
255         # Extract X,Z positions relative to the current halo
               center
256         pos_xz = pos_slide_halo_2[:, [0,2]] - np.array([hx, hz])
257
258         # Apply rotation in  X Z  plane
259         pos_rotated_xz = pos_xz @ R
260
261         # Reinsert Y coordinate unchanged
262         h_pos_rotate = np.copy(pos_slide_halo_2)
263         h_pos_rotate[:, [0,2]] = pos_rotated_xz + np.array([hx,
               hz])
264
```

```
265         # shift halo position such that it is in the centre and
               halos that are moved off
266         # one edge are located appropriately on the other side
267         hx_shift = (h_pos_rotate[:,0]-hx+BoxSize/2)%BoxSize
268         hz_shift = (h_pos_rotate[:,2]-hz+BoxSize/2)%BoxSize
269         hx_shift = hx_shift//L
270         hz_shift = hz_shift//L
271
272         stacked_h_pos_x.append(hx_shift[k])
273         stacked_h_pos_y.append(hz_shift[k])
274
275         print(f"Stacked␣image␣{num}", end='\r')
276
277
278     # ALONG Z-AXIS
279     for q, (hx, hy, hz) in enumerate(pos_slide_halo_3):
280         num += 1
281         # change to voxel co-ords
282         cx = int(np.floor(hx/L)%Grid)
283         cy = int(np.floor(hy/L)%Grid)
284
285         # density field shift of neutrinos to centre halo
286         shifted_density_field_n = np.roll(np.roll(
               mean_density_Neutrino_3.T, shift=-cx+Grid//2, axis=0),
287                                           shift=-cy+Grid//2, axis
                                              =1)
288
289         shifted_density_field_cdm = np.roll(np.roll(
               mean_density_CDM_3.T, shift=-cx+Grid//2, axis=0),
290                                           shift=-cy+Grid//2, axis
                                              =1)
291
292         # rotate the neutrino density field with rotate function
               from scipy
293         shifted_density_field_n = rotate(shifted_density_field_n,
                angles_3[q], axes=(1,0), reshape=False)
294         shifted_density_field_cdm = rotate(
               shifted_density_field_cdm, angles_3[q], axes=(1,0),
               reshape=False)
295
296         # Form a rotation matrix
297         angle_rad = np.deg2rad(angles_3[q])
298
299         R = np.array([
300     [ np.cos(angle_rad), -np.sin(angle_rad)],
301     [ np.sin(angle_rad),  np.cos(angle_rad)]
302     ])
303
304
```

```
305         stacked_n_pos += shifted_density_field_n
306         stacked_cdm_pos += shifted_density_field_cdm
307 
308         # Extract X,Y positions relative to the current halo
               center
309         pos_xy = pos_slide_halo_3[:, [0,1]] - np.array([hx, hy])
310 
311         # Apply rotation in  X Y  plane
312         pos_rotated_xy = pos_xy @ R
313 
314         # Reinsert Z coordinate unchanged
315         h_pos_rotate = np.copy(pos_slide_halo_3)
316         h_pos_rotate[:, [0,1]] = pos_rotated_xy + np.array([hx,
               hy])
317 
318         # shift halo position such that it is in the centre and
               halos that are moved off
319         # one edge are located appropriately on the other side
320         hx_shift = (h_pos_rotate[:,0]-hx+BoxSize/2)%BoxSize
321         hy_shift = (h_pos_rotate[:,1]-hy+BoxSize/2)%BoxSize
322         hx_shift = hx_shift//L
323         hy_shift = hy_shift//L
324 
325         stacked_h_pos_x.append(hx_shift[q])
326         stacked_h_pos_y.append(hy_shift[q])
327 
328         print(f"Stacked␣image␣{num}", end='\r')
329 
330 
331 # Compute the average so not just stacked
332 stacked_n_pos /= num
333 stacked_cdm_pos /= num
334 
335 
336 print('')
337 print("Finished␣plotting!")
338 
339 
340 # optimised factor according to chi-squared
341 f_opt = np.sum(stacked_n_pos * stacked_cdm_pos) / np.sum(
       stacked_n_pos ** 2)
342 
343 print("Optimal␣factor␣=", f_opt)
344 
345 dipole_field = (stacked_n_pos * f_opt - stacked_cdm_pos)
346 
347 fig, ax, = plt.subplots(figsize=(8,8))
348 im= ax.imshow(stacked_n_pos, cmap='plasma', origin='lower')
349 ax_divider = make_axes_locatable(ax)
```

```
cax = ax_divider.append_axes("right", size="5%", pad="2%")
cb = fig.colorbar(im, cax=cax, label='\u03C1 / \u03C1\u0304')
ax.scatter(stacked_h_pos_x, stacked_h_pos_y, c='g', s=1)
ax.set_xlabel("Mpc/h"); ax.set_ylabel("Mpc/h")
ax.set_title(f"Neutrino Overdensity Field, voxel size: {L} Mpc/h,
    Mn_p")
ticks = np.linspace(0, Grid, 10)
tick_labels = [f"{i * L:.1f}" for i in ticks]  # Convert to Mpc/h
ax.set_xticks(ticks); ax.set_xticklabels(tick_labels)
ax.set_yticks(ticks); ax.set_yticklabels(tick_labels)
plt.show()
```

A.5 Appendix E

The chi-squared value for the optimum factor is determined by minimising the chi-squared value. This was done by taking the derivative of the equation

$$\chi^2 = \sum_x (f\delta_\nu(x) - \delta_{CDM}(x)) \tag{22}$$

with respect to f and setting the equation as equal to zero; such that

$$\frac{d\chi^2}{df} = 2\sum_x ((f\delta_\nu(x) - \delta_{CDM})\delta_\nu(x)) = 0 \tag{23}$$

It is then easy to see

$$f = \frac{\sum_x \delta_{CDM}(x)\delta_\nu(x)}{\sum_x \delta_\nu^2(x)} \tag{24}$$

A.6 Appendix F

M_ν	f_{opt}
M_ν^+	1.14
M_ν^{++}	1.68
M_ν^{+++}	2.14

Table 1: Presented here are the optimal factors for each of the neutrino masses used in equation 14 and derived using the method described in appendix A.5