Scientific project report

Andrei Znobishchev

03 November 2016

Abstract

The following is a report for the Department of General and Applied Physics of "Moscow Institute of Physics and Technology" on Andrei Znobishchev's latest results from his scientific project. This work is a primary part of Bachelor's Diploma project.
Scientific advisor: Sergey Pilipenko. Created in LaTeX.

1 Introduction

Current observations of great number of galaxies in space the size of hundreds of megaparsecs show that the distribution of galaxies is not uniform (Fig.1 and Fig.2).

This heterogeneity is a result of the growth of small fluctuations, which occurred due to the gravitational instability in the early Universe. There is a theory that describes the process of this growth. It is also possible to simulate it using computational methods. However, how can we describe the observable distribution quantitatively?
One of basic quantities is a mass function n(M) of cosmic structures, defined by

$$dN = n(M)dM, \tag{1}$$

dN is a number of objects per unit volume with mass grater than M, but less than M + dM.

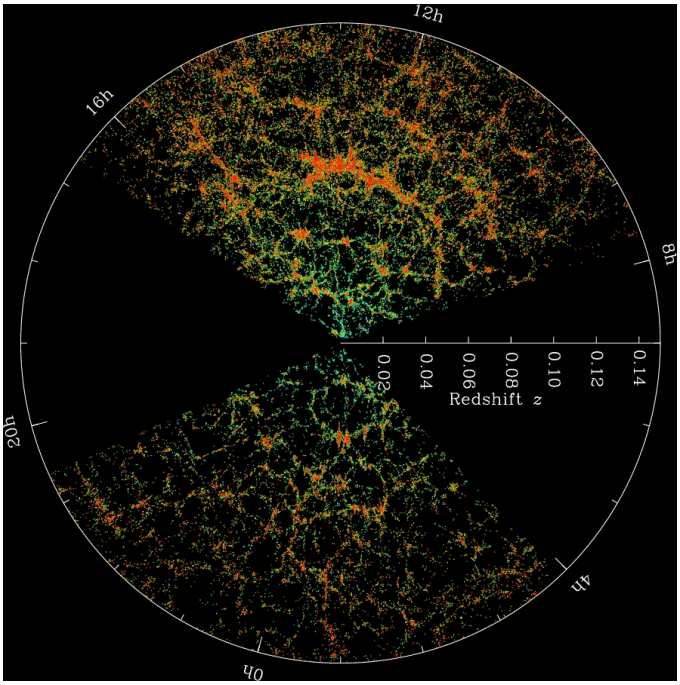

Figure 1: A map of the Universe from SDSS where the distance to galaxies is given in terms of their redshift. Credit: SDSS

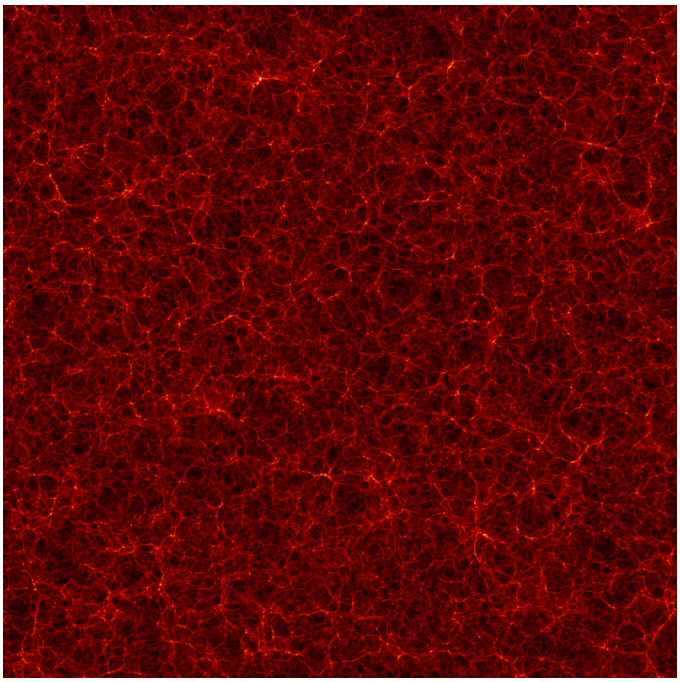

Figure 2: Slice through the MultiDark simulation (MDR1) at redshift z = 0.53

From an observational point of view, it is clear how to find the mass function. One needs to count objects of a given mass in a selected volume of space. However, it is practical to have an analytical formula for the mass function. And the mathematical expression was derived by Press and Schechter in 1974. The following is commonly known as the Press-Schechter formalizm.

Let δ(x,t) be the density fluctuation field (later I will omit the time dependence). Another parameter is R: we will not take into account information about the behaviour of δ(x) on scales smaller than R. The filter for this is a "window" function W(x - x'):

$$\delta(x,R) = \int d^3x \delta(x') W_R(x - x'). \tag{2}$$

There might be different choices for a window function. We will use the *top hat function*:

$$\begin{aligned} W(x-x') &= 1, \ |x-x'| < R, \\ &= 0, \ |x-x'| \geq R \end{aligned} \tag{3}$$

The sphere of radius R contains a mass

$$M = \frac{4\pi}{3}\rho_0 R^3, \tag{4}$$

As there is a direct relation between R and M, we can call $\delta(x,R) = \delta(x,M) \equiv \delta_M$. The key point of Press and Schechter's work is the assumption that the density field has a Gaussian probability distribution:

$$P(\delta_M)d\delta_M = \frac{1}{\sqrt{2}\pi\sigma_M} \exp(-\frac{\delta_M^2}{2\sigma_M^2})d\delta_M, \tag{5}$$

σ_M is the variance of the density field that was filtered on a scale R enclosing a mass M. Let δ_c be some critical density value that will be defined later. The probability that δ_M exceeds δ_c is given by:

$$P_{>\delta_c}(M) = \int_{\delta_c}^{\infty} P(\delta_M)d\delta_M. \tag{6}$$

The probability (6) depends on the filter mass M and on the redshift z, because the variance

$$\sigma_M^2 = \sigma_R^2 = \frac{1}{2\pi^2}\int_0^{\infty} dk k^2 P_M(k,z)\tilde{W}_R^2(k), \tag{7}$$

P_M(k,z) is the matter power spectrum and $\tilde{W}_R^2$(k) is the Fourier transform of the top hat filter function.

Finally, the mass function n(M) should be defined as

$$\begin{aligned} n(M)dM &= \frac{\rho_0}{M}[P_{>\delta_c}(M) - P_{>\delta_c}(M+dM)] \\ &= -\frac{\rho_0}{M}\frac{dP_{>\delta_c}}{dM}dM \\ &= -\frac{\rho_0}{M}\frac{dP_{>\delta_c}}{d\sigma_M}\frac{d\sigma_M}{dM}dM. \end{aligned} \tag{8}$$

Unfortunately, this formula is not absolutely correct, as it does not take into account underdense regions (those with $\delta < 0$. Another assumption of Press and Schechter was that underdense regions will accrete onto overdensities and they added a factor of 2 to the formula (8).

Combining (5)-(8) we have

$$n(M)dM = -\sqrt{\frac{2}{\pi}}\frac{d\sigma_M}{dM}\frac{\rho_0\delta_c}{M\sigma_M^2}\exp(-\frac{\delta_c^2}{2\sigma_M^2})dM \tag{9}$$

2 Calculation of mass function

The standard procedure is to evaluate the mass function (9) by calculating the variance (7) using the matter power spectrum from the linear theory.

In cosmology it is possible to calculate the matter power spectrum theoretically. It is stated that the initial matter power spectrum $P_i(k) \propto k$ (here "initial" refers to the period with quantum fluctuations before inflation). Then this spectrum changes. Firstly, some modes go beyond the horizon during inflation and then fit the limits again during "normal" expansion. Secondly, the spectrum is affected by the pressure of matter: the pressure blurs perturbations on some scales. All these processes can be described by the "transfer" function T(k) (Mo and others, §4.3; *Eisenstein&Hu*, 1997). The final spectrum is P(k) $= P_i(k) \cdot T^2$(k). The matter power spectrum can be measured quite accurately on some scales using the cosmic microwave background. The standard matter power spectrum is given in the Figure 3.

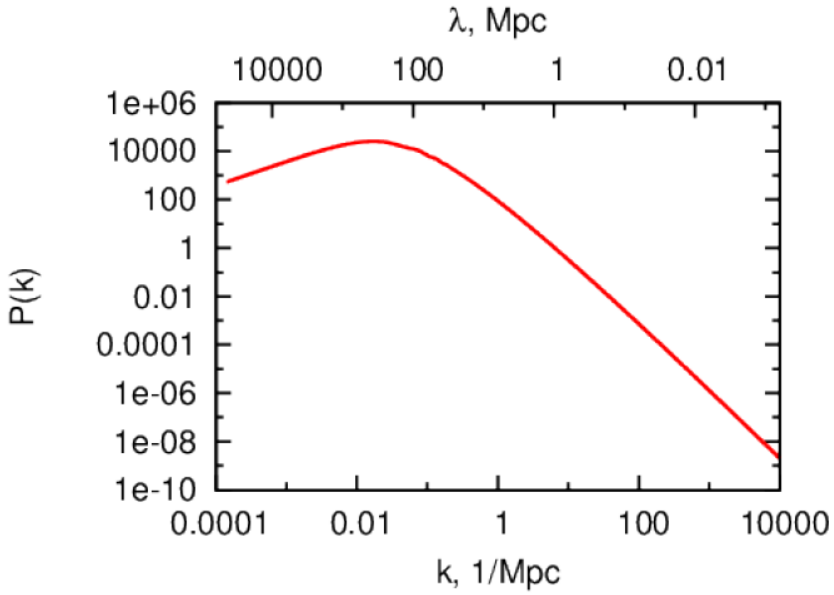

Figure 3: The matter power spectrum

This spectrum is approximated (Bardeen and others, 1986):

$$P(k) = A(T(k))^2 k, \tag{10}$$

$$T(k) = \frac{ln(1+2.34q)}{2.34q}(1+3.89q+(16.1q)^2+(5.64q)^3+(6.71q)^4)^{-\frac{1}{4}}, \tag{11}$$

$q = \frac{k}{\Omega_m h^2}, A \approx 4.735 \cdot 10^6$.
k is measured in h/Mpc, $\Omega_m \approx 0.27, h = \frac{H}{100km/s/Mpc} \approx 0.73$ - dimensionless Hubble parameter.
The Fourier transform of the window function is $\tilde{W}_R(k)$:

$$\tilde{W}_R(k) = \frac{3sin(kR) - 3kRcos(kR)}{(kR)^3} \tag{12}$$

Now it is possible to calculate the variance (7) using equations (10)-(12).

2.1 Approximate integration

First of all, the spectrum (10)-(11) has two asymptotes. $P \propto k$, if k<<1, and $P \propto k^{-3}$, if k>>1.
Moreover, $\tilde{W}_R(k)$ can be approximated:
1) If kR<<1, sin(kR)≈kR and cos(kR)≈ $1-(kR)^2/2$.
$\sigma_M^2 = \frac{1}{2\pi^2}\int_0^\infty dk k^2 P(k)$
2) If kR>>1, 3kRcos(kR)>>3sin(kR).
$\sigma_M^2 = \frac{1}{2\pi^2}\int_0^\infty dk k^2 P(k) 9(kR)^2 cos^2(kR)/(kR)^6 =$ using $cos^2(kR) = (1+cos(2kR))/2$ and Rimann's Theorem about oscillating functions $= \frac{1}{2\pi^2}\int_0^\infty dk k^2 P(k) \frac{9}{2(kR)^4}$

I used both approximations and calculated (7) for two scales: $R > 1/k_0$ and $R < 1/k_0$, where $k_0 \approx \Omega_m h^2$ - the typical k, where P(k) changes its behaviour.

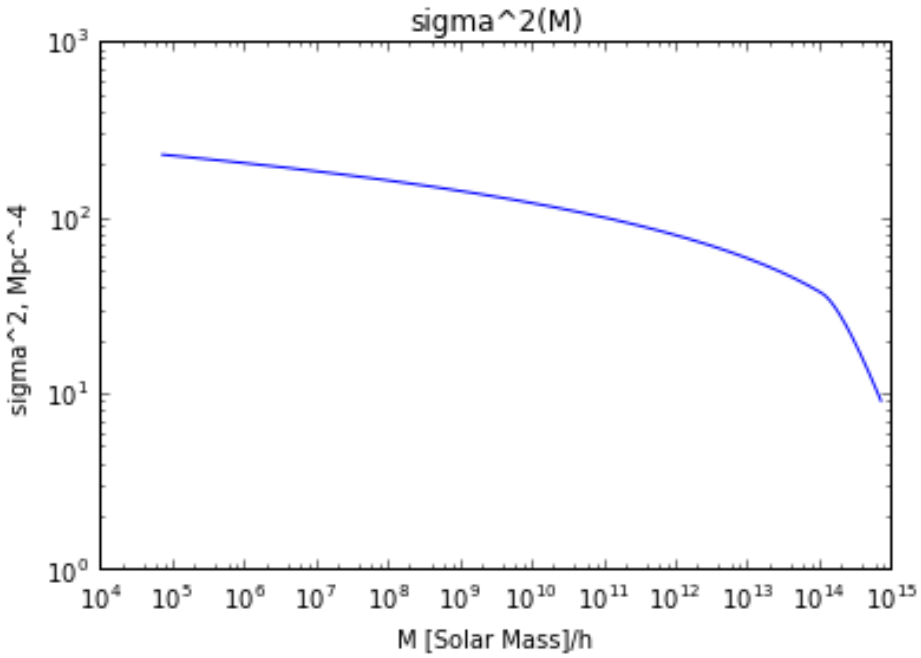

Figure 4: The variance σ_M^2. Approximate integration. Logarithmic scale

The result is a good evaluation of the variance (7), however, the estimated error of the variance (7), when k>> 1, is quite big. That it why it was decided to calculate the same quantity numerically.

2.2 Numerical integration using Python

The calculation was conducted using Python and its libraries "scipy" and "numpy". The results are in the Figures 5 and 6. The following is my code for calculation the variance and the mass function.

```
import numpy as np
import matplotlib.pyplot as plt
from numpy import pi as pi
import scipy.integrate as integrate

A = 4.375e6
Omega = 0.27
h = 0.73
M_S = 1.98e30
Mpc = 3.0856776e22
rho = 0.3*9.31e-27*Mpc**3/h**3
sgm_s=1.686
k_0 = Omega*h**2
g = 4*pi*rho/3

def Rad (M):
    return (M/g)**(1/3)

def P(x):
    k = np.exp(x)
    return A*k*(T(k))**2

def T(x):
    q = x/(Omega*h**2)
    return np.log(1+2.34*q)*(2.34*q)**(-1)*(1+3.89*q+(16.1*q)**2+
    (5.64*q)**3+(6.71*q)**4)**(-1/4)

def W(x,R):
    k = np.exp(x)
    return (3*np.sin(k*R)-3*k*R*np.cos(k*R))/(k*R)**3
    #return 1/(R)**4/k**3

def al(t, M):
    return np.exp(t)*g**(-1/3)*M**(1/3)

def br(t, M):
    a = al(t, M)
    return 3*np.sin(a) - 3*a*np.cos(a)

def sgm2(M):
    R = Rad (M)
    return ( integrate.quad(lambda t:
    np.exp(3*t)*P(t)*(W(t,R))**2/(2*pi**2), np.log(1e-2), np.log(1e5))[0] )

def der_sgm2(M):
    return ( integrate.quad(lambda t:
    np.exp(3*t)*P(t)*2*br(t,M)*(br(t,M)-al(t,M)**2*np.sin(al(t,M)))/
    (al(t,M)**6*M*2*pi**2), np.log(1e-2), np.log(1e5))[0] )
```

```
def dersgm(M):
    return der_sgm2(M)/(2*np.sqrt(sgm2(M)))

def n(M):
    return np.sqrt(2/pi)*rho*dersgm(M)*sgm_s*
    np.exp(-sgm_s**2/(2*sgm2(M)))/(M*sgm2(M))

M_0 = M_S*1e5
M_F = M_S*1e15
x = np.logspace(np.log10(M_0), np.log10(M_F), num = 200)
y = [sgm2(x[i]) for i in range (len(x))]
y1 = [n(x[i]) for i in range (len(x))]
y2 = [y1[i]*x[i] for i in range (len(x))]
plt.title('Mass function')
#plt.title('sigma^2(M)')
plt.xlabel('M [Solar Mass]/h')
plt.ylabel('dN(>M)/dlogM')
#plt.ylabel('sigma^2, Mpc^-4')
plt.xscale('log')
plt.yscale('log')
plt.plot (x*h/M_S, y2)
plt.show()
```

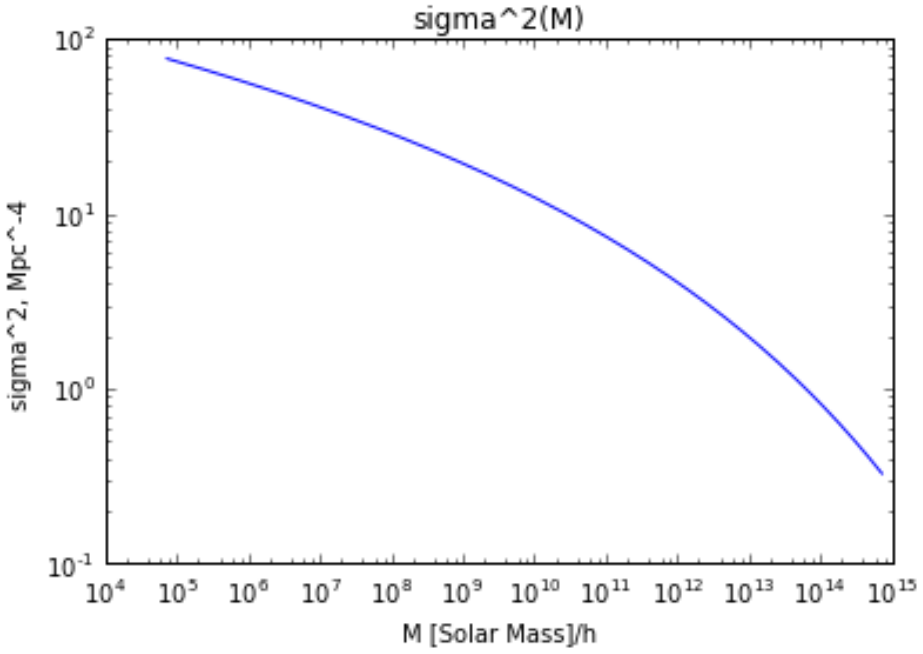

Figure 5: The variance σ_M^2. Numerical integration in Python. Logarithmic scale

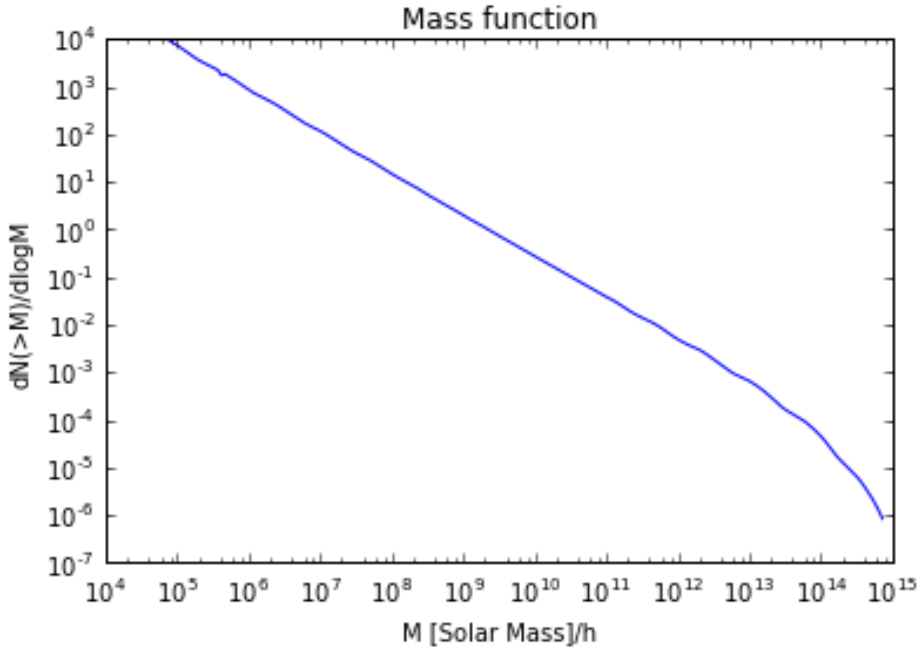

Figure 6: The mass function n(M). Numerical integration in Python. Logarithmic scale

3 Analysis of cosmological simulation

The next step was to check the latest Sergey Pilipenko's simulation. Its size is 32Mpc*32Mpc*32Mpc and it has 512^3 particles. The large number of particles requires using fast algorithms for calculating space density. The space was divided in 100^3 cells and covered with $k \cdot 100^3$ spheres of radius R. k is a natural number. R and k can be changed. The density was calculated in each sphere.
In addition, some parts of the code (especially those with long arrays) were rewritten in C and embedded in Python using "scipy.weave" library and the program calculated density 27 times faster. However, it is still more convenient to use Python in the rest parts of the code, because Python has "pygadgetreader" library for reading the simulation, "numpy" library and other useful functions.

The result is in the Figure 7.

```
from __future__ import division #so that 3/2 = 1.5,
#but not 1 as in Python2.*
import pygadgetreader as pgr
import numpy as np
from numpy import pi
from time import time
from scipy import weave

t0 = time()
coord = pgr.readsnap ('rsnap_002.dat', 'pos', 1)
t1 = time() - t0
maxcoord = 32
numofcells = 100
hx, hy, hz = maxcoord/numofcells, maxcoord/numofcells, maxcoord/numofcells
numofsph = numofcells + 1 # = number of knots
R = 0.999*hx
V = 4/3*pi*R**3

nop = int(np.size(coord)/3) #number of particles
```

```
#nop = int(1e6);

rho = 1/V

xmax, ymax, zmax = 32, 32, 32
xmin, ymin, zmin = 0, 0, 0
mx = int((xmax - xmin)/hx)
my = int((ymax - ymin)/hy)
mz = int((zmax - zmin)/hz)
part = np.zeros((3))
density = np.zeros((numofsph,numofsph,numofsph))

r1 = int(R/hx) + 1
r2 = int(R/hy) + 1
r3 = int(R/hz) + 1

def boundcond ():
    density[100,:,:] = density [0, :, :]
    density[:,100,:] = density [:, 0, :]
    density[:,:,100] = density [:, :, 0]

ccode = \
        """
        part[0] = coord[i*3]; part[1] = coord[i*3+1];
        part[2] = coord[i*3+2];
        """
ccode2 = \
        """
        int index = 0;
        int meff, neff, peff;
        meff = m + i; neff = n + j; peff = p + k;
        if (meff<0) {meff+=numofsph-1;}
        if (neff<0) {neff+=numofsph-1;}
        if (peff<0) {peff+=numofsph-1;}
        if (meff>=numofsph) {meff-=numofsph+1;}
        if (neff>=numofsph) {neff-=numofsph+1;}
        if (peff>=numofsph) {peff-=numofsph+1;}
        index = peff+neff*numofsph+meff*numofsph*numofsph;
        density[index]+=rho;
        """
for i in range (nop):
    weave.inline (ccode, ['i', 'part', 'coord'], compiler = 'gcc')
    #part = coord[i]
    x = part[0]; y = part[1]; z = part[2]
    m = int(x/hx); n = int(y/hy); p = int(z/hz)
    for i in range (-r1+1, r1+1):
        for j in range (-r2+1, r2+1):
            for k in range (-r3+1, r3+1):
                xsp = (m+i)*hx; ysp = (n+j)*hy; zsp = (p+k)*hz
                if ((x-xsp)**2 + (y-ysp)**2 + (z - zsp)**2 <= R**2):
                    weave.inline (ccode2, ['density', 'm', 'i', 'n', 'j', 'p', 'k',
                    'rho', 'numofsph'], compiler = 'gcc')
```

```
boundcond()
t2 = time() - t1 - t0
print ('time for reading = ', t1, 'time for calculating density', t2)
```

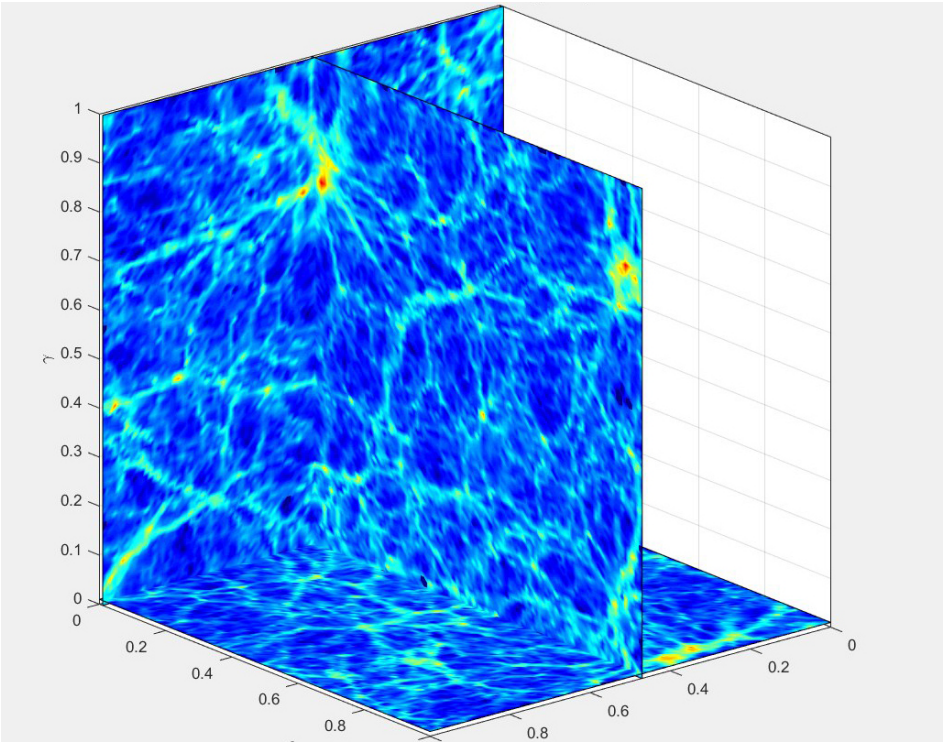

Figure 7: X, Y and Z slices of the calculated density distribution

4 Conclusion and future goals

Now I am analysing this cosmological simulation of Sergey Pilipenko, calculate the density variance for different magnitudes of R and compare it with the calculated numerically (in 2.2). However, the absolute concurrence is not expected. The main reason for this is that the size of the cube in the simulation (32Mpc*32Mpc*32Mpc) is relatively small.
Furthermore, I need to take into account the growth factor, because data is provided for z = 1 (Information about the growth factor - Barkana, Loeb. The first sources of light and the reionization of the universe (PR349, 2001)). There are also some requirements, one of them - periodic boundary conditions. They are already checked. Another one - spheres can not be too small in this method. The radius of each sphere should be big enough, so that the mass inside each sphere is greater than the mass of the biggest halo in this simulation. To check it, there are special programs called 'halo finder' programs. I will be using the one called 'Rockstar HF'.

References

[1] Misner C.W., Thorne K.S., Wheeler J.A., 1973, W.H. Freeman, 1279 pp.

[2] Press W.H., Schechter P., 1974. Astrophys. J. 187, 425.

[3] Mo H., Bosch F., White S., 2010, Cambridge University Press, 840 pp.

[4] Barkana R., Loeb A., 2001, Physics Reports 349, 125-238.

[5] Arkhipova N.A., Komberg B.V., Lukash V.N., Miheeva E.V., 2007, Astronomy J., 84(10), 874-884