A REPORT ON

Dark Matter Power Spectrum Modelling at high redshift

BY

Nikhil P.S. Bisht	2017B5A70610G	(MSc. Hons.) Physics + (B.E. Hons.) Computer Science

AT

Inter University Centre for Astronomy and Astrophysics, Pune

A Practice School-I station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

11th July, 2019

Inter University Centre for Astronomy and Astrophysics, Pune

A Practice School-1 Station of

BIRLA INSTITUTE OF TECHNOLOGY SCIENCE, PILANI

A Project report on

Dark Matter Power Spectrum Modelling at High Redshift

11th July, 2019

Author:
Nikhil P.S. Bisht
2017B5A70610G

Project Mentor:
Dr. Aseem Paranjape

Academic year 2019-2020

Acknowedgment

I would like to express my sincere thanks to Inter University Centre for Astronomy and Astrophysics, Pune and especially Dr. Dipankar Bhattacharya, for providing us with such great projects. I would like to take this opportunity to thank our Practice School Instructor in-charge Dr. Prasant Samantray to be a constant source of motivation and guide us through our journey in this project. I would also like to express my gratitude towards Dr. Aseem Paranjape, the project mentor for his constant guidance and unique way of explaining things and allowing me to explore the subject in its entirety. I would also like to thank Ms. Sujatha Ramakrishnan, for her support during the latter half of the project.
Lastly I would like to thank the PS Division of BITS Pilani, K. K. Birla Goa Campus to have implemented the Practice School 1 Program and giving us the priceless experience at the time we needed the most. Thanks a lot.

Birla Institute of Technology and Science, Pilani

Practice School Division

Station: Inter University Centre for Astronomy and Astrophysics, Pune
Duration: 8 weeks
Date of Start: 22nd May, 2019
Date of Submission: 11th July, 2019
Title of the Project: Dark Matter Power Spectrum Modelling at High Redshift
Name: Nikhil P.S. Bisht
ID No.: 2017B5A70610G
Discipline: (MSc. Hons.) Physics + (B.E. Hons.) Computer Science
Name of Project Mentor: Dr. Aseem Paranjape
Name of PS faculty: Dr. Prasant Samantray
Key Words: Cosmology, Large Scale Structure, Fourier Analysis, Power Spectrum, Zel'dovich Approximation
Project Area: Cosmological Perturbations, Fourier Analysis, N-Body Simulation, Formation of Large Scale Structure, Power Spectrum as an Observable

Abstract:

We look at how the Large Scale Structure of the universe was formed from Cosmological perturbations by first looking at the Eulerian Scheme (both Linear and Non-Linear Perturbation theories) and Lagrangian Schemes (Zel'dovich Approximation). We also take a look at Fourier Analysis and how to use it to solve differential equations. Finally, we look at simulations of a large set of collision-less particles submit to gravitational attraction in 1D, 2D and 3D and observe and try to analyse their evolution.

Contents

List of Figures 5

1 Introduction 6

2 Eulerian Dynamics 7

2.1 The Vlasov Equation 7

2.2 Eulerian Linear Perturbation Theory 9

2.3 Fourier Analysis 11

2.4 Eulerian Non-Linear Perturbation Theory 13

3 Lagrangian Dynamics 15

4 Conclusion 24

References 25

Appendices 26

A Discrete Fourier Transform codes 26

B Fast Fourier Transform code 28

C Zel'dovich Simulation Codes 29

D Zel'dovich 2-D timing analysis 35

E Zel'dovich Phase Space Simulation codes 37

List of Figures

1 Time Taken (s) Vs $\sqrt{number\ of\ particles}$ 16
2 Time Taken (s) vs Number of particles 17
3 $y = nlog_2(n)$ 17
4 Evolution of 50 particles with Sinusoidal Density contrast (1D) 18
5 Evolution of phase space ($\dot{x}$ vs x) (1D) 18
6 Evolution of 2500 particles with Sinusoidal Density contrast (2D) 19
7 Evolution of phase space ($\dot{x}$ vs (x,y) and ($\dot{y}$ vs (x,y)) (2D) 20
8 Evolution of 8000 particles with Sinusoidal Density contrast (3D) 21
9 Evolution of phase space (Quiver Plot) 22
10 Evolution of 8000 particles with Gaussian Density contrast (3D) 23
11 Evolution of phase space (Quiver Plot) 23

1 Introduction

The Universe, a 93 billion ly diameter and a 13.8 billion years old entity, in which whatever we know and will know exists, is full of mysteries. Whatever we observe is just 4.6 % of the entire universe, the rest is termed Dark Matter (24 %) and Dark Energy (71.4 %).
There have been several theories as to what Dark Matter is and how it interacts with other matter. Based on observations of motion of stars around Milky Way's (and other galaxies') Centre, scientists more or less figured out how it is distributed around them and hypothesized that it interacts only through gravity.
There have been several models fitting the observations like the λCDM model, etc. and various types of dark matter like Cold, Warm, Hot, Ballistic Dark Matter hypothesized. To check the validity of these models and rule out incorrect ones, a new observable is needed. This came in the form of Power Spectrum obtained from the Large Scale Structure.
As one zooms out of Earth, keeps on zooming out, they will first see our Galaxy (on a scale of 10^5 ly), then the Local Group (order of 10^8 ly), then the Virgo and then Laniakea Supercluster (order of 10^{10} ly). At this scale (greater than 100 Mpc) one sees superclusters, walls, filaments like structure formed by millions and millions of galaxies, termed as Large Scale Structures.
The most natural explanation of these structures is that they are a result of amplification of small initial fluctuations due to gravitational interaction of Collisionless Cold Dark Matter particles in the expanding universe.
This project aims at simulating these fluctuations and essentially working on how to use power spectrum to constrain Dark Matter Energy Density in the universe.

2 Eulerian Dynamics

2.1 The Vlasov Equation

For a large set of particles of mass m which interact only gravitationally, one can write the equation of motion like:

$$\frac{d\overrightarrow{v}}{dt} = -\frac{\partial\varphi}{\partial\overrightarrow{r}} \tag{1}$$

where $\overrightarrow{v}$ is the velocity of a particle at position $\overrightarrow{r}$ and φ is the potential induced by the local density field $\rho(\overrightarrow{r})$ defined as:

$$\varphi\left(\overrightarrow{r}\right) = G\int d^3\overrightarrow{r}\,\frac{\rho(\overrightarrow{r})}{|\overrightarrow{r}\,' - \overrightarrow{r}|} \tag{2}$$

To understand gravitational instabilities, one first changes to the appropriate coordinate system: comoving coordinates ($\overrightarrow{x}$) which are coordinates invariant to the expansion of the universe connected to particle positions as $\overrightarrow{r} = a(\tau)\overrightarrow{x}$ and conformal time τ related to cosmic time as $dt = a(\tau)d\tau$ where $a(\tau)$ is the cosmological scale factor. One also defines the conformal expansion rate $\mathcal{H} := dln(a)/d\tau = Ha = \dot{a}$. Now, we write the Friedmann equations, which are the equations of motion valid in an arbitrary homogeneous and isotropic background Universe:

$$\frac{\partial\mathcal{H}\left(\tau\right)}{\partial\tau} = \left(\Omega_\Lambda\left(\tau\right) - \frac{\Omega_m\left(\tau\right)}{2}\right)\mathcal{H}^2\left(\tau\right) \quad ; \quad \left(\Omega_{tot}\left(\tau\right) - 1\right)\mathcal{H}^2\left(\tau\right) = k \tag{3}$$

where Ω_m is the ratio of matter density to critical density, Ω_Λ is the the ratio between the energy density due to the cosmological constant and the critical density, Ω_{tot} is $\Omega_m + \Omega_\Lambda$ and k = -1,0,1 for Ω_{tot} less than, equal to or greater than 1 (corresponding to Hyperbolic, Flat or Spherical geometry of the Universe respectively).

We now look at deviations from smooth Hubble flow by introducing the density contrast $\delta(\overrightarrow{x})$ and peculiar velocity $\overrightarrow{u})$ as:

$$\rho\left(\overrightarrow{x},\tau\right) := \bar{\rho}\left(\tau\right)\left[1 + \delta(\overrightarrow{x},\tau)\right] \quad ; \quad \overrightarrow{v}\left(\overrightarrow{x},\tau\right) := \mathcal{H}\overrightarrow{x} + \overrightarrow{u}\left(\overrightarrow{x},\tau\right) \tag{4}$$

and the cosmological gravitational potential Φ with:

$$\varphi\left(\overrightarrow{x},\tau\right) := -\frac{1}{2}\frac{\partial\mathcal{H}}{\partial\tau}x^2 + \Phi\left(\overrightarrow{x},\tau\right) \tag{5}$$

so that the Poisson equation reads:

$$\nabla^2\Phi\left(\overrightarrow{x},\tau\right) = \frac{3}{2}\Omega_m(\tau)\mathcal{H}^2(\tau)\delta\left(\overrightarrow{x},\tau\right) \tag{6}$$

Using Eq. (1) and writing $\overrightarrow{p} = am\overrightarrow{u}$, we can write:

$$\frac{d\overrightarrow{p}}{d\tau} = -am\nabla\Phi(\overrightarrow{x}) \tag{7}$$

Let us now define the particle number density in phase space by $f(\overrightarrow{x},\overrightarrow{p},\tau)$, applying phase space conservation:
$\frac{df}{d\tau} = \frac{\partial f}{\partial\tau} + \frac{\partial\overrightarrow{x}}{\partial\tau}\cdot\nabla f + \frac{\partial\overrightarrow{p}}{\partial\tau}\cdot\frac{\partial f}{\partial\overrightarrow{p}} = 0$
which gives the Vlasov Equation:

$$\frac{\partial f}{\partial\tau} + \frac{\overrightarrow{p}}{ma}\cdot\nabla f - am\nabla\Phi\cdot\frac{\partial f}{\partial\overrightarrow{p}} = 0 \tag{8}$$

The Vlasov equation, being a non-linear partial differential equation involving seven variables, is very difficult to solve, thus we solve for evolution of spatial distribution rather than full phase-space dynamics.
We first take zeroth, first and second order momentum moments of the distribution function:

$$\int d^3\overrightarrow{p}f\left(\overrightarrow{x},\overrightarrow{p},\tau\right) := \rho\left(\overrightarrow{x},\tau\right) \tag{9}$$

$$\int d^3\overrightarrow{p}\frac{\overrightarrow{p}}{am}f\left(\overrightarrow{x},\overrightarrow{p},\tau\right) := \rho\left(\overrightarrow{x},\tau\right)u\left(\overrightarrow{x},\tau\right) \tag{10}$$

$$\int d^3\overrightarrow{p}\frac{p_ip_j}{a^2m^2}f\left(\overrightarrow{x},\overrightarrow{p},\tau\right) := \rho\left(\overrightarrow{x},\tau\right)u_i\left(\overrightarrow{x},\tau\right)u_j\left(\overrightarrow{x},\tau\right) + \sigma_{ij}\left(\overrightarrow{x},\tau\right) \tag{11}$$

where $\sigma_{ij}\left(\overrightarrow{x},\tau\right)$ is the stress tensor, i.e. the equation of state of the cosmological fluid. We now derive continuity equation by taking the zeroth moment of the Vlasov Equation (Eq. (8)) and using the above equations:

$$\int d^3\overrightarrow{p}\left(\frac{\partial f}{\partial\tau} + \frac{\overrightarrow{p}}{ma}\cdot\nabla f - am\nabla\Phi\cdot\frac{\partial f}{\partial\overrightarrow{p}}\right) = 0$$

The third term goes to 0, and we end up with the continuity equation which describes the conservation of mass:

$$\frac{\partial\delta\left(\overrightarrow{x},\tau\right)}{\partial\tau} + \nabla\cdot\left\{\left[1+\delta\left(\overrightarrow{x},\tau\right)\right]\overrightarrow{u}\left(\overrightarrow{x},\tau\right)\right\} = 0 \tag{12}$$

Now we take the first moment of the Vlasov Equation:

$$\int d^3\vec{p}\,\frac{\vec{p}}{am}\left(\frac{\partial f}{\partial\tau}+\frac{\vec{p}}{ma}\cdot\nabla f-am\nabla\Phi\cdot\frac{\partial f}{\partial\vec{p}}\right)=0$$

After solving, we obtain the Euler Equation:

$$\frac{\partial\vec{u_i}\,(\vec{x},\tau)}{\partial\tau}+h\,(\tau)\,\vec{u_i}\,(\vec{x},\tau)+\vec{u_j}\,(\vec{x},\tau)\cdot\nabla_j\vec{u_i}\,(\vec{x},\tau)=-\nabla_i\varphi(x,\tau)-\frac{1}{\rho}\nabla_j(\rho\sigma_{ij}) \quad (13)$$

Since the stress tensor characterizes deviation of particle motion from a single stream flow, it is a good approximation to set $\sigma_{ij}\approx 0$, since the dominant term is the first one, at least initially when structures did not have time to collapse and virialize.

2.2 Eulerian Linear Perturbation Theory

The Universe is assumed to be smooth at large scales, thus we linearize the previous set of equation by introducing the divergence of the velocity field, $\theta(\vec{x},\tau):=\nabla\cdot\vec{u}(\vec{x},\tau)$ and the vorticity, $\vec{w}(\vec{x},\tau):=\nabla\times\vec{u}(\vec{x},\tau)$:

$$\frac{\partial\delta(\vec{x},\tau)}{\partial\tau}+\theta\,(\vec{x},\tau)=0 \quad (14)$$

$$\frac{\partial\vec{u}\,(\vec{x},\tau)}{\partial\tau}+\mathcal{H}\,(\tau)\,\vec{u}\,(\vec{x},\tau)=-\nabla\Phi\,(\vec{x},\tau) \quad (15)$$

Taking divergence and curl of Eq. (15):

$$\frac{\partial\theta\,(\vec{x},\tau)}{\partial\tau}+H\,(\tau)\,\theta\,(\vec{x},\tau)+\frac{3}{2}\Omega_m(\tau)\mathcal{H}^2(\tau)\delta\,(\vec{x},\tau)=0 \quad (16)$$

$$\frac{\partial\vec{w}\,(\vec{x},\tau)}{\partial\tau}+H\,(\tau)\,\vec{w}\,(\vec{x},\tau)=0 \quad (17)$$

Thus, from Eq. (17) the vorticity falls away inversely proportional to the scale factor ($\vec{w}\,(\tau)\propto a^{-1}$) in the linear case. So, we can safely assume any initial vorticity to decay away.
We can write a separable solution to the density contrast as $\delta(\vec{x},\tau)=D_1(\tau)\delta(\vec{x},0)$ where $D_1(\tau)$ is the Linear Growth Factor. Taking time derivative of Eq. (16) and putting in Eq. (14), we get:

$$\frac{\mathrm{d}^2D_1(\tau)}{\mathrm{d}\tau^2}+\mathcal{H}(\tau)\frac{\mathrm{d}D_1(\tau)}{\mathrm{d}\tau}=\frac{3}{2}\Omega_m(\tau)\mathcal{H}^2(\tau)D_1(\tau) \quad (18)$$

We now denote its two solutions as the fastest growing mode ($D_1^{(+)}(\tau)$ and the slowest growing mode ($D_1^{(+)}(\tau)$). Naturally, the density contrast will be given by:

$$\delta(\overrightarrow{x},\tau) = D_1^{(+)}(\tau)A(\overrightarrow{x}) + D_1^{(-)}(\tau)B(\overrightarrow{x}) \tag{19}$$

where $A(\overrightarrow{x})$ and $B(\overrightarrow{x})$ describe the initial density field. Thus, by using Eq. (14), we get:

$$\theta(\overrightarrow{x},\tau) = -\mathcal{H}(\tau)\left[f(\Omega_m,\Omega_\Lambda)\,A(\overrightarrow{x}) + g\,(\Omega_m,\Omega_\Lambda)\,B(\overrightarrow{x})\right] \tag{20}$$

where the functions f and g are given by:

$$f(\Omega_m,\Omega_\Lambda) := \frac{\mathrm{d}\ln D_1^{(+)}}{\mathrm{d}\ln a} = \frac{1}{\mathcal{H}}\frac{\mathrm{d}\ln D_1^{(+)}}{\mathrm{d}\tau} \quad g\,(\Omega_m,\Omega_\Lambda) := \frac{1}{\mathcal{H}}\frac{\mathrm{d}\ln D_1^{(-)}}{\mathrm{d}\tau} \tag{21}$$

There are three most important cases:

1. $\Omega_m = 1$ and $\Omega_\Lambda = 0$ (Einstein-de Sitter Universe):

$$D_1^{(+)} = a, \quad D_1^{(-)} = a^{-3/2}, \quad f(1,0) = 1 \tag{22}$$

2. $\Omega_m < 1$ and $\Omega_\Lambda = 0$:

$$D_1^{(+)} = 1 + \frac{3}{x} + 3\sqrt{\frac{1+x}{x^3}}\ln[\sqrt{1+x} - \sqrt{x}] \quad D_1^{(-)} = \sqrt{\frac{1+x}{x^3}} \tag{23}$$

   where $x := 1/\Omega_m - 1$ and the f is given by $f(\Omega_m, 0) \approx \Omega_m^{3/5}$. One sees that as Ω_m goes to 0, the perturbations cease to grow.

3. For a general case, one approximates the value to be:

$$D_1^{(+)} \approx \left(\frac{5}{2}\right)\frac{a\Omega_m}{\Omega_m^{4/7} - \Omega_A + (1+\Omega_m/2)\,(1+\Omega_\Lambda/70)} \tag{24}$$

$$D_1^{(-)} = \frac{\mathcal{H}}{a} \tag{25}$$

$$f(\Omega_m,\Omega_\Lambda) \approx \frac{1}{\left[1 - \left(\Omega_0 + \Omega_\Lambda^0 - 1\right)a + \Omega_\Lambda^0 a^3\right]^{0.6}} \tag{26}$$

   where $\Omega_\Lambda^0 \equiv \Omega_\Lambda(a=1)$.

2.3 Fourier Analysis

Before we proceed further, we need to understand how Fourier Analysis works as it is a very important and useful tool in solving differential equations.

Fourier Integral Transform:

Given a function $f(x)$ and its Fourier transform $\tilde{f}(k)$, the integral relation between them is:

$$f(x) = \int_{-\infty}^{\infty} \frac{dk}{2\pi} e^{ikx} \tilde{f}(k) \tag{27}$$

$$\tilde{f}(k) = \int_{-\infty}^{\infty} dx e^{-ikx} f(x) \tag{28}$$

Some important results are:

$$\int_{-\infty}^{\infty} dk e^{ik(x-x')} = 2\pi\delta_D\left(x - x'\right) \tag{29}$$

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2} \longleftrightarrow e^{-k^2\sigma^2/2} \tag{30}$$

$$\frac{1}{x} \longleftrightarrow i\pi(k) \tag{31}$$

$$\theta(x) = \frac{1}{2} + \frac{1}{2}\,\mathrm{sgn}(x) \longleftrightarrow \pi\delta_D(k) - i/k \tag{32}$$

$$\tilde{f}^{(n)}(k) = (ik)^n \tilde{f}(k) \tag{33}$$

$$\nabla f(\vec{x}) \to \widetilde{\nabla f}(\vec{k}) = (i\vec{k})\tilde{f}(\vec{k}) \tag{34}$$

$$\nabla^2 f(\vec{x}) \to \widetilde{\nabla^2 f}(\vec{k}) = -k^2 \tilde{f}(\vec{k}) \tag{35}$$

Discrete Fourier Transform:

The Discrete Fourier transform pairs $\tilde{f}_m$ and f_n are related as:

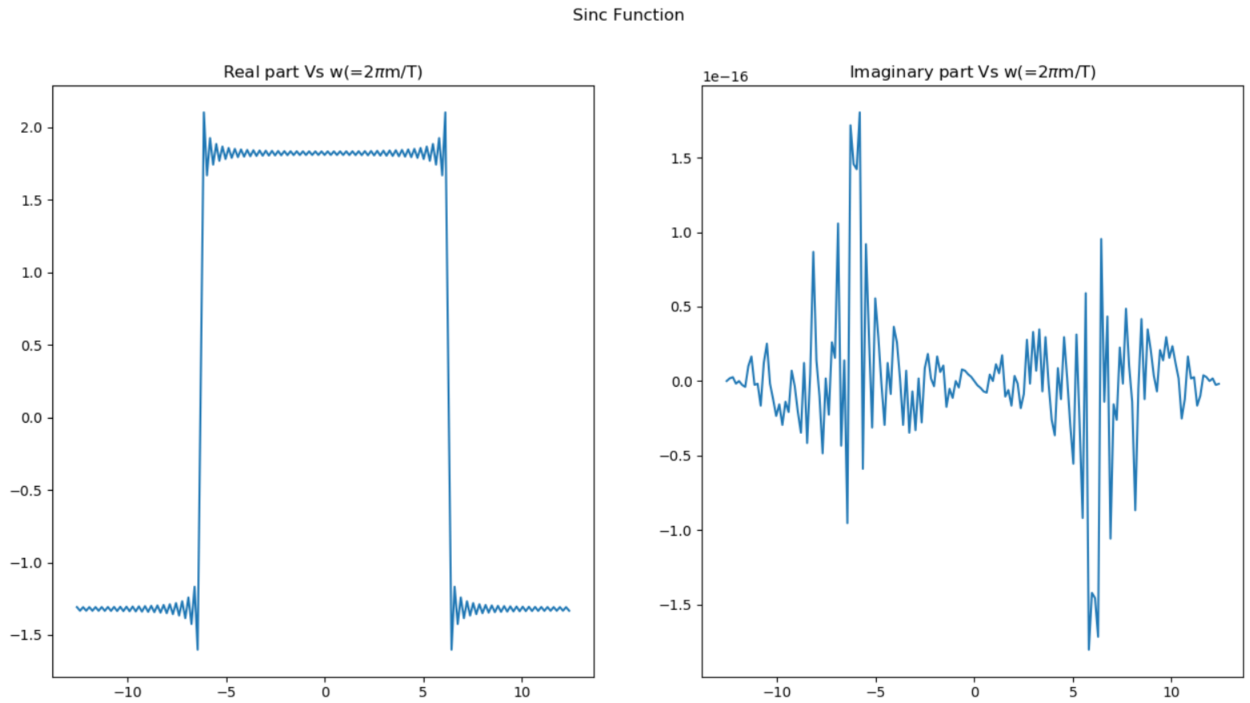
$$\tilde{f}_m = \frac{1}{N} \sum_{n=0}^{N-1} f_n e^{-2\pi imn/N} \tag{36}$$

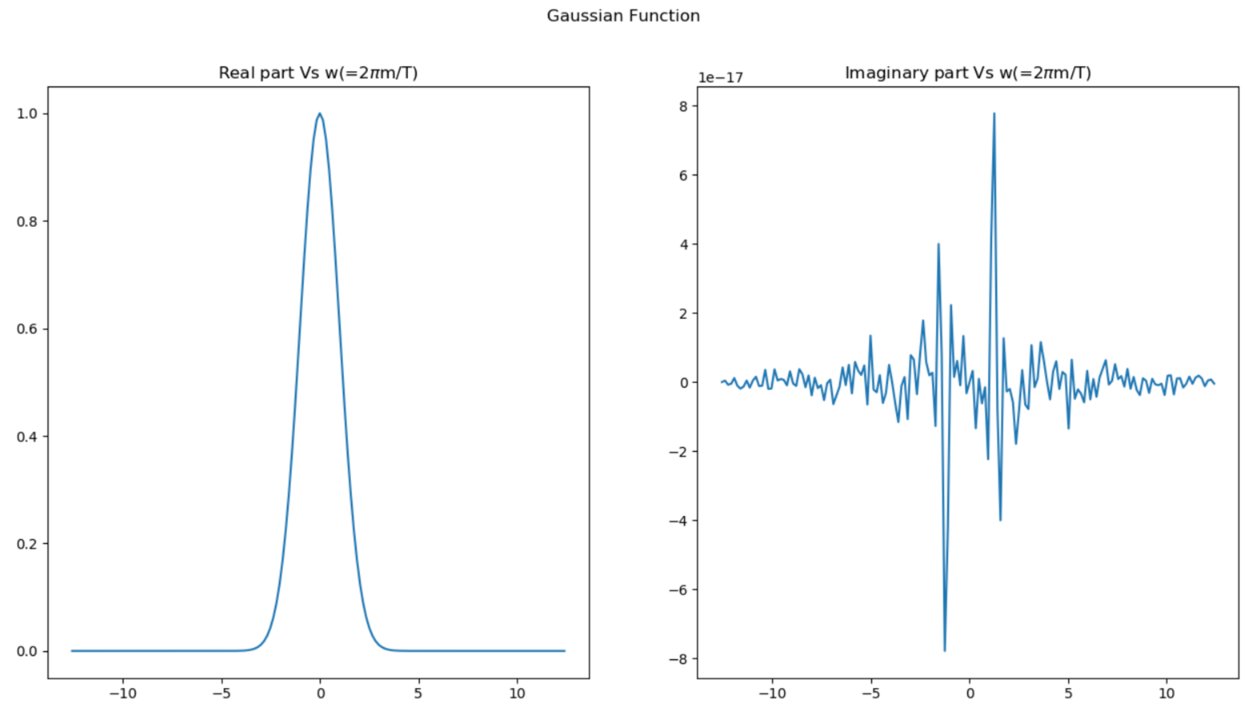
and

$$f_n = \sum_{m=0}^{N-1} \tilde{f}_m e^{2\pi imn/N} \tag{37}$$

where $f_n := f(n\Delta t)$.

The results of Fourier Transforming a sinc function and the Gaussian are as follows (see Appendices A and B):





Computationally, the faster way to do a Fourier transform is called the Fast Fourier Transform (FFT). Scipy and Numpy both have inbuilt modules to calculate fourier transforms using FFT. The change in complexity is from $O(N^2)$ for DFT to O(NlogN) for FFT. We get the same results as above, only faster.

2.4 Eulerian Non-Linear Perturbation Theory

Now that we are familiar with fourier transforms, we can study the Non-Linear perturbation theory. To proceed further, we will make an assumption that any initial vorticity vanishes [1]. Now, we make another assumption that it is possible to expand the density and velocity fields about the linear solutions:

$$\delta(\vec{x}, t) = \sum_{n=1}^{\infty} \delta^{(n)}(\vec{x}, t), \quad \theta(\vec{x}, t) = \sum_{n=1}^{\infty} \theta^{(n)}(\vec{x}, t) \tag{38}$$

where $\delta^{(1)}$ and $\theta^{(1)}$ are linear in the initial density field, $\delta^{(2)}$ and $\theta^{(2)}$ are quadratic in the initial density field, etc.

Now, we take the Fourier Transform of Eq. (12):

$$\frac{\partial\tilde{\delta}(\vec{k}, \tau)}{\partial\tau} + \tilde{\theta}(\vec{k}, \tau) = -\int \mathrm{d}^3\vec{k}_1 \mathrm{d}^3\vec{k}_2 \delta_D\left(\vec{k} - \vec{k}_{12}\right) \alpha\left(\vec{k}_1, \vec{k}_2\right) \tilde{\theta}\left(\vec{k}_1, \tau\right) \tilde{\delta}\left(\vec{k}_2, \tau\right) \tag{39}$$

and the divergence then Fourier Transform of Eq. (13):

$$\begin{aligned}\frac{\partial\tilde{\theta}(\vec{k},\tau)}{\partial\tau} + \mathcal{H}(\tau)\tilde{\theta}(\vec{k}, \tau) + \tfrac{3}{2}\Omega_m\mathcal{H}^2(\tau)\tilde{\delta}(\vec{k}, \tau) = -\int \mathrm{d}^3\vec{k}_1 \mathrm{d}^3\vec{k}_2 \delta_D\left(\vec{k} - \vec{k}_{12}\right) \\ \times\beta\left(\vec{k}_1, \vec{k}_2\right) \tilde{\theta}\left(\vec{k}_1, \tau\right) \tilde{\theta}\left(\vec{k}_2, \tau\right)\end{aligned} \tag{40}$$

where:

$$\alpha\left(\vec{k}_1, \vec{k}_2\right) \equiv \frac{\vec{k}_{12} \cdot \vec{k}_1}{k_1^2}, \quad \beta\left(\vec{k}_1, \vec{k}_2\right) \equiv \frac{k_{12}^2\left(\vec{k}_1 \cdot \vec{k}_2\right)}{2k_1^2k_2^2}$$

For Einstein-de Sitter Cosmology, the solutions are given by [2]:

$$\delta_n(\mathrm{k}) = \int \mathrm{d}^3\mathrm{q}_1 \ldots \int \mathrm{d}^3\vec{q}_n \delta_D\left(\mathrm{k} - \mathrm{q}_{1\ldots n}\right) F_n\left(\mathrm{q}_1, \ldots, \mathrm{q}_n\right) \delta_1\left(\mathrm{q}_1\right) \ldots \delta_1\left(\mathrm{q}_n\right) \tag{41}$$

$$\theta_n(\vec{k}) = \int \mathrm{d}^3\vec{q}_1 \ldots \int \mathrm{d}^3\vec{q}_n \delta_D\left(\vec{k} - \vec{q}_{1\ldots n}\right) G_n\left(\vec{q}_1, \ldots, \vec{q}_n\right) \delta_1\left(\vec{q}_1\right) \ldots \delta_1\left(\vec{q}_n\right) \tag{42}$$

where the kernels F_n and G_n are given by:

$$
\begin{aligned}
F_n\left(\vec{q}_1, \ldots, \vec{q}_n\right) = & \sum_{m=1}^{n-1} \frac{G_m\left(\vec{q}_1, \ldots, \vec{q}_m\right)}{(2n+3)(n-1)}\left[(2n+1)\alpha\left(\vec{k}_1, \vec{k}_2\right) F_{n-m}\left(\vec{q}_{m+1}, \ldots, \vec{q}_n\right)\right. \\
& \left.+2\beta\left(\vec{k}_1, \vec{k}_2\right) G_{n-m}\left(\vec{q}_{m+1}, \ldots, \vec{q}_n\right)\right] \\
G_n\left(\vec{q}_1, \ldots, \vec{q}_n\right) = & \sum_{m=1}^{n-1} \frac{G_m\left(\vec{q}_1, \ldots, \vec{q}_m\right)}{(2n+3)(n-1)}\left[3\alpha\left(\vec{k}_1, \vec{k}_2\right) F_{n-m}\left(\vec{q}_{m+1}, \ldots, \vec{q}_n\right)\right. \\
& \left.+2n\beta\left(\vec{k}_1, \vec{k}_2\right) G_{n-m}\left(\vec{q}_{m+1}, \ldots, \vec{q}_n\right)\right]
\end{aligned}
\tag{43}
$$

where $\mathrm{k}_1 \equiv \mathrm{q}_1 + \ldots + \mathrm{q}_m$, $\mathrm{k}_2 \equiv \mathrm{q}_{m+1} + \ldots + \mathrm{q}_n$, $\mathrm{k} \equiv \mathrm{k}_1 + \mathrm{k}_2$, and $F_1 = G_1 \equiv 1$
One also uses these to derive recursion relation for vertices ν_n and μ_n which correspond to the spherical average of the PT kernels:

$$
\begin{aligned}
\nu_n & \equiv n! \int \tfrac{\mathrm{d}\Omega_1}{4\pi} \ldots \tfrac{\mathrm{d}\Omega_n}{4\pi} F_n\left(\vec{k}_1, \ldots, \vec{k}_n\right) \\
\mu_n & \equiv n! \int \tfrac{\mathrm{d}\Omega_1}{4\pi} \ldots \tfrac{\mathrm{d}\Omega_n}{4\pi} G_n\left(\vec{k}_1, \ldots, \vec{k}_n\right)
\end{aligned}
\tag{44}
$$

Thus, we get the relations as:

$$
\begin{aligned}
\nu_n & = \textstyle\sum_{m=1}^{n-1} \begin{pmatrix} n \\ m \end{pmatrix} \tfrac{\mu_m}{(2n+3)(n-1)} \left[(2n+1)\nu_{n-m} + \tfrac{2}{3}\mu_{n-m}\right] \\
\mu_n & = \textstyle\sum_{m=1}^{n-1} \begin{pmatrix} n \\ m \end{pmatrix} \tfrac{\mu_m}{(2n+3)(n-1)} \left[3\nu_{n-m} + \tfrac{2}{3}n\mu_{n-m}\right]
\end{aligned}
\tag{45}
$$

3 Lagrangian Dynamics

Now, we will do the transformation of going from Eulerian to Lagrangian scheme. The difference is that in Eulerian, we dealt with density and velocity fields and their equations of motion whereas in Lagrangian, we will be dealing with trajectories of particles or fluid elements. Our interest will be the Displacement field $\vec{\psi}(\vec{q})$ which maps the initial particle positions to Eulerian particle positions :

$$\vec{x} = \vec{q} + \vec{\psi} \tag{46}$$

Thus, the Equations of motion will be:

$$\frac{\mathrm{d}^2 \mathrm{x}}{\mathrm{d}\tau^2} + \mathcal{H}(\tau)\frac{\mathrm{dx}}{\mathrm{d}\tau} = -\nabla\Phi \tag{47}$$

where the Φ is the gravitational potential and gradient is with respect to Eulerian coordinates We can take divergence of this equation to get:

$$J(\vec{q},\tau)\nabla \cdot \left[\frac{\mathrm{d}^2\psi}{\mathrm{d}\tau^2} + \mathcal{H}(\tau)\frac{\mathrm{d}\psi}{\mathrm{d}\tau}\right] = \frac{3}{2}\Omega_m \mathcal{H}^2(J-1) \tag{48}$$

where we have:

$$1 + \delta(\mathrm{x}) = \frac{1}{\mathrm{Det}\left(\delta_{ij} + \Psi_{i,j}\right)} \equiv \frac{1}{J(\mathrm{q},\tau)} \tag{49}$$

We can convert this equation into fully Lagrangian coordinates, which will be non-linear in $\vec{\psi}(\vec{q})$ and can be expanded about its linear solution. The linear solution is:

$$\nabla_q \cdot \vec{\psi}^{(1)} = -D(\tau)\delta(\vec{q}) \tag{50}$$

The Zel'dovich Approximation consists in using the linear displacement field as an approximate solution for the dynamical equations:

$$1 + \delta(\vec{x},\tau) = \frac{1}{[1-\lambda_1 D_1(\tau)]\,[1-\lambda_2 D_1(\tau)]\,[1-\lambda_3 D_1(\tau)]} \tag{51}$$

where λ_i are the local eigenvalues of the tidal tensor $\psi_{i,j}$.
We can now use Eq. (46) and Eq. (50) to simulate a large set of particles according to a specified density contrast by determining $\vec{\psi}$.
First we use the fact that the vorticity of the system is 0 by rewriting Eq. (46):

$$\vec{\psi} = \nabla \phi \tag{52}$$

$$\nabla^2 \phi = -D(\tau)\delta(\vec{q}) \tag{53}$$

where ϕ is a scalar field. Then, we Fourier transform the equation:

$$\tilde{\phi} = D(\tau)\frac{\tilde{\delta}(\vec{k})}{\vec{k}^2} \tag{54}$$

Thus, the Fourier transform of the original field will be:

$$\tilde{\vec{\psi}} = D(\tau)\frac{\tilde{\delta}(\vec{k})}{\vec{k}^2} \cdot i\vec{k} \tag{55}$$

We can inverse Fourier Transform to get the displacement field:

$$\vec{\psi} = D(\tau) IFT \left[\frac{\tilde{\delta}(\vec{k})}{\vec{k}^2} \cdot i\vec{k} \right] \tag{56}$$

Plugging this back in Eq. (46) and iterating as τ changes will simulate a large set of particles. Before we see the results of the simulation, we need to do a time analysis of the algorithm (see Appendix D).

First, we plot the time taken for the code to run iterated 100 times (scaled linearly) vs the square root of the number of particles:

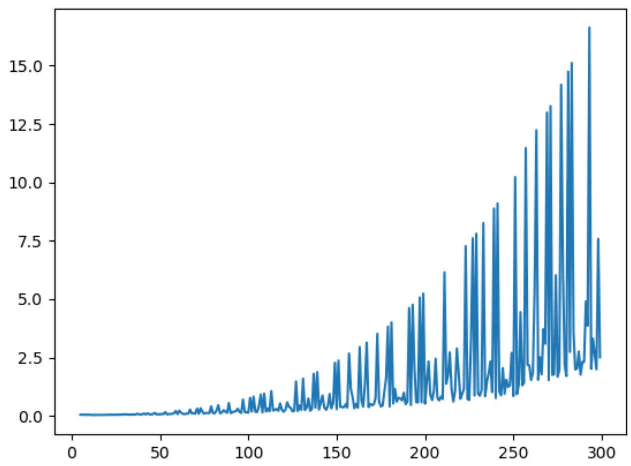


Figure 1: Time Taken (s) Vs $\sqrt{number\ of\ particles}$

Now we plot the average time taken for the code to run once vs the number of particles (increased as powers of 2) and compare with a (linearly scaled) plot of $y = nlog_2(n)$

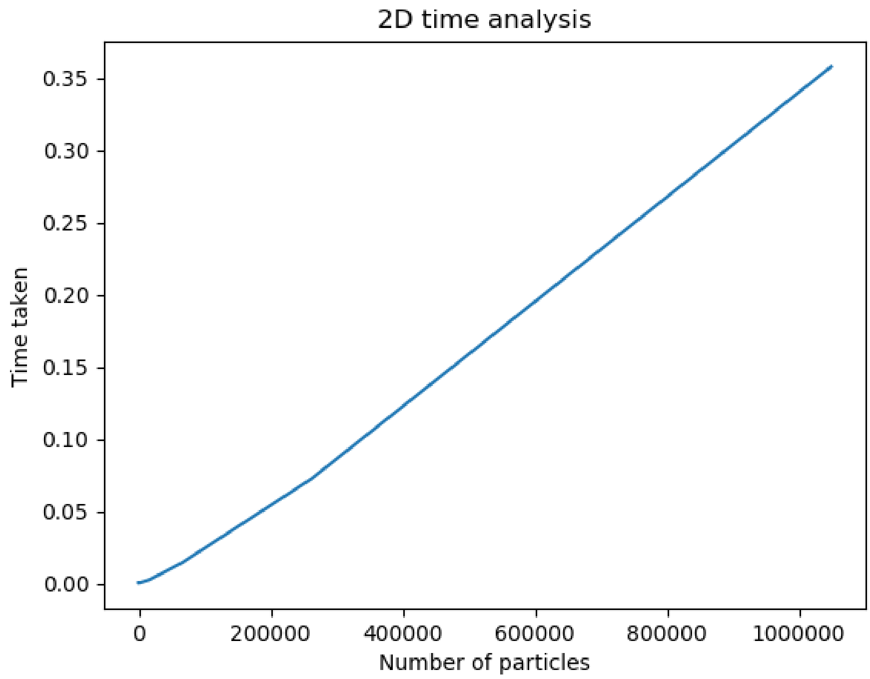


Figure 2: Time Taken (s) vs Number of particles

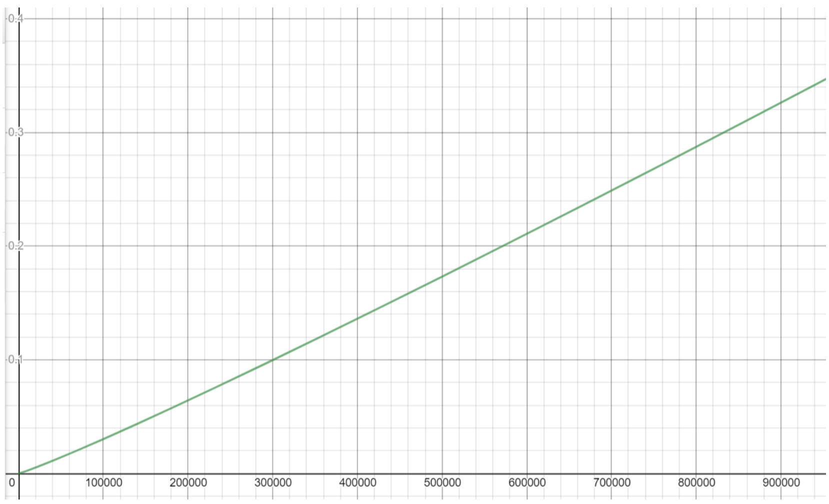


Figure 3: $y = nlog_2(n)$

So, the algorithm has the same complexity (O(nlogn)) as FFT (Fast Fourier Transform). Now, we will look at the result of the simulations of the particle positions in 1D, 2D and 3D and their phase spaces (2D, 4D and 6D respectively) under Sinusoidal and Gaussian density contrasts.First up are the 1D Plots:

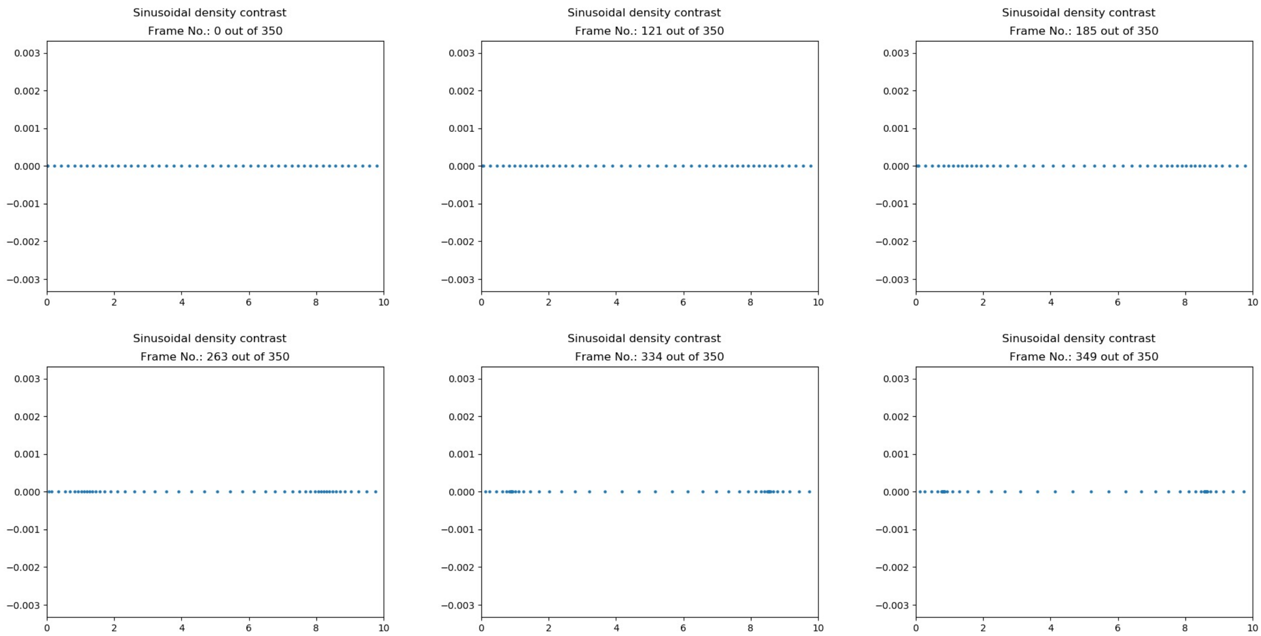


Figure 4: Evolution of 50 particles with Sinusoidal Density contrast (1D)

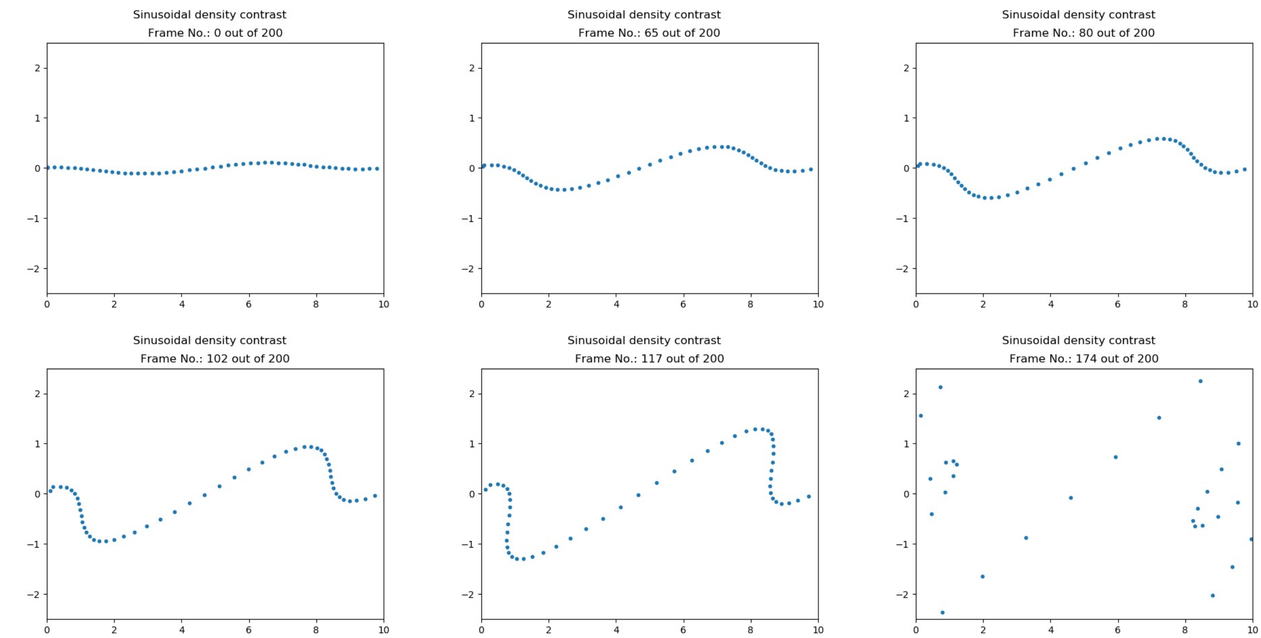


Figure 5: Evolution of phase space ($\dot{x}$ vs x) (1D)

Next we have the 2D plots:

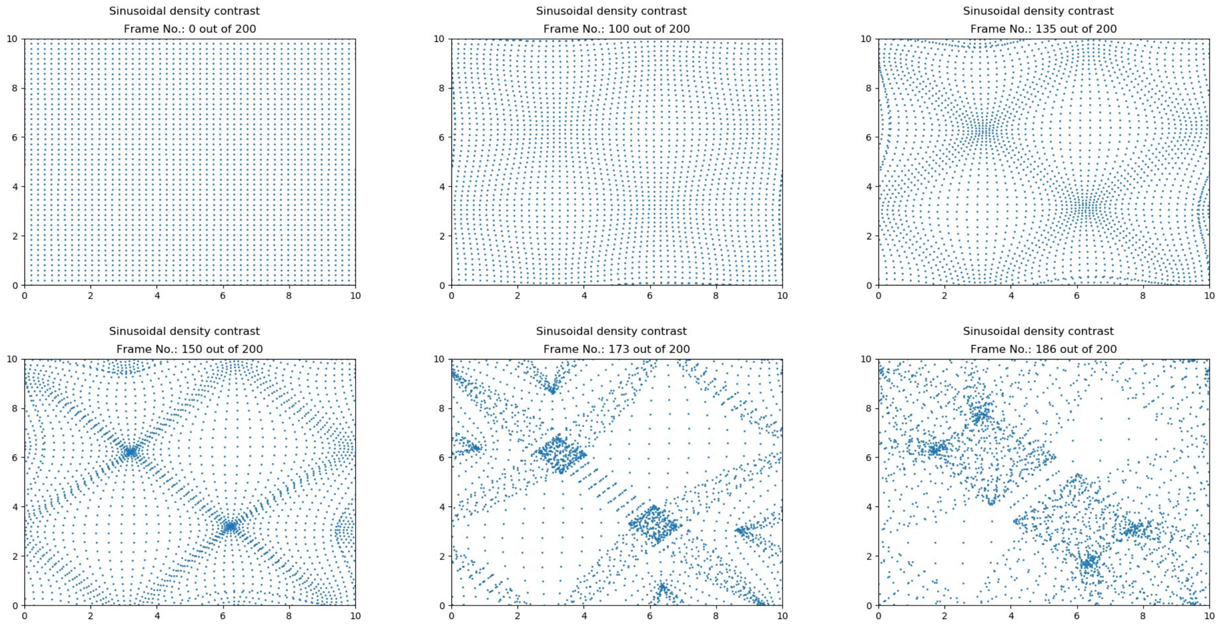
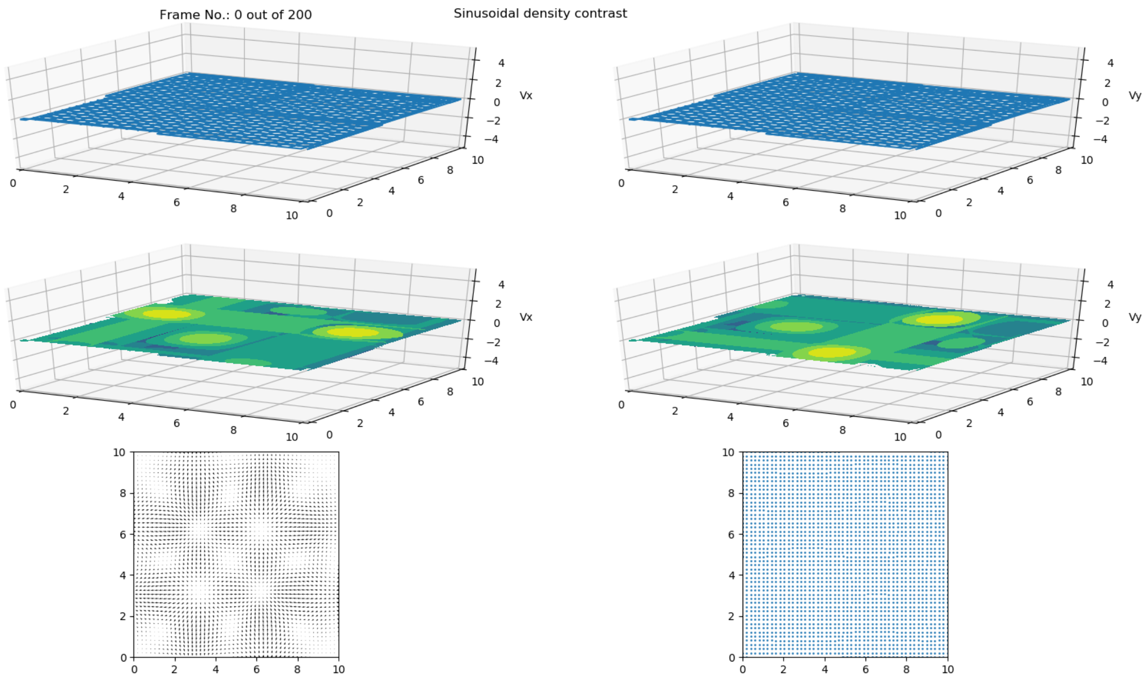


Figure 6: Evolution of 2500 particles with Sinusoidal Density contrast (2D)



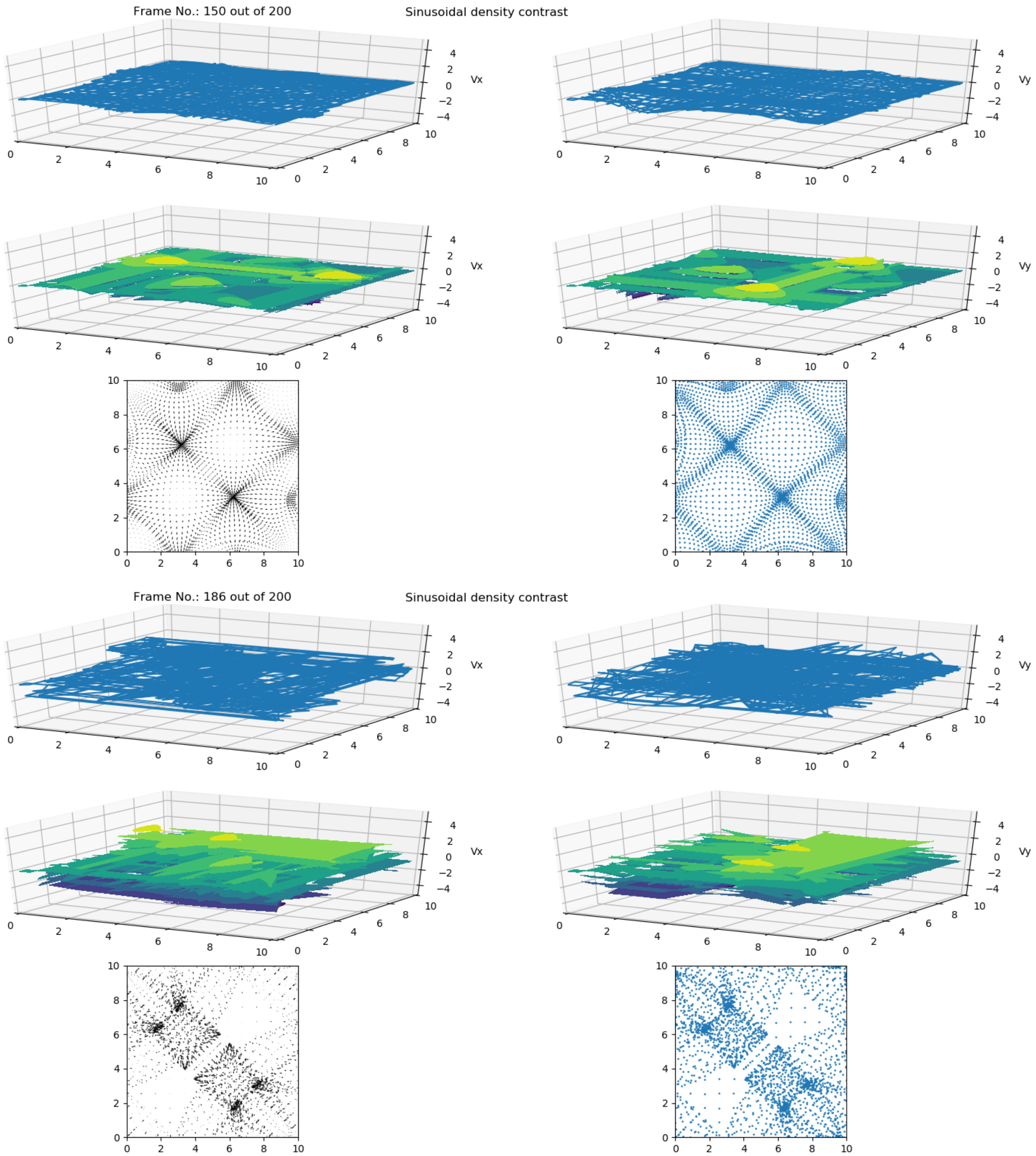


Figure 7: Evolution of phase space ($\dot{x}$ vs (x,y) and ($\dot{y}$ vs (x,y)) (2D)

Lastly we have the 3D plots:

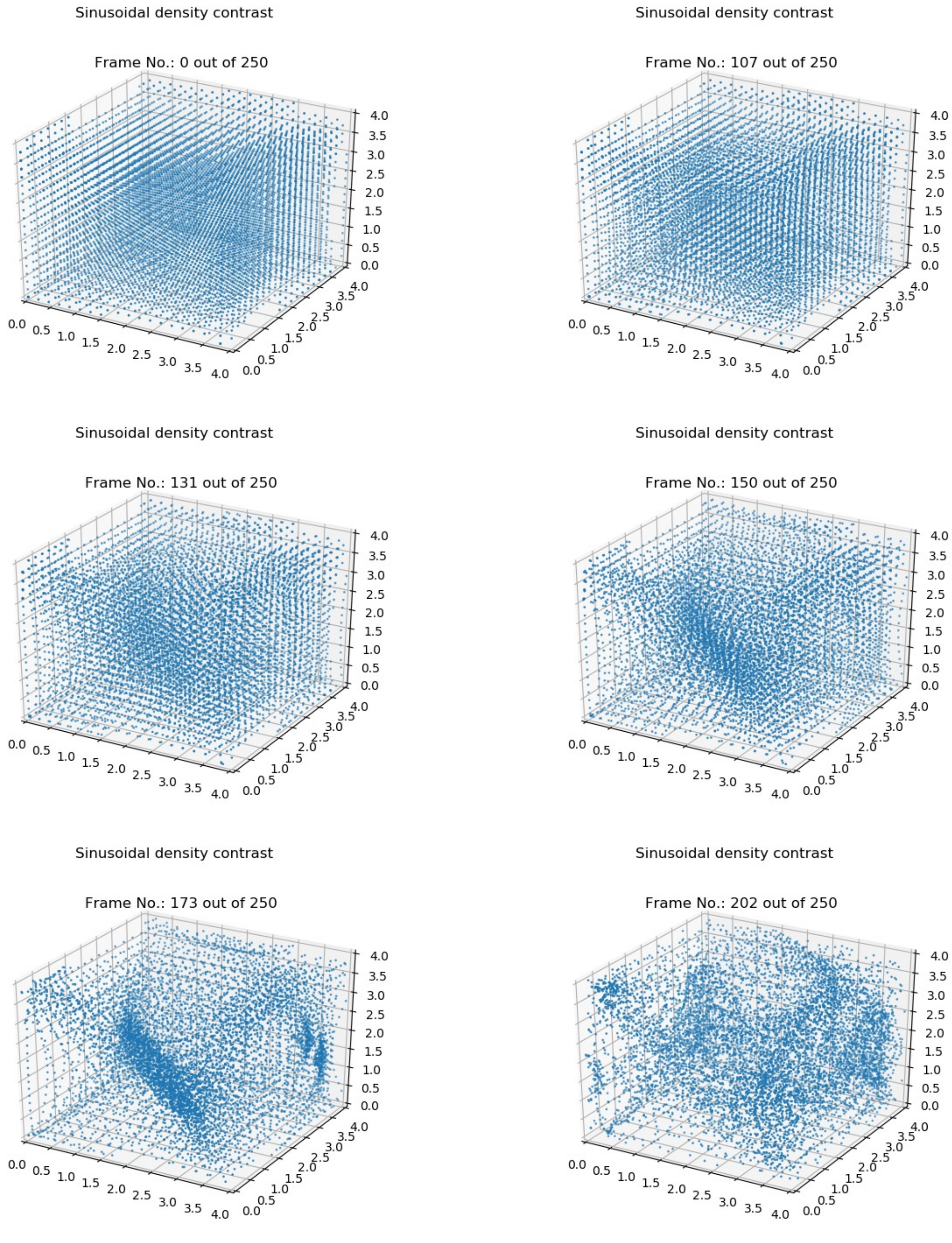


Figure 8: Evolution of 8000 particles with Sinusoidal Density contrast (3D)

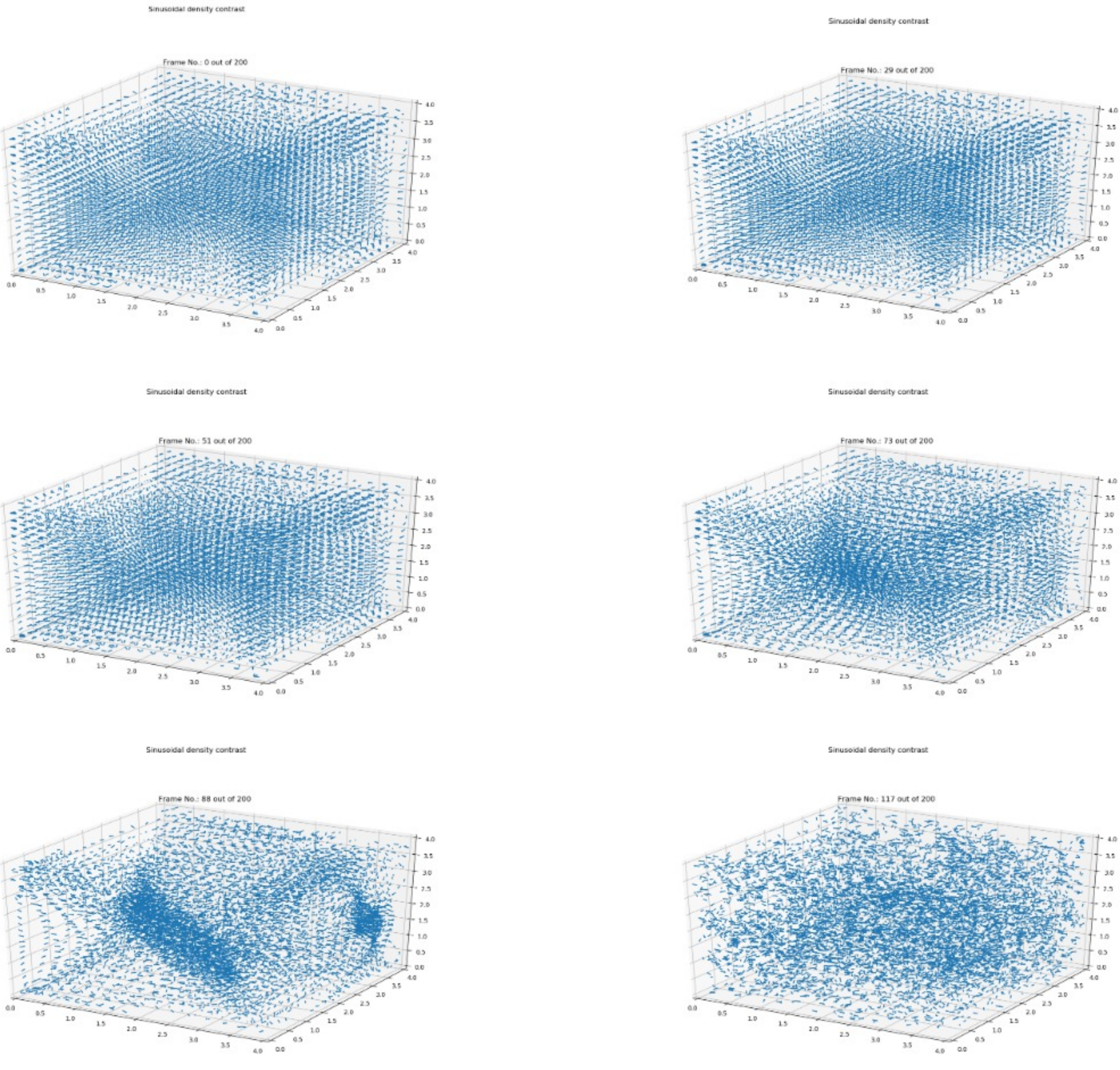


Figure 9: Evolution of phase space (Quiver Plot)

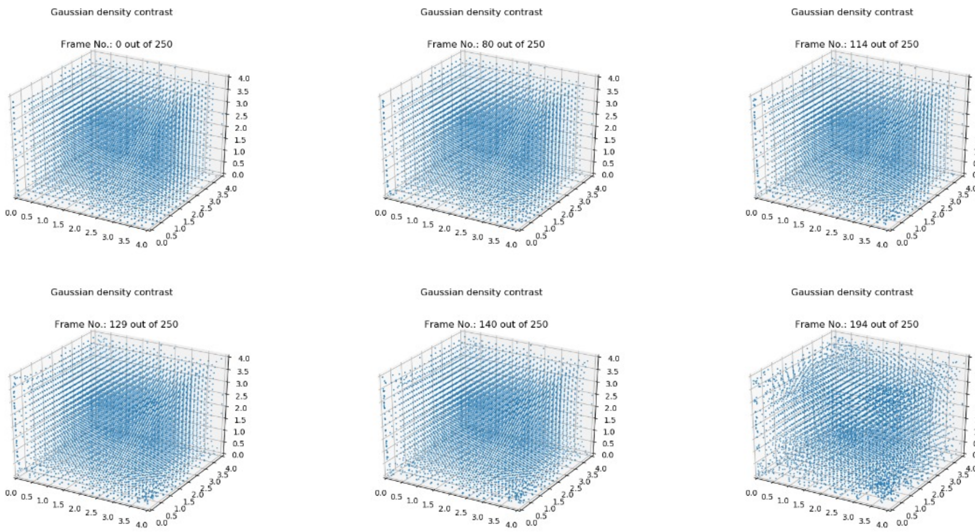


Figure 10: Evolution of 8000 particles with Gaussian Density contrast (3D)

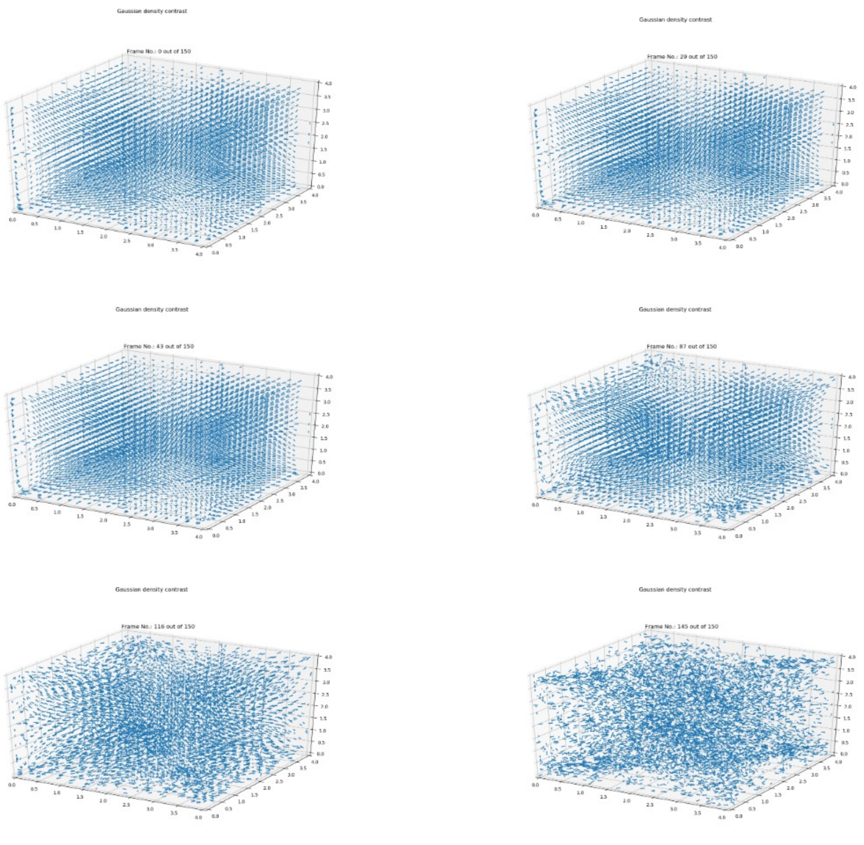


Figure 11: Evolution of phase space (Quiver Plot)

4 Conclusion

The universe, as large as it is, has a certain structure at large scales. This structure can be mathematically modelled by various analytical means and its evolution can be studied. We looked at these analytical means in terms of Linear and Non-Linear Perturbation Theories and tried to simulate using the Linear Lagrangian Perturbation Theories. Once achieved, one can go further to study the Second order Lagrangian Perturbation theory (2LPT) and make even better models. There certainly are approximations, we assume collisionless particles and this assumption breaks down when particles coalesce into something called as shell crossing.
But, using this data, one can try to analyse the evolution of the universe into what it is now and try to constrain the dark matter parameters and work towards finding the best fit model.

References

[1] Bernardeau, F., Colombi, S., Gaztanaga, E., Scoccimarro, R. (2001). Large-Scale Structure of the Universe and Cosmological Perturbation Theory. (astro-ph/0112551v1), 18-19

[2] Jain, B. Bertschinger, E. (1993). Second Order Power Spectrum and Nonlinear Evolution at High Redshift. (astro-ph/9311070v1), 6

Appendices

A Discrete Fourier Transform codes

Without Numpy:

```python
import math, cmath,timeit
import matplotlib
from matplotlib import pyplot as plt
#initialization
T = int(input("Enter the Time Period (even): "))
N = int(input("Enter the number equally spaced samples (even): "))
freal = []
fimag = []
w = []
delt= T/N
const=-2*math.pi*(1j)
#range of t: [-T/2,T/2)
#range of n: [-N/2, N/2-1]
def DFT(m,N,delt):
    fm = 0
    for n in range(-N//2,N//2):
        f=f1(n*delt)
        fm+=(f*cmath.exp((const*m*n)/N))
    return fm
def f1(t):
    if t!=0:
        f1 = (math.sin(2*math.pi*t)/t)
    else:
        f1=1
    return f1
def f2(t):
    f2 = (math.exp(-0.5*t*t)/math.sqrt(2*math.pi))
    return f2
start = timeit.timeit()
for m in range(-N//2,N//2):
    freal.append((DFT(m,N,delt).real*delt))
    fimag.append((DFT(m,N,delt).imag*delt))
    w.append(2*math.pi*m/T)
    #print(DFT(m,N,delt))
end = timeit.timeit()
```

```
print(end - start)

plt.plot(w,freal)
plt.title("Real part Vs w")
plt.show()
plt.plot(w, fimag)
plt.title("Imaginary part Vs w")
plt.show()
```

With Numpy:

```
import math, cmath,timeit
import numpy as np
import matplotlib
from matplotlib import pyplot as plt
#initialization
T = int(input("Enter the Time Period (even): "))
N = int(input("Enter the number equally spaced samples (even): "))
freal = []
fimag = []
w = []
delt= T/N
const=(-2*math.pi*(1j))/N
wl = np.empty([N,N],dtype =complex)
fn = np.empty([N],dtype =complex)
#range of t: [-T/2,T/2)
#range of n: [-N/2, N/2-1]
def DFT(N,delt):
    for n in range(0,N):
        fn[n] = f1((n-N//2)*delt)
        for m in range(0,N):
            wl[m][n]= cmath.exp(const*(m-N//2)*(n-N//2))
    return np.dot(wl,fn)
def f1(t):
    if t!=0:
        f1 = (math.sin(2*math.pi*t)/t)
    else:
        f1=1
    return f1
def f2(t):
    f2 = (math.exp(-0.5*t*t)/math.sqrt(2*math.pi))
    return f2
```

```
start = timeit.timeit()
F=DFT(N,delt)
for m in range(0,N):
    freal.append((F[m].real*delt))
    fimag.append((F[m].imag*delt))
    w.append(2*math.pi*(m-N//2)/T)
    #print(DFT(N,delt)[m])
end = timeit.timeit()
print(end - start)

plt.plot(w,freal)
plt.title("Real part Vs w")
plt.show()
plt.plot(w, fimag)
plt.title("Imaginary part Vs w")
plt.show()
```

B Fast Fourier Transform code

```
import math, cmath,timeit
import matplotlib, scipy
from matplotlib import pyplot as plt
import numpy as np
from scipy.fftpack import fft, ifft,fftfreq,rfftfreq, fftshift
#initialization
T = int(input("Enter the Time Period (even): "))
N = int(input("Enter the number equally spaced samples (even): "))
delt= T/N
const=-2*math.pi*(1j)
t = np.linspace(-T/2, T/2, N)
xf = rfftfreq(N,1/T)[1:]
f1 = np.piecewise(t, [t!=0 , t==0], [np.sin(2*np.pi*t)/t,1])
f2 = (np.exp(-0.5*t*t)/np.sqrt(2*math.pi))
start = timeit.timeit()
y= fftshift(fft(f2))[1:]
iy=(ifft(y))
yf = np.abs(y)
iyf = np.abs(iy)
end = timeit.timeit()
print(xf)
print(end - start)
```

```
plt.plot(xf,iyf)
plt.grid()
plt.show()
```

C Zel'dovich Simulation Codes

1-D Motion:

```
import numpy as np
import math
import matplotlib,scipy
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as animation
from scipy.fftpack import fft, ifft,fftfreq,rfftfreq, fftshift
print('''Different eras:
      1. Radiation Dominated era
      2. Matter Dominated era
      3. Dark-energy Dominated era''')
ch=int(input("Enter the era:")) #set to 3
H=0.7088 #km/s/Mpc
t=-3
def av(t):
    if ch==1:
        ax = t**0.5
    elif ch==2:
        ax = t**(2/3)
    elif ch==3:
        ax = math.exp(H*t)
    return ax
n=int(input("Enter the number of particles:")) #set to 50
Om=float(input("Enter the value of Omega_m:"))  #set to 1
Ol=float(input("Enter the value of Omega_lambda:")) #set to 0
def D1p(a):
    if Om==1 and Ol==0:
        D1p = a
    elif Om<1 and Ol==0:
        x=((1/Om)-1)
        D1p = (1+(3/x)+3*math.pow((1+x)/x**3,0.5)*math.log(((1+x)**0.5)
    -(x**0.5)))
    else:
        D1p = ((2.5*a*Om)/((Om**(4/7))-Ol+(1+0.5*Om)*(1+(Ol/70))))
```

```
    return D1p
#Zeldovich
A=1
B=1
xlim=n/5
init = np.linspace(0,xlim,n)
#Fourier Transform
T = init[-1]-init[0]
N = n
x=init
delta = A*np.sin(B*x)
xf = (fftfreq(N,T/N))*2*np.pi
xf+=1e-16
deltilde=(fft(delta))
psitildei=(((1j)*deltilde)/(xf))
psii=np.real((ifft(psitildei)))
delt=0.01
time=350
for i in range(0,time):
    a=av(t+i*delt)
    D1=D1p(a)
    psi=D1*psii
    plt.xlim(0,xlim)
    plt.ion()
    plt.suptitle("Sinusoidal density contrast")
    plt.title("Frame No.: %d"%i+" out of %d"%time)
    plt.scatter(np.mod(init+psi,xlim),[0]*n,s=5)
    plt.show()
    plt.savefig('Images/%d'%i)
    plt.pause(0.01)
    if i!=time-1:
        plt.clf()
```

2-D Motion:

```
import numpy as np
import math
import matplotlib,scipy
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as animation
from scipy.fftpack import fft, ifft,fft2, ifft2, fftfreq,rfftfreq,
```

```
   fftshift
print('''Different eras:
      1. Radiation Dominated era
      2. Matter Dominated era
      3. Dark-energy Dominated era''')
ch=int(input("Enter the era:")) #set to 3
H=0.7088 #km/s/Mpc
t=-10
def av(t):
    if ch==1:
        ax = t**0.5
    elif ch==2:
        ax = t**(2/3)
    elif ch==3:
        ax = math.exp(H*t)
    return ax
n=int(input("Enter the number of particles:")) #set to 50
Om=float(input("Enter the value of Omega_m:"))  #set to 1
Ol=float(input("Enter the value of Omega_lambda:")) #set to 0
def D1p(a):
    if Om==1 and Ol==0:
        D1p = a
    elif Om<1 and Ol==0:
        x=((1/Om)-1)
        D1p = (1+(3/x)+3*math.pow((1+x)/x**3,0.5)*math.log(((1+x)**0.5)
   -(x**0.5)))
    else:
        D1p = ((2.5*a*Om)/((Om**(4/7))-Ol+(1+0.5*Om)*(1+(Ol/70))))
    return D1p
#Zeldovich
xlim=n/5
ylim=n/5
A=1
B=1
C=1
initx = np.linspace(0,xlim,n)
inity = np.linspace(0,ylim,n)
posx,posy = np.meshgrid(initx,inity)
#Fourier Transform
x, y = np.meshgrid(initx, inity, sparse=False, indexing='ij')
#FUNCTION
sigma=0.05
```

```
#delta = 1/(2*np.pi*sigma**2) * np.exp(-0.5 * (x ** 2 + y ** 2)/sigma**2)
delta = np.sin(x)*np.sin(y)
F = fft2(delta)
T = initx[-1]-initx[0]
xaf = (fftfreq(F.shape[0],d=2))*2*np.pi
xaf+=1e-16
yaf = (fftfreq(F.shape[1],d=2))*2*np.pi
yaf+=1e-16
kx, ky = np.meshgrid(xaf, yaf, sparse=False, indexing='ij')
#x-coordinate
psitildeix=(((1j)*F*kx)/(kx**2+ky**2))
psiix=np.real((ifft2(psitildeix)))
#y-coordinate
psitildeiy=(((1j)*F*ky)/(kx**2+ky**2))
psiiy=np.real((ifft2(psitildeiy)))
#psimx,psimy = np.meshgrid(psiix,psiiy)
delt=0.05
time=350
for i in range(0,time):
    a=av(t+i*delt)
    D1=D1p(a)
    psix=D1*psiix
    psiy=D1*psiiy
    plt.xlim(0,xlim)
    plt.ylim(0,ylim)
    plt.ion()
    plt.suptitle("Gaussian density contrast")
    plt.title("Frame No.: %d"%i+" out of %d"%time)
    plt.scatter(np.mod(posx+psix,xlim),np.mod(posy+psiy,ylim),s=1)
    plt.savefig('Images/%d'%i)
    plt.show()
    plt.pause(0.01)
    if i!=time-1:
        plt.clf()
```

<u>3-D Motion</u>:

```
import numpy as np
import math
import matplotlib,scipy
from matplotlib import pyplot as plt
```

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as animation
from scipy.fftpack import fft, ifft,fftn, ifftn,fftfreq,rfftfreq,
    fftshift
print('''Different eras:
      1. Radiation Dominated era
      2. Matter Dominated era
      3. Dark-energy Dominated era''')
ch=int(input("Enter the era:")) #set to 3
H=0.7088 #km/s/Mpc
t=-10
def av(t):
    if ch==1:
        ax = t**0.5
    elif ch==2:
        ax = t**(2/3)
    elif ch==3:
        ax = math.exp(H*t)
    return ax
n=int(input("Enter the number of particles:")) #set to 20
Om=float(input("Enter the value of Omega_m:"))  #set to 1
Ol=float(input("Enter the value of Omega_lambda:")) #set to 0
def D1p(a):
    if Om==1 and Ol==0:
        D1p = a
    elif Om<1 and Ol==0:
        x=((1/Om)-1)
        D1p = (1+(3/x)+3*math.pow((1+x)/x**3,0.5)*math.log(((1+x)**0.5)
    -(x**0.5)))
    else:
        D1p = ((2.5*a*Om)/((Om**(4/7))-Ol+(1+0.5*Om)*(1+(Ol/70))))
    return D1p
#Zeldovich
xlim=n/5
ylim=n/5
zlim=n/5
A=1
B=1
C=1
initx = np.linspace(0,xlim,n)
inity = np.linspace(0,ylim,n)
initz = np.linspace(0,zlim,n)
```

```
posx,posy,posz = np.meshgrid(initx,inity,initz)
#Fourier Transform
x, y, z= np.meshgrid(initx, inity, initz, sparse=False, indexing='ij')
#FUNCTION
sigma=0.05
delta = 1/(2*np.pi*sigma**2) * np.exp(-0.5 * (x ** 2 + y ** 2 + z ** 2)
    /sigma**2)
#delta = np.sin(x)*np.sin(y)*np.sin(z)
F = fftn(delta)
T = initx[-1]-initx[0]
xaf = (fftfreq(F.shape[0],d=3))*2*np.pi
xaf+=1e-16
yaf = (fftfreq(F.shape[1],d=3))*2*np.pi
yaf+=1e-16
zaf = (fftfreq(F.shape[2],d=3))*2*np.pi
zaf+=1e-16
kx, ky, kz= np.meshgrid(xaf, yaf, zaf, sparse=False, indexing='ij')
#x-coordinate
psitildeix=(((1j)*F*kx)/(kx**2+ky**2+kz**2))
psiix=np.real((ifftn(psitildeix)))
#y-coordinate
psitildeiy=(((1j)*F*ky)/(kx**2+ky**2+kz**2))
psiiy=np.real((ifftn(psitildeiy)))
#z-coordinate
psitildeiz=(((1j)*F*kz)/(kx**2+ky**2+kz**2))
psiiz=np.real((ifftn(psitildeiz)))
#psimx,psimy = np.meshgrid(psiix,psiiy)
delt=0.05
time=250
fig = plt.figure()
plt.ion()
for i in range(0,time):
    ax = fig.add_subplot(111, projection='3d')
    a=av(t+i*delt)
    D1=D1p(a)
    psix=D1*psiix
    psiy=D1*psiiy
    psiz=D1*psiiz
    plt.suptitle("Gaussian density contrast")
    plt.title("Frame No.: %d"%i+" out of %d"%time)
    ax.set_xlim3d(0,xlim)
    ax.set_ylim3d(0,ylim)
```

```
    ax.set_zlim3d(0,zlim)
    ax.scatter3D(np.mod(posx+psix,xlim),np.mod(posy+psiy,ylim),np.mod(
   posz+psiz,zlim),s=1)
    plt.savefig('Images/%d'%i)
    plt.show()
    plt.pause(0.01)
    if i!=time-1:
        plt.clf()
```

D Zel'dovich 2-D timing analysis

```
import numpy as np
import math, time
import matplotlib,scipy
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as animation
from scipy.fftpack import fft, ifft,fft2, ifft2, fftfreq,rfftfreq,
   fftshift
print('''Different eras:
      1. Radiation Dominated era
      2. Matter Dominated era
      3. Dark-energy Dominated era''')
ch=int(input("Enter the era:")) #set to 3
H=0.7088 #km/s/Mpc
t=-10
def av(t):
    if ch==1:
        ax = t**0.5
    elif ch==2:
        ax = t**(2/3)
    elif ch==3:
        ax = math.exp(H*t)
    return ax
Om=float(input("Enter the value of Omega_m:"))  #set to 1
Ol=float(input("Enter the value of Omega_lambda:")) #set to 0
def D1p(a):
    if Om==1 and Ol==0:
        D1p = a
    elif Om<1 and Ol==0:
        x=((1/Om)-1)
```

```
            D1p = (1+(3/x)+3*math.pow((1+x)/x**3,0.5)*math.log(((1+x)**0.5)-(x**0.5)))
    else:
        D1p = ((2.5*a*Om)/((Om**(4/7))-Ol+(1+0.5*Om)*(1+(Ol/70))))
    return D1p
#Zeldovich 2D time
N=[]
TIME=[]
nvalues=[]
for i in range(1,11):
    nvalues.append(math.pow(2,i))
print(nvalues)
for n in nvalues:
    start = time.time()
    for i in range(100):
        xlim=n/5
        ylim=n/5
        A=1
        B=1
        C=1
        initx = np.linspace(0,xlim,n)
        inity = np.linspace(0,ylim,n)
        posx,posy = np.meshgrid(initx,inity)
        #Fourier Transform
        x, y = np.meshgrid(initx, inity, sparse=False, indexing='ij')
        #FUNCTION
        sigma=0.05
        #delta = 1/(2*np.pi*sigma**2) * np.exp(-0.5 * (x ** 2 + y ** 2)/sigma**2)
        delta = np.sin(x)*np.sin(y)
        F = fft2(delta)
        T = initx[-1]-initx[0]
        xaf = (fftfreq(F.shape[0],d=2))*2*np.pi
        xaf+=1e-16
        yaf = (fftfreq(F.shape[1],d=2))*2*np.pi
        yaf+=1e-16
        kx, ky = np.meshgrid(xaf, yaf, sparse=False, indexing='ij')
        #x-coordinate
        psitildeix=(((1j)*F*kx)/(kx**2+ky**2))
        psiix=np.real((ifft2(psitildeix)))
        #y-coordinate
        psitildeiy=(((1j)*F*ky)/(kx**2+ky**2))
```

```
        psiiy=np.real((ifft2(psitildeiy)))
        i+=1
    end = time.time()
    N.append(n*n)
    print(n*n)
    TIME.append((end-start)/100)
plt.plot(N,TIME)
plt.xlabel("Number of particles")
plt.ylabel("Time taken")
plt.title("2D time analysis")
plt.show()
```

E Zel'dovich Phase Space Simulation codes

1-D Motion:

```
import numpy as np
import math
import matplotlib,scipy
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as animation
from scipy.fftpack import fft, ifft,fftfreq,rfftfreq, fftshift
print('''Different eras:
      1. Radiation Dominated era
      2. Matter Dominated era
      3. Dark-energy Dominated era''')
ch=int(input("Enter the era:")) #set to 3
H=0.7088 #km/s/Mpc
t=-3
def av(t):
    if ch==1:
        ax = t**0.5
    elif ch==2:
        ax = t**(2/3)
    elif ch==3:
        ax = math.exp(H*t)
    return ax
n=int(input("Enter the number of particles:")) #set to 50
Om=float(input("Enter the value of Omega_m:"))  #set to 1
Ol=float(input("Enter the value of Omega_lambda:")) #set to 0
def D1p(a):
```

```
    if Om==1 and Ol==0:
        D1p = a
    elif Om<1 and Ol==0:
        x=((1/Om)-1)
        D1p = (1+(3/x)+3*math.pow((1+x)/x**3,0.5)*math.log(((1+x)**0.5)
    -(x**0.5)))
    else:
        D1p = ((2.5*a*Om)/((Om**(4/7))-Ol+(1+0.5*Om)*(1+(Ol/70))))
    return D1p
#Zeldovich
A=1
B=1
xlim=n/5
init = np.linspace(0,xlim,n)
#Fourier Transform
T = init[-1]-init[0]
N = n
x=init
delta = A*np.sin(B*x)
xf = (fftfreq(N,T/N))*2*np.pi
xf+=1e-16
deltilde=(fft(delta))
psitildei=(((1j)*deltilde)/(xf))
psii=np.real((ifft(psitildei)))
delt=0.03
time=200
for i in range(0,time):
    a1=av(t+i*delt)
    a2=av(t+(i+1)*delt)
    D1=D1p(a1)
    dD1=((D1p(a2)-D1p(a1))/delt)
    psi=D1*psii
    dpsi=dD1*psii
    plt.xlim(0,xlim)
    plt.ylim(-n/20,n/20)
    plt.ion()
    plt.suptitle("Sinusoidal density contrast")
    plt.title("Frame No.: %d"%i+" out of %d"%time)
    plt.scatter(np.mod(init+psi,xlim),dpsi,s=10)
    plt.savefig('Images/%d'%i)
    plt.show()
    plt.pause(0.001)
```

```
    if i!=time-1:
        plt.clf()
```

2-D Motion:

```
import numpy as np
import math
import matplotlib,scipy
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as animation
from scipy.fftpack import fft, ifft,fft2, ifft2, fftfreq,rfftfreq,
    fftshift
print('''Different eras:
      1. Radiation Dominated era
      2. Matter Dominated era
      3. Dark-energy Dominated era''')
ch=int(input("Enter the era:")) #set to 3
H=0.7088 #km/s/Mpc
t=-10
def av(t):
    if ch==1:
        ax = t**0.5
    elif ch==2:
        ax = t**(2/3)
    elif ch==3:
        ax = math.exp(H*t)
    return ax
n=int(input("Enter the number of particles:")) #set to 50
Om=float(input("Enter the value of Omega_m:"))  #set to 1
Ol=float(input("Enter the value of Omega_lambda:")) #set to 0
def D1p(a):
    if Om==1 and Ol==0:
        D1p = a
    elif Om<1 and Ol==0:
        x=((1/Om)-1)
        D1p = (1+(3/x)+3*math.pow((1+x)/x**3,0.5)*math.log(((1+x)**0.5)
    -(x**0.5)))
    else:
        D1p = ((2.5*a*Om)/((Om**(4/7))-Ol+(1+0.5*Om)*(1+(Ol/70))))
    return D1p
#Zeldovich
```

```
xlim=n/5
ylim=n/5
zlim=n/5
A=1
B=1
C=1
initx = np.linspace(0,xlim,n)
inity = np.linspace(0,ylim,n)
posx,posy = np.meshgrid(initx,inity)
#Fourier Transform
x, y = np.meshgrid(initx, inity, sparse=False, indexing='ij')
#FUNCTION
sigma=0.05
#delta = 1/(2*np.pi*sigma**2) * np.exp(-0.5 * (x ** 2 + y ** 2)/sigma
    **2)
delta = np.sin(x)*np.sin(y)
F = fft2(delta)
T = initx[-1]-initx[0]
xaf = (fftfreq(F.shape[0],d=2))*2*np.pi
xaf+=1e-16
yaf = (fftfreq(F.shape[1],d=2))*2*np.pi
yaf+=1e-16
kx, ky = np.meshgrid(xaf, yaf, sparse=False, indexing='ij')
#x-coordinate
psitildeix=(((1j)*F*kx)/(kx**2+ky**2))
psiix=np.real((ifft2(psitildeix)))
#y-coordinate
psitildeiy=(((1j)*F*ky)/(kx**2+ky**2))
psiiy=np.real((ifft2(psitildeiy)))
#psimx,psimy = np.meshgrid(psiix,psiiy)
delt=0.05
time=200
fig = plt.figure(figsize=[16,9])
plt.ion()
cc=40
for i in range(0,time):
    a1=av(t+i*delt)
    a2=av(t+(i+1)*delt)
    D1=D1p(a1)
    dD1=((D1p(a2)-D1p(a1))/delt)
    dpsix=dD1*psiix
    psix=D1*psiix
```

```
    dpsiy=dD1*psiiy
    psiy=D1*psiiy
    plt.ion()
    plt.suptitle("Gaussian density contrast")
    #plots
    #Subplot 1
    ax = fig.add_subplot(3,2,1, projection='3d',aspect = 'auto')
    plt.title("Frame No.: %d"%i+" out of %d"%time)
    ax.set_xlim3d(0,xlim)
    ax.set_ylim3d(0,ylim)
    ax.set_zlim3d(-zlim/2,zlim/2)
    ax.plot_wireframe(np.mod(posx+psix,xlim),np.mod(posy+psiy,ylim),dpsix,rcount=cc,ccount=cc)
    ax.set_zlabel("Vx")
    #Subplot 2
    ax = fig.add_subplot(3,2,2, projection='3d',aspect = 'auto')
    ax.set_xlim3d(0,xlim)
    ax.set_ylim3d(0,ylim)
    ax.set_zlim3d(-zlim/2,zlim/2)
    ax.plot_wireframe(np.mod(posx+psix,xlim),np.mod(posy+psiy,ylim),dpsiy,rcount=cc,ccount=cc)
    ax.set_zlabel("Vy")
    #Subplot 3
    ax = fig.add_subplot(3,2,3, projection='3d',aspect = 'auto')
    ax.set_xlim3d(0,xlim)
    ax.set_ylim3d(0,ylim)
    ax.set_zlim3d(-zlim/2,zlim/2)
    ax.contourf(np.mod(posx+psix,xlim),np.mod(posy+psiy,ylim),dpsix)
    ax.set_zlabel("Vx")
    #Subplot 4
    ax = fig.add_subplot(3,2,4, projection='3d',aspect = 'auto')
    ax.set_xlim3d(0,xlim)
    ax.set_ylim3d(0,ylim)
    ax.set_zlim3d(-zlim/2,zlim/2)
    ax.contourf(np.mod(posx+psix,xlim),np.mod(posy+psiy,ylim),dpsiy)
    ax.set_zlabel("Vy")
    #Subplot5
    plt.subplot(3,2,5,aspect = 'equal')
    plt.xlim(0,xlim)
    plt.ylim(0,ylim)
    plt.quiver(np.mod(posx+psix,xlim),np.mod(posy+psiy,ylim),dpsix,dpsiy)
```

```
    #Subplot6
    plt.subplot(3,2,6,aspect = 'equal')
    plt.xlim(0,xlim)
    plt.ylim(0,ylim)
    plt.scatter(np.mod(posx+psix,xlim),np.mod(posy+psiy,ylim),s=0.8)
    fig.tight_layout()
    plt.savefig('Images/%d'%i)
    plt.show()
    plt.pause(0.01)
    if i!=time-1:
        plt.clf()
```

3-D Motion:

```
import numpy as np
import math
import matplotlib,scipy
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as animation
from scipy.fftpack import fft, ifft,fftn, ifftn,fftfreq,rfftfreq,
    fftshift
print('''Different eras:
      1. Radiation Dominated era
      2. Matter Dominated era
      3. Dark-energy Dominated era''')
ch=int(input("Enter the era:")) #set to 3
H=0.7088 #km/s/Mpc
t=-10
def av(t):
    if ch==1:
        ax = t**0.5
    elif ch==2:
        ax = t**(2/3)
    elif ch==3:
        ax = math.exp(H*t)
    return ax
n=int(input("Enter the number of particles:")) #set to 20
Om=float(input("Enter the value of Omega_m:"))  #set to 1
Ol=float(input("Enter the value of Omega_lambda:")) #set to 0
def D1p(a):
    if Om==1 and Ol==0:
```

```
        D1p = a
    elif Om<1 and Ol==0:
        x=((1/Om)-1)
        D1p = (1+(3/x)+3*math.pow((1+x)/x**3,0.5)*math.log(((1+x)**0.5)
    -(x**0.5)))
    else:
        D1p = ((2.5*a*Om)/((Om**(4/7))-Ol+(1+0.5*Om)*(1+(Ol/70))))
    return D1p
#Zeldovich
xlim=n/5
ylim=n/5
zlim=n/5
A=1
B=1
C=1
initx = np.linspace(0,xlim,n)
inity = np.linspace(0,ylim,n)
initz = np.linspace(0,zlim,n)
posx,posy,posz = np.meshgrid(initx,inity,initz)
#Fourier Transform
x, y, z= np.meshgrid(initx, inity, initz, sparse=False, indexing='ij')
#FUNCTION
sigma=0.05
delta = 1/(2*np.pi*sigma**2) * np.exp(-0.5 * (x ** 2 + y ** 2 + z ** 2)
    /sigma**2)
#delta = np.sin(x)*np.sin(y)*np.sin(z)
F = fftn(delta)
T = initx[-1]-initx[0]
xaf = (fftfreq(F.shape[0],d=3))*2*np.pi
xaf+=1e-16
yaf = (fftfreq(F.shape[1],d=3))*2*np.pi
yaf+=1e-16
zaf = (fftfreq(F.shape[2],d=3))*2*np.pi
zaf+=1e-16
kx, ky, kz= np.meshgrid(xaf, yaf, zaf, sparse=False, indexing='ij')
#x-coordinate
psitildeix=(((1j)*F*kx)/(kx**2+ky**2+kz**2))
psiix=np.real((ifftn(psitildeix)))
#y-coordinate
psitildeiy=(((1j)*F*ky)/(kx**2+ky**2+kz**2))
psiiy=np.real((ifftn(psitildeiy)))
#z-coordinate
```

```python
psitildeiz=(((1j)*F*kz)/(kx**2+ky**2+kz**2))
psiiz=np.real((ifftn(psitildeiz)))
#psimx,psimy = np.meshgrid(psiix,psiiy)
delt=0.1
time=150
fig = plt.figure(figsize=(16,9))
plt.ion()
for i in range(0,time):
    ax = fig.add_subplot(111, projection='3d')
    a1=av(t+i*delt)
    a2=av(t+(i+1)*delt)
    D1=D1p(a1)
    dD1=((D1p(a2)-D1p(a1))/delt)
    dpsix=dD1*psiix
    psix=D1*psiix
    dpsiy=dD1*psiiy
    psiy=D1*psiiy
    dpsiz=dD1*psiiz
    psiz=D1*psiiz
    plt.suptitle("Gaussian density contrast")
    ax.set_xlim3d(0,xlim)
    ax.set_ylim3d(0,ylim)
    ax.set_zlim3d(0,zlim)
    plt.title("Frame No.: %d"%i+" out of %d"%time)
    ax.quiver(np.mod(posx+psix,xlim),np.mod(posy+psiy,ylim),np.mod(posz
   +psiz,zlim),dpsix,dpsiy,dpsiz,length=0.08,normalize=True)
    plt.show()
    plt.savefig('Images/%d'%i)
    plt.pause(0.01)
    if i!=time-1:
        plt.clf()
```