Candidate Number: 134736

ANALYTICAL AND COMPUTATIONAL COSMOLOGY: From Equations of Fluid's to Cosmological Parameter Estimation

Supervisor: Dr David Seery
Word Count: 8887

Department of Physics and Astronomy

May 16, 2018

UNIVERSITY OF SUSSEX

Abstract

Most cosmological models suggest that the universe originated from a singularity of infinite density, gravitational potential and temperature. During the early expansion all matter was continuously and evenly distributed through space, but as the universe cooled, the uniform density distribution began to fluctuate creating gravitational potential wells.

This paper investigates the growth in the structure of the early universe from a homogeneous and isotropic state to what is observed today. This is done by modelling the early universe using equations that describe fluid dynamics in their vector forms and using perturbation theory to replicate these fluctuations and their growth over time.

By modelling the perturbations, it is possible to computationally simulate the theoretical power spectra observed from Earth, dependent on the angle of line of sight and k-space, using a Python package called CAMB. This theoretical model of the multipole power spectrum is then statistically compared and convolved with raw-data from the WiggleZ Dark Energy Survey observed from the Anglo Australian Telescope in New South Wales, in a rigorous manner to compute the chi-square and likelihood for seven different regions in the night sky. Up until this point, all the code has been built up from the arguments of six cosmological parameters.

CosmoSIS is a cosmological parameter estimation package. It can be used to find the minimum likelihood by applying numerical methods to already existing code and returning the optimal cosmological parameters. By running CosmoSIS in the Sussex University's HPC environment, called Apollo, 10,000 combinations of the optimal cosmological parameters are generated and plotted together using GetDist where the implications of the correlations are remarked on.

Contents

Preface

Section 1 is a literature review, where all the material has been expanded to provide a more detailed development from my own derivations. This consists of the work discussed and analysed with Dr David Seery, along with the books that David had recommended, and any information that I could find online through Google and Wikipedia to consolidate my understanding.

Section 2 is based on explanations from David Seery regarding the purpose of each bit of code, but the code itself is all my own work (except for the code setup for CosmoSIS to run in Apollo, such as making the class RSDPk). The data used comes from the WizCOLA suite based on observational data from the WiggleZ Dark Energy Survey. I have produced all the plots throughout this paper myself, and all the explanations are my own, following researches online and discussions with David Seery.

1 Fluid Dynamics to Cosmological Perturbations

1.1 Section 1: Introduction

On a cosmological scale, galaxy cluster formation can be modelled using vector calculus and the equations of fluid and continuum mechanics. That is, it is still possible to describe the universe without the need for complexity, such as general relativity.

During the early stages of the universe, all matter was compressed to a singularity of infinite pressure, temperature and gravitational potential. As the universe expanded and cooled, matter was evenly and continuously distributed in all directions, but over time, matter began clustering together due to its own self-gravity. This effect created irregularities in the matter-density distribution, fluctuating around a smooth and constant background density leading to the formation of gravitational potential wells.

The purpose of Section 1 is to learn more about the structure of the universe and how it departed from uniform density. This is done by providing an analytical development of vector calculus to the mathematical descriptions of fluid and continuum mechanics. These equations are then perturbed to an expanding background to model the density and gravitational fluctuations in the universe. Starting from the basics, all the theory is based on derivation where the physics and equations that are explored later on are built up and derived from the ones before.

We first cover the fundamental identities and theorems in vector calculus and apply them to the derivations of three useful equations in fluid dynamics that play a large role in cosmology. Up until this point the universe is assumed to be completely uniform, where no fluctuations in density and gravitational potential exist. After deriving these equations, we apply perturbation theory to analyse a slight imbalance in the distribution of density, gravitation potential and flow velocity. Expanding and simplifying these equations using Hubble's constant and a few analytical techniques, we arrive with the equations in their perturbed states. Further development can be made to derive important cosmological parameters, where these parameters are used in simulating a cosmological model of the power spectrum of distant galaxies in Section 2.

1.2 Fundamental Concepts in Vector Calculus

Fluid dynamics is concerned with the behavior of liquids, gases and plasmas and the forces between them. The subject depends heavily on algebraic manipulation through multi-variable vector-calculus. A brief review in vector-calculus identities is provided, along with the divergence theorem, and the material derivative that accounts for changes with respect to time and three-dimensional space.

1.2.1 Vector Identities

This page is provided to give a recap of relevant vector calculus identities in the three-dimensional Cartesian coordinate plane.

Del or nabla, is an operator, which means that it acts on a given vector or scaler function, and describes the directional rate of change in three dimensions.

$$\vec{\nabla} \;=\; \vec{i}\,\frac{\partial}{\partial x} \;+\; \vec{j}\,\frac{\partial}{\partial y} \;+\; \vec{k}\,\frac{\partial}{\partial z}$$

where $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$, $\frac{\partial}{\partial z}$ are partial derivatives that only commute with their corresponding $\vec{i}$, $\vec{j}$, $\vec{k}$ vectors.

When $\vec{\nabla}$ operates on a **scalar** function $f(x, y, z)$, the result is the **gradient** of $f(x, y, z)$.

$$\vec{\nabla} f(x,y,z) \;=\; \vec{i}\,\frac{\partial}{\partial x} f(x,y,z) \;+\; \vec{j}\,\frac{\partial}{\partial y} f(x,y,z) \;+\; \vec{k}\,\frac{\partial}{\partial z} f(x,y,z)$$

When $\vec{\nabla}$ operates on a **vector** function $\vec{F}(x, y, z)$, the result is the **divergence** of $\vec{F}(x, y, z)$.

$$\vec{\nabla} \cdot \vec{F}(x,y,z) \;=\; \frac{\partial}{\partial x} F_x(x,y,z) \;+\; \frac{\partial}{\partial y} F_y(x,y,z) \;+\; \frac{\partial}{\partial z} F_z(x,y,z)$$

When computing the directional rate of change of a **vector** function $\vec{F}(x, y, z)$ perpendicular to its direction, the result is called the **curl** of $\vec{F}(x, y, z)$.

$$\vec{\nabla} \times \vec{F}(x,y,z) \;=\begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ F_x & F_y & F_z \end{vmatrix}$$

$$\vec{\nabla} \times \vec{F}(x,y,z) \;=\; (\frac{\partial F_z}{\partial y} \;-\; \frac{\partial F_y}{\partial z})\,\vec{i} \;+\; (\frac{\partial F_x}{\partial z} \;-\; \frac{\partial F_z}{\partial x})\,\vec{j} \;+\; (\frac{\partial F_y}{\partial x} \;-\; \frac{\partial F_x}{\partial y})\,\vec{k}$$

When $\vec{\nabla}$ operates on the **gradient** of $\vec{f}(x, y, z)$, the result is the **Laplacian** of $\vec{F}(x, y, z)$.

$$\vec{\nabla}^2 f(x,y,z)) \;=\; \frac{\partial^2}{\partial x^2} f(x,y,z) \;+\; \frac{\partial^2}{\partial y^2} f(x,y,z) \;+\; \frac{\partial^2}{\partial z^2} f(x,y,z)$$

1.2.2 The Divergence Theorem

In vector calculus, the divergence theorem allows the flux of a vector field $\vec{F}$ to be described, both in terms of the enclosed area cut by $\vec{F}$, and in terms of the enclosed volume crossed by $\vec{F}$. It provides the useful relationship between the surface integral and the volume integral for an enclosed region acted upon by a vector field. Namely by definition, the divergence theorem states that the outward flux of a vector field through a closed surface is equal to the volume integral of the divergence over the region enclosed by the surface.

The divergence theorem is a fundamental result. It is normally expressed in three dimensions, however it can be generalized to any number of dimensions. In two dimensions, it is equivalent to Green's theorem. In one dimension, it is equivalent to the fundamental theorem of calculus.

For the sake of intuition and simplicity, it is easier to derive the divergence theorem by considering a rectangular box of lengths $a \leq x \leq b$, $c \leq y \leq d$, $e \leq z \leq f$ in the Cartesian plane. Consider a smooth vector field $\vec{F}$ defined on the rectangular box of volume V, and let Φ represent the flux on each plane on the box cut by $\vec{F}$.

The flux passing through the box in the x direction from a to b is

$$\begin{aligned}\Phi_1 \ + \ \Phi_2 \ &= \ \int_{A_1} \vec{F} \cdot d\vec{A} \ + \ \int_{A_2} \vec{F} \cdot d\vec{A} \\ &= \ - \int_e^f \int_c^d F(a,y,z)\,dy\,dz \ + \int_e^f \int_c^d F(b,y,z)\,dy\,dz \\ &= \ \int_e^f \int_c^d (\,F(b,y,z) \ - \ F(z,y,z)\,)\,dy\,dz\end{aligned}$$

where the first term is evaluated as negative because the vector F points in the opposite direction to the vector dA on the plane A_1.

Now, from the fundamental theorem of calculus

$$(\,F(b,y,z) \ - \ F(a,y,z)\,) = \int_a^b \frac{\partial F_x}{\partial x}\,dx$$

hence by substitution,

$$\Phi_1 \ + \ \Phi_2 \ = \ \int_e^f \int_c^d \int_a^b \frac{\partial F_x}{\partial x}\,dx\,dy\,dz$$

where the infinitesimal volume of the box dV is represented by $dx, dy, dz,$ the result becomes,

$$\Phi_1 \ + \ \Phi_2 \ = \ \int_V \frac{\partial F_x}{\partial x}\,dV$$

By similar argument, we can find expressions of the flux passing through all planes of the box:

in the y direction,

$$\begin{aligned}\Phi_3 \;+\; \Phi_4 \;&=\; \int_{A_3} \vec{F}\cdot d\vec{A} \;+\; \int_{A_4} \vec{F}\cdot d\vec{A}\\ &=\; \int_e^f \int_a^b \int_c^d \frac{\partial F_y}{\partial y}\, dy\, dx\, dz\\ &=\; \int_V \frac{\partial F_y}{\partial y}\, dV\end{aligned}$$

in the z direction,

$$\begin{aligned}\Phi_5 \;+\; \Phi_6 \;&=\; \int_{A_5} \vec{F}\cdot d\vec{A} \;+\; \int_{A_6} \vec{F}\cdot d\vec{A}\\ &=\; \int_a^b \int_c^d \int_e^f \frac{\partial F_z}{\partial z}\, dz\, dy\, dx\\ &=\; \int_V \frac{\partial F_z}{\partial z}\, dV\end{aligned}$$

Then, by adding adding up all terms in the x, y and z directions, the total flux passing through the box is

$$\Phi_{\text{total}} \;=\; \int_V \frac{\partial F_x}{\partial x}\, dV \;+\; \int_V \frac{\partial F_y}{\partial y}\, dV \;+\; \int_V \frac{\partial F_z}{\partial z}\, dV$$

$$\Phi_{\text{total}} \;=\; \int_V (\,\frac{\partial F_x}{\partial x} \;+\; \frac{\partial F_y}{\partial y} \;+\; \frac{\partial F_z}{\partial z}\,)\, dV$$

Finally, substituting in for the divergence of $\vec{F}(x, y, z)$ simplifies the total flux to

$$\Phi_{\text{total}} \;=\; \int_V \vec{\nabla} \cdot \vec{F}(x, y, z)\, dV$$

Remembering that the total flux can also be represented by

$$\Phi_{\text{total}} \;=\; \int_A \vec{F}(x, y, z)\cdot d\vec{A}$$

Then the flux equivalence of the surface to volume integral is known as the divergence theorem

$$\int_A \vec{F}\cdot d\vec{A} \;=\; \int_V \vec{\nabla}\cdot\vec{F}\, dV$$

1.2.3 The Material Derivative

In fluid and continuum mechanics, where physical quantities and properties are not discrete, the material derivative describes the rate of change of a physical quantity associated with a material or fluid element that is subjected to space and time variations whilst moving along a path with velocity $\vec{v}$.

The material derivative differs from other vector or scalar derivatives as it deals with the rate of change with respect to both space and time for a physical quantity of interest, opposed to an operator such as del, that only computes the time derivative for a quantity of interest.

In order to mathematically describe the kinematics of a fluid, the nature of the fluid's 'flow' and how the flow changes over time, consider a function F that depends on three-dimensional space x, y, z and time t. Let F, for example, represent a quantity of interest such as a component of the fluid's momentum $\vec{P_x}$, or the fluid's density ρ.

Then, the function F and its rate $\frac{d}{dt}F$ is denoted by:

$$\begin{aligned} F &= F(x(t), y(t), z(t), t) \\ \frac{DF}{Dt} &= \frac{d}{dt}F(x(t), y(t), z(t), t) \end{aligned}$$

where $\frac{DF}{Dt}$ is known as the material or convective derivative.

Furthermore, we can evaluate $\frac{DF}{Dt}$ by expanding it into partial derivatives

$$\frac{DF}{Dt} = \frac{\partial}{\partial t}F_x(x, y, z, t) + \frac{\partial}{\partial t}F_y(x, y, z, t) + \frac{\partial}{\partial t}F_z(x, y, z, t) + \frac{\partial}{\partial t}F(x, y, x, t)$$

and by using the chain rule,

$$\frac{DF}{Dt} = \frac{dx}{dt}\frac{\partial F_x}{\partial x} + \frac{dy}{dt}\frac{\partial F_y}{\partial y} + \frac{dz}{dt}\frac{\partial F_z}{\partial z} + \frac{\partial F}{\partial t}$$

where the local flow of the fluids velocity components are:

$$v_x = \frac{dx}{dt}, \qquad v_y = \frac{dy}{dt}, \qquad v_z = \frac{dz}{dt}$$

Now, if we look at the divergence of $\vec{v}$

$$\vec{\nabla} \cdot \vec{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}$$

and multiplying the divergence of $\vec{v}$ by F gives,

$$\begin{aligned} (\vec{\nabla} \cdot \vec{v})\ F &= \frac{\partial v_x}{\partial x}F_x + \frac{\partial v_y}{\partial y}F_y + \frac{\partial v_z}{\partial z}F_z \\ &= v_x\frac{\partial F_x}{\partial x} + v_y\frac{\partial F_y}{\partial y} + v_z\frac{\partial F_z}{\partial z} \end{aligned}$$

Finally substituting this expression back into the material derivative above, we see that the material derivative takes the form

$$\frac{DF}{Dt} \;=\; (\vec{\nabla}\cdot\vec{v})\;F \;+\; \frac{\partial F}{\partial t}$$

As the function F is non-specific and subject to any physical quantity of choice, the material derivative is simply an operator that acts on F, in much the same way that ∇ or $\frac{\partial}{\partial t}$ operates on a function F.

Thus, the material derivative is given by

$$\frac{D}{Dt} \;=\; \vec{\nabla}\cdot\vec{v} \;+\; \frac{\partial}{\partial t}$$

1.3 Relevant Equations in Fluid Dynamics and Cosmology

On the scale of the universe, the distribution and continuous regularity of matter can be described by continuum mechanics and in particular, by using three important equations in fluid dynamics. In this section, the divergence theorem and the material derivative is all that is required to derive these three equations in their vector forms.

To put these equations into context [1], we define an ideal fluid by considering a small fluid element, suspended in a large, uniform, static medium:

- The fluid element is incompressible and cannot change its volume V as it moves
- The density ρ is constant and the same for all elements for all time t
- The force exerted over the surface of the element dS is

  $$\vec{F} \;=\; \vec{P}\,dS$$

  where the vector $\vec{P}$ is equivalent to $P\,\vec{n}$. P is a scalar function, and $\vec{n}$ points in the direction of the force.

From these properties, the three equations of interest are:
The **Continuity Equation** is concerned with the conservation of the rate of change of mass

$$\frac{D\rho}{Dt} \;+\; \vec{\nabla}\cdot\rho\,\vec{v} \;=\; 0$$

where $\vec{v}$, ρ is velocity and and density respectively.
Whilst the velocity field at any point in space is controlled by the **Euler Equation's of motion**

$$\frac{D\vec{v}}{Dt} \;=\; -\,\frac{1}{\rho}\,\vec{\nabla}\cdot\vec{P} \;-\; \nabla\phi$$

where $\vec{P}$, $\nabla\phi$ is the pressure and acceleration due to gravity respectively.
And lastly, the Gravitational Potential ϕ at each point is given by **Poisson's Equation**

$$\vec{\nabla}^2\phi \;=\; 4\pi G\,\rho$$

where G is the universal gravitational constant.

1.3.1 The Continuity Equation

The continuity equation is founded on the law of conservation, such that if a physical quantity is conserved then so is its rate. although the equation is dependent on mass, it can also be related to other physical quantities of interest. The continuity equation states that the rate at which mass enters a system must be equal to the rate at which mass leaves the system and the mass accumulation still within the system.

The equation is derived just from equating two ideas, such that the **total rate of mass decrease** is equivalent to the **mass flux out** of its surface.

The **total rate of mass decrease** is

$$-\frac{d}{dt}\int_m dm \;=\; -\int_V \frac{D\rho}{Dt}\,dV$$

where $\frac{D}{Dt}$ is the material derivative and m, ρ and V is mass, density and volume respectively.

Now consider a infinitesimal volume element $\vec{dr}\,\vec{dA}$. Using the chain rule, the rate of decrease in mass is equivalent to

$$\frac{dm}{dt} \;=\; \frac{dm}{dV}\frac{\vec{dr}}{dt}\vec{dA}$$

$$=\; \rho\,\vec{v}\cdot\vec{dA}$$

Hence, the **mass flux out** through the surface is

$$\int_A \rho\,\vec{v}\cdot\vec{dA} \;=\; \int_V \vec{\nabla}\cdot\rho\,\vec{v}\,dV$$

by use of the Divergence Theorem.

Finally, equating the **total rate of mass decrease** to the **mass flux out** through the surface gives

$$-\int_V \frac{D\rho}{Dt}\,dV \;=\int_V \vec{\nabla}\cdot\rho\,\vec{v}\,dV$$

Then dropping the volume integral signs

$$-\frac{D\rho}{Dt} \;=\; \vec{\nabla}\cdot\rho\,\vec{v}$$

and rearranging,

$$\frac{D\rho}{Dt} \;+\; \vec{\nabla}\cdot\rho\,\vec{v} \;=\; 0$$

is the continuity equation in vector form.

The material derivative term accounts for the mass accumulation within the system whilst the divergence term accounts for the flow of mass entering and leaving the system.

1.3.2 Euler's Equations of Motion

Euler's equations describe the rate of change velocity of a point moving along a path with respect to variations in its spacial coordinates x, y, z and time t. The equations are derived by equating all forces acting on a fluid element suspended in a medium under Newton's **second** and **third** laws of motion.

Consider a small fluid element with density ρ and volume V suspended under the weight of gravity g. Splitting the forces acting on the mass into components gives us:

- The force due to gravity acting on the mass

$$\vec{F_g} = \int_m \vec{g}\, dm$$

- The force due to the pressure of the surrounding fluid

$$\vec{F_S} = -\int_S \vec{P} \cdot d\vec{S}$$

  where $d\vec{S}$ points normal to the surface of the mass.

By Newton's **second** law, the net force acting on the mass can be expressed as the rate of change of momentum

$$\int_m \vec{a}\, dm = \int_m \frac{D\vec{v}}{Dt}\, dm$$

where $\frac{D\vec{v}}{Dt}$ is the material derivative.

By Newton's **third** law, the net force acting on the mass is the sum of the individual forces

$$\int_m \frac{D\vec{v}}{Dt}\, dm = \int_m \vec{g}\, dm - \int_S \vec{P} \cdot d\vec{S}$$

and using the divergence theorem on the surface term, and expressing the mass element dm in terms of ρdV, the net force becomes

$$\int_V \frac{D\vec{v}}{Dt} \rho\, dV = \int_V \vec{g} \rho\, dV - \int_V \vec{\nabla} \cdot \vec{P}\, dV$$

Finally, dropping the integral signs and dividing through by ρ, we find that

$$\frac{D\vec{v}}{Dt} = -\frac{1}{\rho}\, \vec{\nabla} \cdot \vec{P} + \vec{g}$$

Now, as $\vec{g}$ is a vector, we can define $\vec{g}$ as the gradient of a scalar

$$\vec{g} = -\vec{\nabla}\phi$$

where ϕ is a scaler field known as the gravitational potential.

Hence, Euler's equations of motion is

$$\frac{D\vec{v}}{Dt} = -\frac{1}{\rho}\, \vec{\nabla} \cdot \vec{P} - \nabla\phi$$

1.3.3 Poisson's Equation in Newtonian Gravity

Poisson's equation for gravity is very similar to Gauss's law in electrostatics, and expresses the gravitational potential in terms of a continuous density distribution.

To derive it, assume two large, spherical masses, m and M, forming a system under their own gravitational pull at a distance r apart. Newton's law of gravitation states that the acceleration experienced by the smaller mass due to the larger mass is,

$$\vec{g}(\vec{r}) \; = \; - \, \frac{G\,M}{\vec{r}^2}$$

By the shell theorem, the force experienced by the smaller mass m is exactly the same when its mass is condensed at a single point, as to when its mass is distributed evenly in all directions at the same distance r from M. Therefore we assume that mass m is evenly distributed around the larger mass M, and we compute the gravitational flux passing through it by integrating over the whole surface at distance r.

This gives us

$$\begin{aligned}\int_A \vec{g} \cdot d\vec{A} \; &= \; - \, \frac{G\,M}{r^2} \int_A dA \\ &= \; - \, \frac{G\,M}{r^2} (4\pi r^2) \\ &= \; - \, 4\pi G\,M\end{aligned}$$

Using the divergence theorem, the gravitational flux becomes

$$\int_A \vec{g} \cdot d\vec{A} \; = \; \int_V \vec{\nabla} \cdot \vec{g} \, dV$$

Provided we assume the spherical surface of the two masses are the same, and stating mass M in terms of density ρ and volume V as

$$M \; = \; \int_V \rho \, dV$$

We then substitute both these expressions back in for the gravitational flux, as above, giving us the expression

$$\int_V \nabla \cdot \vec{g} \, dV \; = \; - \, 4\pi G \int_V \rho \, dV$$

Finally, by dropping the integral signs, Poisson's equation for Newtonian gravity becomes

$$\vec{\nabla} \cdot \vec{g} \; = \; - \, 4\pi G \, \rho$$

Again, we can state $\vec{g}$ as the gradient of ϕ

$$\vec{\nabla}^2 \phi \; = \; 4\pi G \, \rho$$

which is Poisson's equation.

1.4 The Perturbed Fluid Equations

The cosmological principle says that the universe is the same, everywhere, on the significantly large scale. It is the idea that the distribution of matter throughout the universe is homogeneous and isotropic since the gravitational potential governing it must be uniform. This idea is a consequence of the big bang theory where all matter in the universe expanded from a singularity and was distributed evenly in a very high temperature and density state. Up until this point, we have assumed this property of the universe to be true. However, observational data supports the idea that the density distribution of the cosmos is very close to being the same in all directions (isotropic), and very close to being the same in all locations (homogeneous). The aim of studying cosmological irregularities is to understand the processes that caused the universe to depart from uniform density [2].

1.4.1 The Density Contrast

In studying how matter in the expanding universe responds to its own self-gravity, a linear solution for how the matter behaves can be found by expanding the Euler, Continuity and Poisson equations in terms of a dimensionless density perturbation.

The density contrast, also known as the fractional density perturbation, is a parameter used to model density fluctuations and their associated gravitational potential distributions.

The density contrast is defined as

$$\delta(\vec{x}) \;=\; \frac{\rho(\vec{x}) \;-\; \rho_0}{\rho_0}$$

where $\rho(\vec{x})$ is the density irregularity located at a point $\vec{x}$ in a smooth background of uniform density ρ_0 that corresponds to the perturbation $\delta(\vec{x})$, where $\delta(\vec{x})$ is a small dimensionless variable such that $\delta(\vec{x}) << 1$.

Rearranged we have

$$\rho(\vec{x}) \;=\; \rho_0(1 \;+\; \delta(\vec{x}))$$

where any particle located at a point of density fluctuation $\vec{x}$ will also experience an associated fluctuation in velocity $\vec{v}(\vec{x})$ and gravitational potential $\phi(\vec{x})$.

Similarly, like the density contrast, we take perturbations with respect to velocity $\vec{v}(\vec{x})$ and gravitational potential $\phi(\vec{x})$:

$$\vec{v}(\vec{x}) \;=\; \vec{v}_0(1 \;+\; \delta(\vec{x}))$$

$$\phi(\vec{x}) \;=\; \phi_0(1 \;+\; \delta(\vec{x}))$$

Now we have a full set of perturbations that can be applied to the Euler, Continuity and Poisson equations to better understand the irregularities in the structure of the universe.

1.4.2 Euler, Continuity and Poisson Perturbations

Substituting in the perturbations $\rho(\vec{x})$, $\vec{v}(\vec{x})$ and $\phi(\vec{x})$ gives us:

- Continuity

$$\frac{D\left[\rho_0(1 + \delta)\right]}{Dt} + \vec{\nabla}\cdot\left[\rho_0(1 + \delta)\right]\left[\vec{v}_0(1 + \delta)\right] = 0$$

- Euler

$$\frac{D\left[\vec{v}_0(1 + \delta)\right]}{Dt} = -\frac{1}{\left[\rho_0(1 + \delta)\right]}\vec{\nabla}\cdot\vec{P} - \vec{\nabla}\left[\phi_0(1 + \delta)\right]$$

- Poisson

$$\vec{\nabla}^2\left[\phi_0(1 + \delta)\right] = 4\pi G\left[\rho_0(1 + \delta)\right]$$

To simplify each equation, it is best to make use of the material derivative, and the chain rule in the form

$$\vec{\nabla}\cdot(\vec{x}\,\vec{y}) = \vec{x}\,(\vec{\nabla}\cdot\vec{y}) + \vec{y}\,(\vec{\nabla}\cdot\vec{x})$$

The perturbation of the **continuity equation** gives:

$$= \rho_0\frac{D\delta}{Dt} + (1 + \delta)\frac{D\rho_0}{Dt} + (1 + \delta)\vec{\nabla}\cdot\left[\rho_0\left(\vec{v}_0 + \delta\vec{v}_0\right)\right]$$

$$= \rho_0\frac{D\delta}{Dt} + (1 + \delta)\left[\frac{D\rho_0}{Dt} + \vec{\nabla}\cdot\rho_0\,\vec{v}_0 + \delta\,\vec{\nabla}\cdot\rho_0\,\vec{v}_0\right]$$

remembering that,

$$\frac{D\rho_0}{Dt} + \vec{\nabla}\cdot\rho_0\,\vec{v_0} = 0$$

then the perturbed continuity equation becomes

$$\rho_0\frac{D\delta}{Dt} + \delta\,(1 + \delta)\vec{\nabla}\cdot\rho_0\,\vec{v}_0 = 0$$

Euler's equation and Poisson's equation need a little bit more work than this. Consider Hubble's constant from the proportionality $\vec{v}_0 = H_0\vec{x}$. In particular, consider the sum of the partial components in directions x, y, z of the divergence of $\vec{v}_0$, and the perturbation of $\vec{v}_0$ with the divergence of $\vec{v}_0$ such that:

$$\vec{\nabla}\cdot\vec{v}_0 = \sum_{j=1}^{3}\frac{\partial}{\partial x_j}(H_0x_j) = 3H_0$$

and

$$\delta\vec{v}_0\,\vec{\nabla}\cdot\vec{v}_0 = \delta\sum_{ij=1}^{3}(H_0x_i)\frac{\partial}{\partial x_j}(H_0x_j) = H_0\,\delta\vec{v}_0$$

Perturbation of the **Euler Equation** gives:

$$\vec{v}_0 \frac{\partial \delta}{\partial t} \; + \; \delta \, H_0 \vec{v}_0 \;\; = \; - \, \Phi_0 \, \vec{\nabla} \delta$$

where the pressure term has been neglected, there is no pressure in space. The potential Φ_0 is continuous and constant with no fluctuations, hence its rate of change is zero. The analysis of the term containing $\vec{v}$ has the same story as in the continuity equation, where the first step is to use the material derivative.

For future analysis, the perturbation of the **Poisson Equation** isn't needed, but out of interest the result is:

$$\ddot{\delta} \; + \; 2H_0 \dot{\delta} \; = \; (4\pi G \rho_0 \; + \; c_s^2 \, \nabla^2)\delta$$

where c_s is the speed of sound, and $\dot{\delta} \; = \; - \, \vec{\nabla} \cdot \delta \vec{v}_0$

1.5 Section 1: Discussion

In this section, we have derived the Euler, Continuity and Poisson equations and perturbed them to an expanding background giving the equations in terms of the density contrast. Further development can be made to put the equations into co-moving coordinates but at this stage this is all we need for Section 2.

The point of Section 1 was to gain an understanding of fluid dynamics and how the analysis applies to cosmology, whilst also providing a layer of understanding for the material in certain parts of Section 2, and how this approach using vector calculus applies.

2 Galaxy Multipole Power Spectra Computations

2.1 Section 2: Introduction

Section 2 explores how these equations combine to give a theoretical model of the power spectrum that can be observed from distant galaxies. The power spectrum is the intensity distribution described by a source over a continuous range of values, in this case the wavenumber k. From this, we use the programming language Python to convolve it with an unbiased set of data, generated from actual observed data from a simulation suite called WizCOLA. This convolvement causes the theoretical data to be slightly less biased than it was before.

From the WizCOLA data suite we compute the chi-square by combining the convolution matrix with another function called delta, where delta is code that takes the difference between observed raw-data and the convolved theoretical model. The chi-square is then looped over seven different regions or fields in the night sky, where the observational data has been recorded from the WiggleZ Survey, measuring redshifts of power spectra from the Anglo Australian Telescope in New South Wales. With a few tweaks to the code, this result gives us the likelihood for each of the seven regions.

CosmoSIS is a parameter estimation package that takes the initial parameters and applies numerical methods to converge the likelihood to a minimum value and then returns the optimized parameters. Due to the complexity of the code, the the time taken to run the job is extremely long and therefore the code is executed in Apollo, which is a HPC environment at the University of Sussex. The optimized parameters are then plotted against each other using a Python Package called GetDist where the results are remarked on.

2.2 CAMB Interpolation from Cosmological Parameters

The first objective is to create a theoretical model of the power spectrum. To do so, we use a Python cosmological interpolation package to fix or fit a function to a given set of cosmological parameters and then incorporate the interpolation into the material from Section 1 to produce an array of power spectrum values with k dependence.

CAMB stands for Code for Anisotropies in the Microwave Background [3]. It is a cosmology code package written in Python for calculating CMB and matter power spectra, as well as general utility function for cosmological calculations where all input and output is in conformal time and units of mega-parsecs [4].

In Object Oriented Programming, the class

```
class camb.model.CAMBparams
```

is an object storing the parameters for a CAMB calculation, including cosmological parameters and settings for what to calculate. When a new object is instantiated, default parameters are set automatically, unless set otherwise.

From the parameters in the model, the first job is to create a spline interpolation of the power spectrum as a function of the wave number k, where k is in units of $hMpc^{-1}$. In order to set the appropriate parameters, we first assign them values and then override the original values in the model by defining them in a function that returns the spline interpolation. This is shown in the Python script below:

```python
# Parameters
H0 = 68
ombh2 = 0.005
omch2 =  0.05
As = 1E-9
ns = 0.9
f = 0.3

def generate_Pk(H0, ombh2, omch2, As, ns):
    pars = camb.CAMBparams()
    pars.set_cosmology(H0=H0, ombh2=ombh2, omch2=omch2)
    pars.set_dark_energy()
    pars.InitPower.set_params(As=As, ns=ns)
    pars.set_matter_power(redshifts=[0.44], kmax=0.5)

    # Linear Spectra
    pars.NonLinear = model.NonLinear_none
    results = camb.get_results(pars)

    # Create a Spline from the Matter Power Spectrum
    kh, z, Pk = results.get_linear_matter_power_spectrum()
    spline = InterpolatedUnivariateSpline(kh, Pk[0,:], ext = 'raise')

    return spline
```

Note that the module importations are already in place.

The modified parameters that have been set above are [5]:

`set_cosmology` : sets cosmological parameters in terms of their physical densities.
`ombh2` is the physical density in baryons
`omch2` is the physical density in cold dark matter
`H0` is the Hubble parameter in $km/s/Mpc$

`set_dark_energy` : automatically sets the dark energy parameters.

`InitPower.set_params` : sets the Initial-Power primordial power spec parameters.
`As` is the co-moving curvature power at k
`ns` is the scalar spectral index

`set_matter_power` : sets parameters for calculating matter power spectra and transfer functions
`redshifts` gives the array of redshifts to calculate
`kmax` is the maximum k to calculate

Finally, we use Scipys spline interpolation function:

`scipy.interpolate.InterpolatedUnivariateSpline` : returns a one-dimensional interpolating spline for a given set of data points.

2.3 The Theoretical Power Spectrum

In section 1 we derived the Continuity, Euler and Poisson equation and then perturbed them to an expanding background. Through further analysis and manipulation, it is possible to incorporate cosmological parameters such as f and Θ that change the physical properties and the behavior of these three equations in density and potential fluctuations. This is done by expanding on the acceleration due to gravity and the flow velocity terms, by defining gravity as $\vec{g} = \vec{\nabla}^2 f$ and the velocity divergence as $\Theta = \vec{\nabla} \cdot \vec{v}$.

For better intuition of these relationships, the gravitational potential is already defined by the negative gradient of $\vec{g}$. Therefore the rate of change of f, or how f grows and changes with distance, is described by the gravitational potential.

The velocity divergence on the other hand, is a simplifying assumption that says that all matter flow in gravitational wells has no curl. On a very large scale, most of the matter does tend to flow directly down to the centre, however, on smaller scales matter tends to off-shoot the centre on approach, similar to that of the orbit of comets in a solar system or the flow of water down a plughole. This assumption is then simply disregarding smaller scale effects.

These parameters are described by the equations:

$$f = \frac{d\ln D}{d\ln a} = \frac{d\ln D}{H dt}$$

$$\Theta = -Hf\delta$$

where f called the linear growth factor, $D(t)$ is called the linear growth function and δ is the density contrast.

2.3.1 Redshift Space Distortions

Due to the accelerating expansion of the universe, observed galaxies appear elongated along the line of sight from earth. The edges of the galaxies furthest away are observed to be receding faster than the edges closest to us, leading to the edges furthest away appearing more heavily red shifted than would normally be expected. This effect is known as the Kaiser effect and needs to accounted for in the theoretical model of the power spectrum.

The Kaiser effect described by

$$\delta_s = \delta - \frac{\Theta}{H} cos^2\theta$$

where θ is the angle of line of sight from earth, and δ is the density contrast, and δ_s is the density contrast with the redshifts accounted for.

Now if we set

$$\mu^2 = cos^2\theta$$

then, including the parameters above gives

$$\delta_s = \delta(1 + f\mu^2)$$

In exactly the same way, the expectation value of δ_s is evaluated as

$$\begin{aligned}\langle\delta_s(k)\delta_s(k')\rangle &= (2\pi)^3\delta(k + k')P_s(k)\\ &= \langle\delta(k)\delta(k')\rangle - \frac{2\mu^2}{H}\langle\delta(k)\Theta(k')\rangle + \frac{\mu^2}{H^2}\langle\Theta(k)\Theta(k')\rangle\end{aligned}$$

substituting for the velocity divergence, Θ

$$\begin{aligned}&= \langle\delta(k)\delta(k')\rangle + 2f\mu^2\langle\delta(k)\delta(k')\rangle + f^2\mu^2\langle\delta(k)\delta(k')\rangle\\ &= (1 + 2f\mu^2 + f^2\mu^2)\langle\delta(k)\delta(k')\rangle\end{aligned}$$

which gives us

$$\underbrace{\langle\delta_s(k)\delta_s(k')\rangle}_{P_s(k,\mu)} = (1 + f\mu^2)^2\underbrace{\langle\delta(k)\delta(k')\rangle}_{P(k)}$$

The power spectrum describes the intensity distribution of a signal as a function of frequency or wavelength over a continuous range. $P_s(k,\mu)$ is the power spectrum with redshift-space distortions as a function of $P(k)$ - the power spectrum with no redshift. $P_s(k,\mu)$ is the predicted result of the observed power spectrum. k is the wave number - $\frac{2*\pi}{\lambda}$ as we are dealing with k-space in measurements of Mpc^{-1} (inverse mega-parsecs). The relevance to PyCAMB with this result, is that we have already generated the power spectrum with no redshift in the code above. It is the matter power interpolation `Pk`.

2.3.2 Fourier Modes

As the observed power spectrum $P_s(k,\mu)$ depends on θ - the angle of line of sight from earth, $P_s(k,\mu)$ is therefore a periodic function. By the **Dirichlet conditions**, any continuous, periodic function can be evaluated using a Fourier series.

Fourier series expansion takes the form

$$F_n(x) = \sum_{m=0}^{\infty}(C_0 + C_1\cos(mx) + C_2\sin(mx))$$

The observed power spectrum $P_s(k,\mu)$ can therefor be evaluated as

$$P_s(k,\mu) = \sum_{m}^{\infty}P_m L_m(\mu)$$

Where P_m are the Fourier coefficients, these are constant or average values around which the power spectrum fluctuates with θ dependence. $L_m(\mu)$ is the Fourier Mode that is set as a function of μ. As such, we have assigned the Fourier modes to be such of a specific group of functions called the Legendre polynomials. By choosing to sum over Legendre polynomials, we have allowed for the exploitation of their useful properties.

2.3.3 Legendre Polynomials

Legendre polynomials have many applications in physics. In particular, they can be used as the coefficients in the expansion of a Newtonian potential, and occur when solving Laplace's equation and other related partial differential equations in spherical coordinates.

Legendre polynomials are the solutions to the Legendre equation, a second-order ordinary differential equation of the form

$$(1 - \mu^2)\frac{d^2y}{d\mu^2} - 2\mu\frac{dy}{d\mu} + n(n + 1)\,y = 0$$

were explicitly, a Legendre polynomial L_n is the solution for the corresponding integer n in the Legendre equation, where they can be solved using the standard power series method.

A property of the Legendre polynomials is that they are either symmetric or anti-symmetric for all μ, that is:

- For any **even** integer n, the polynomial L_n is **even**
- For any **odd** integer n, the polynomial L_n is **odd**

such that the following relationship holds

$$L_n(-\mu) = (-1)^n L_n(\mu)$$

The first three **even** Legendre polynomials are:

$$\begin{aligned} L_0(\mu) &= 1 \\ L_2(\mu) &= \frac{3}{2}\mu^2 - \frac{1}{2} \\ L_4(\mu) &= \frac{35}{8}\mu^4 - \frac{15}{4}\mu^2 + \frac{3}{8} \end{aligned}$$

Plotting the Legendre polynomials gives:

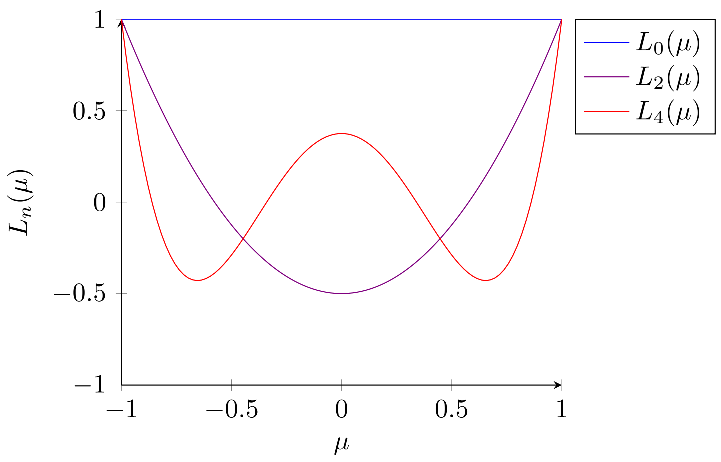

Figure 1: *Legendre polynomials between the range:* $-1 \leq \mu \leq 1$

The Legendre polynomials have another special property where they are orthogonal over $-1 \leq \mu \leq 1$ such that they satisfy the condition

$$\int_{-1}^{1} L_n(\mu) L_m(\mu) d\mu \;=\; \frac{2}{2n \,+\, 1}\,\delta_{mn}$$

where δ_{mn} is the Kronecker delta function thats equal to 1 if $m = n$ or 0 otherwise.

2.3.4 Constructing the Theoretical Model

In 2.3.2, we expressed the observed power spectrum as the Fourier series

$$P_s(k,\mu) \;=\; \sum_{n}^{\infty} P_n L_n(\mu)$$

where,

$$P_s(k,\mu) \;=\; (1 \,+\, f\mu^2)^2\, P(k)$$

Now using the Fourier series, and the Legendre property of orthogonality, we now evaluate the following integral and remark on the result

$$\begin{aligned}
\int_{-1}^{1} P_s(k,\mu) L_m(\mu)\, d\mu \;&=\; \int_{-1}^{1} (\,\sum_{n}^{\infty} P_n L_n(\mu)\,)\, L_m(\mu)\, d\mu \\
&=\; (\,\int_{-1}^{1} L_n(\mu) L_m(\mu)\, d\mu\,) \sum_{m=0}^{\infty} P_n \\
&=\; \frac{2}{(2n \,+\, 1)}\,\delta_{mn} \sum_{n=0}^{\infty} P_n
\end{aligned}$$

Rearranging, we find that

$$\begin{aligned}
\delta_{mn} \sum_{n=0}^{\infty} P_n \;&=\; \frac{(2n \,+\, 1)}{2} \int_{-1}^{1} P_s(k,\mu) L_m(\mu)\, d\mu \\
&=\; \frac{(2n \,+\, 1)}{2} \int_{-1}^{1} (1 \,+\, f\mu^2)^2\, P(k) L_m(\mu)\, d\mu
\end{aligned}$$

From this, we have obtained the sum of the Fourier coefficients multiplied by the Dirac delta function δ_{mn}. This is the sum of the coefficients upon which the power spectrum fluctuates around. n denotes the Legendre polynomial, for which we want the first three even numbers: $n = \{0, 2, 4\}$.

The power spectrum P_n then has three associated poles:

- The monopole: P_0
- The quadrapole: P_2
- The hexadecapole: P_4

where the sum of all three are referred to as the multipole power spectrum.

The code to create the theoretical model of the multipole power spectrum is as follows.

Defining the integrand:

```python
def Integrand(u, Pk, m, f):

    if m is 0:
        L = 1.
    elif m is 2:
        L = 3./2.*(u**2) - 1./2.
    elif m is 4:
        L = 35./8.*(u**4) - 15./4.*(u**2) + 3./8.
    else:
        raise RuntimeError

    return ((1. + f*(u**2))**2)*Pk*L
```

Computing the integral:

```python
def Integral(Pk, f, m):

    ks = np.arange(0.01, 0.51, 0.02)
    Integral = []
    for k in ks:
        Int, err = integrate.quad(Integrand, -1., 1., args=(Pk, m, f))
        Int = ((2.*m + 1.)/2.)*Int
        Integral.append(Int)

    return Integral
```

This computation gives back the integral depending on the result of the Legendre polynomial n. For each polynomial there are 25 data points in terms of k increasing in $0.02Mpc^{-1}$. What we want is a concatenated array of the monopole, quadrupole and hexadecapole power spectra - P_0, P_2, P_4.

Addition of multipoles:

```python
def Theory_Model(Pk, f):

    return np.array(Integral(Pk, f, 0) + Integral(Pk, f, 2) + Integral(Pk, f, 4))
```

This is the end result of the one-dimensional array of the theoretical power spectrum, consisting of 75 data points in order of P_0, P_2, P_4

2.4 WiggleZ Surveys and WizCOLA

The WiggleZ Dark Energy Survey is a large-scale redshift survey, which was carried out for over 276 nights between August 2006 and January 2011, measuring redshifts for almost 240,000 galaxies with a redshift range of $0.2 < z < 1.0$ in the night sky, at the Anglo Australian Telescope [6]. The aim was to try and understand more about the nature of dark energy by measuring the scale of baryon acoustic oscillations.

The sky distribution of the seven WiggleZ survey regions are compared to the coverage of the SDSS, RCS2 and GALEX data sets at the end of 2008. There are 7 fields with an average area of 140 square degrees totalling approximately 1000 square degrees. The seven fields in 00, 01, 03, 09, 11, 15, 22 hrs are shown here:

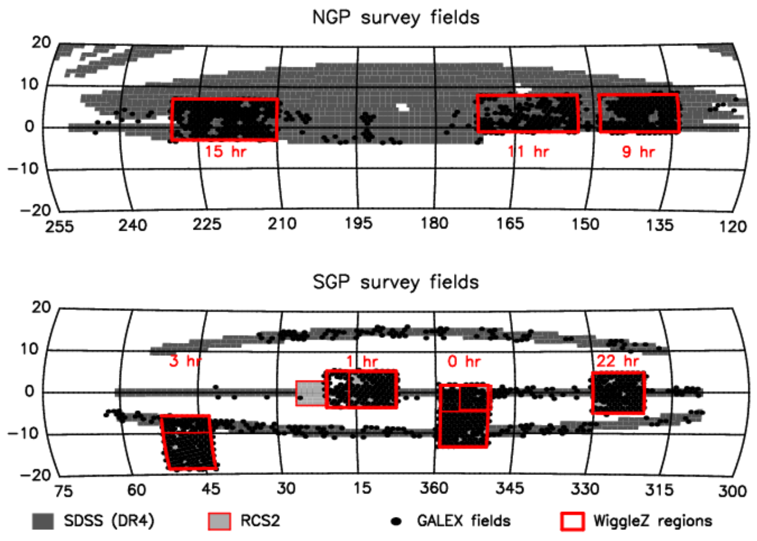

Figure 2: *The seven WiggleZ survey regions* [7]

The WizCOLA simulation suite contains the raw-data of the observed monopole, quadrupole and hexadecapole power spectrum simulated from the WiggleZ survey. The WizCOLA suite also contains the means-data consisting of the Covariance Matrix and Convolution Matrix data for redshifts in the range $0.2 < z < 0.6$. Both data sets consist of the seven data-text files, one for each field or region running from 00hr to 22hr, in the WiggleZ survey.

The observed, raw-data texts have columns: real, k, P_0, P_{0err}, P_2, P_{2err}, P_4, P_{4err}, where the first 15 rows are the power spectrum measurements for the mean of 0 realizations. These are the only data points that will be used and are the ensemble average over WizCOLA. k and P are measured in $hMpc^{-1}$ and $(Mpc\,h^{-1})^3$ respectively.

The means-data text files have columns: $\vec{i}$, $\vec{j}$, P, where the first 45x45 bins make up the Covariance Matrix and the last 75x75 bins make up the Convolution Matrix. Hence P_0, P_2 and P_4 are split between them accordingly in blocks of 15 for Covariance and 25 for Convolution.

2.5 The Covariance Matrix

The covariance matrix, also known as the dispersion matrix, is similar to that of a variance matrix, where variance is defined as

$$C^2 = \frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2$$

and gives a statistical measurement of how the data points x_i differ from the mean value $\bar{x}_j$ of the j_{th} variable.

Then covariance is defined as

$$C_{jk} = \frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)(x_{ik} - \bar{x}_k)$$

and gives a statistical measurement of how the data points x_i differ from the mean value $\bar{x}_j$ and $\bar{x}_k$ of the variables j_{th}, k_{th} variables respectively. Covariance is then a measure of the joint variability of two random variables or the extent to which two random variables differ from their mean value.

2.5.1 Neglecting Unreliable Data

Before putting this into practice, we first need to check the observed raw-data of the multiple power measurements. Graphing the 1hr field shows us two key points, with increasing k. Firstly, all multipoles conjugate together; and secondly, the error bars drop off very rapidly. This is consistent for all fields.

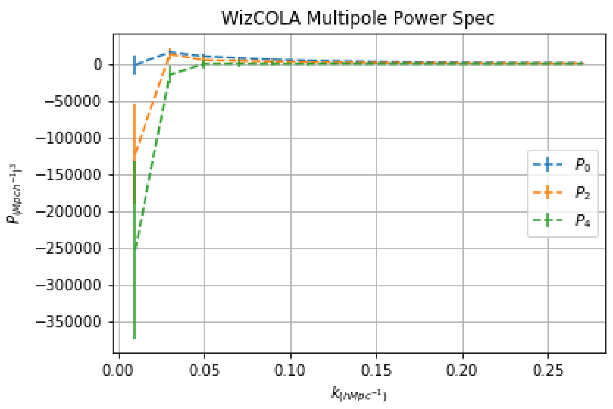

Figure 3: *Error analysis of the multipole power spectrum in the 1hr region*

For these reasons, the initial data point for all P_0, P_2, P_4 power spectra needs to be neglected for each field in WizCOLA and each field in the theoretical model, to maintain consistent and reliable data.

Note, in other texts the power spectrum is normally plotted against $\log k$ producing three lines running in parallel and sloping downwards. This plot is purely to emphasize the reasons for neglecting data.

2.5.2 Constructing the Covariance Matrix

In Python, we want to construct the covariance matrix from the raw-data text files. In order to keep all matrices consistent to the same $n \times n$ size, we need to neglect the first entry to all P_0, P_2, P_4 blocks for each field in both the i direction and the j direction.

One way to do this is to append each column entry, excluding the $\vec{j} = \{1, 16, 31\}$ entries, then delete the the rows $\vec{i} = \{1, 16, 31\}$. Note in the code below, the first row was never appended for simplicity and that the argument `data` is called from `np.loadtxt()` for each field or region:

```python
def Covariance_Matrix(data):

    d = data[:, 2]
    Covariance = []

    for n in range(1, 45):
        P0 = d[1 + 45 * n: 15 + 45 * n]
        P2 = d[16 + 45 * n: 30 + 45 * n]
        P4 = d[31 + 45 * n: 45 + 45 * n]
        P = np.concatenate((P0, P2, P4))
        Covariance.append(P)

    del Covariance[14]
    del Covariance[28]

    return np.array(Covariance)
```

2.5.3 Constructing the Inverse Covariance Matrix

In order to compute the chi-square, we need to construct the inverse covariance matrix. The chi-square in matrix form, is defined as

$$\chi^2 = \sum_{ij} \Delta_i \, C_{ij}^{-1} \, \Delta_j$$

where, C_{ij}^{-1} is the inverse covariance matrix.

Constructing the inverse covariance matrix is more complex then it might seem, as the difference in magnitude between data points is great enough that returning a function, such as `np.linalg.inv(Covariance)`, will not give back unique solutions to a system of linear equations. The pseudo inverse is a generalization of the inverse matrix that gives back an approximate solution to the inverse matrix when unique solutions cannot be found. The code below allows us to construct an inverse covariance matrix, whereby a condition number is implemented. The condition number is roughly the ratio between the two eigenvalues. If the condition number is smaller than a given value, then the proper inverse is computed. If the condition number is too large, then the Pseudo inverse is computed instead.

The inverse covariance matrix is defined here:

```python
def Inverse_Covariance_Matrix(data):

    Covariance = Covariance_Matrix(data)

    # compute eigenvalues and eigenvectors
    w, v = np.linalg.eig(Covariance)

    # construct the condition number
    Condition_No = abs(np.max(w) / np.min(w))

    # if condition number is too large then use the pseudo inverse,
    # otherwise use proper matrix inverse
    if Condition_No > 1E4:
        return np.linalg.pinv(Covariance)
    else:
        return np.linalg.inv(Covariance)
```

In order to check the accuracy of the Inverse Covariance Matrix, we simply compute the Identity Matrix from the dot product of the Convolution with its Inverse:

```python
def Identity_Covariance_Matrix(data):

    Covariance = Covariance_Matrix(data)
    Inverse = Inverse_Covariance_Matrix(data)

    return np.dot(Covariance, Inverse)
```

Printing the Identity Matrix gives back typical values of exactly 1.0 along the diagonal, and values ranging from 1×10^{-14} to 1×10^{-17} on either side.

2.6 The Convolution Matrix

The convolution of two functions, in the case of limits ranging from $-\infty$ to ∞, can be represented by

$$f * g \ = \int_{-\infty}^{\infty} f(t)g(t_0 - t)dt$$

The convolution matrix is a measure of the amount of overlap of all possible or probable data that could exist from initial conditions. It is all the possible scenarios of the multipole power spectrum convolved together, to give an unbiased representation of all spectra, similar to the result of the overlap or interference effect from a large number of sources.

Of course, the data provided by WizCola is not continuous. So the convolution matrix is really described by the discrete convolution

$$(f * g)[n] \ = \sum_{m=-M}^{M} f(n-m)g(m)$$

2.6.1 Constructing the Convolution Matrix

In constructing the convolution matrix, everything is appended. This is because the one-dimensional array `Theory_Model`, with 75 entries, needs to be dotted with the convolution matrix, giving another one-dimensional, vertical array upon which the required sections of rows will be extracted for further calculation.

The code for the convolution matrix is:

```python
def Convolution_Matrix(data):

    d = data[:,2]
    Convolution = []
    for n in range(0, 75):
        P = d[2025+(0 + 75*n) : 2025+(75 + 75*n)]
        Convolution.append(P)

    return np.array(Convolution)
```

2.7 Computing the Likelihood for all regions

So far, we have constructed the theoretical model, the inverse convolution matrix and the covariance matrix. The convolution matrix is the unbiased model where all power spectra is partially everywhere, with little to no differences. Now we need to construct delta, which is the difference between the observed raw-data of the power spectrum and the dot product of the convolution matrix with the theoretical model

$$\Delta_j \;=\; O_j \;-\; C_{ij} \cdot P_j$$

2.7.1 Constructing Delta

`Biased_Theory` in the Python code below, is a 1×75 array computed from the dot product of the 75×75 convolution matrix with the 1×75 theoretical model. The outcome of this new array is the product of a slightly biased prediction that accounts for survey geometry and incompleteness for each of the seven fields. This result then needs to be split accordingly into three, 1×14 long blocks of the new values of P_0, P_2 and P_4, where we have ignored the first entry for each multipole. Such that we have, new P_0 from the 2nd to 15th entry, new P_2 from the 27th to 40th entry and new P_4 from the 51st to the 65th entry. The final step is to concatenate these three blocks together, forming a 1×42 array, and subtract the observed raw-data from it. The observed raw-data is constructed in exactly the same way; a 1×42 sized data array, consisting of the three blocks of multipoles in increasing order.

The Python code for delta is here:

```python
def Delta(means, conv, Pk, f):
    Theory = Theory_Model(Pk,f)
    # convolve the power spectrum values in Theory_Model with convolution matrix
    Biased_Theory = np.dot(conv, Theory)

    # extract the values for each of P0, P2, P4 between k=0.03h/Mpc and k=0.29h/Mpc
    Theory_P0 = Biased_Theory[1:15]
    Theory_P2 = Biased_Theory[26:40]
    Theory_P4 = Biased_Theory[51:65]

    Predictions = np.concatenate((Theory_P0, Theory_P2, Theory_P4))

    return Predictions - means
```

Here, `conv` calls the convolution matrix for each of the 7 fields, and `means` is the observed raw-data where only the last 14 elements for each multipole is called and concatenated. *(see Appendix A)*

2.7.2 Constructing the Likelihood

The likelihood is a function which describes how likely an observed distribution is for a given model. For the case of a Gaussian distribution the relationship is

$$\mathcal{L} \;=\; \exp\left(\frac{-\chi^2}{2}\right) + K$$

where, $\mathcal{L}$ is the likelihood, χ is the chi-square and K are added corrections.

Due to large scales distributions in cosmology, the likelihood is normally represented in logarithmic form. Thus by ignoring K, and calculating the likelihood as its natural log, we see that

$$\ln(\mathcal{L}) \;\simeq\; -\,\frac{\chi^2}{2}$$

$$\simeq\; -\,\frac{1}{2}\sum_{ij}\Delta_i\, C^{-1}_{ij}\,\Delta_j$$

In the Python code, we want to calculate a list of all the likelihoods for all the seven regions. To do so, we call the power spectrum interpolator with the theoretical model in order to generate a for-loop that calls the Delta function, remembering to give all necessary arguments. The Delta function is then dotted with the inverse covariance matrix stored as `inv_Cs` for all regions, and then dotted again with the result. The list of all the likelihoods is then, half of the negative value of this likelihood, adding on the next, for all regions. The code for this is shown here:

```python
def All_Likelihoods(means, inv_Cs, convs, regions, H0, ombh2, omch2, As, ns, Pk, f):

    # call the linear power spectra for this set of cosmological parameters
    Pk = generate_Pk(H0, ombh2, omch2, As, ns)

    # generate theoretical power spectrum sampled on the correct k-grid
    Theory = Theory_Model(PK, f)

    All_Likelihoods = 0.

    # loop over all regions, computing the likelihood for each region
    for r in regions:

        Inv_C = inv_Cs[r]
        Conv = convs[r]
        M = means[r]

        Delta = Delta(M, Conv, Thoery)

        Likelihood = - np.dot(Delta, np.dot(Inv_C, Delta))/2.

        All_Likelihoods += Likelihood

    return All_Likelihoods
```

2.8 From Likelihoods to Parameter Optimization

From the multiple power spectra of a theoretical model combined with raw-data observed from the night sky, we have finally computed the likelihood depending on six cosmological parameters. These parameters are the baryon density $O_{m,b}h^2$, the cold dark matter density $O_{m,c}h^2$, the co-moving curvature power A_s at k, the scalar spectral index n_s, the Hubble constant H_0 and the linear growth factor f. Now the question is, for what combination, or combinations, of values of these parameters gives the best fit for the likelihood. To solve this, we need to use a code based algorithm that applies numerical methods to converge the likelihood to a minimum from a given set of initial parameters, thus returning the optimal parameter solutions. Of course, as there are so many variables with this problem, there is not going to be only one set of solutions to the optimal parameters as any combination can be close to the minimum likelihood in any number of ways. This is what CosmoSIS does.

2.8.1 CosmoSIS

CosmoSIS is a cosmological parameter estimation package. It is a framework for structuring cosmological parameter estimation in the form of calculation modules in Python. By taking a modular approach, larger and more complicated code is broken up into smaller parts, such that each module has a specific task to complete. CosmoSIS connects together existing code for predicting cosmic observables, and makes mapping out experimental likelihoods with a range of different techniques much more accessible [8] [9]. Although CosmoSIS can be run on any local machine, the job would take a considerable amount of time to complete due to the vast size of the calculation.

2.8.2 Apollo

Fortunately, the University of Sussex provides a HPC environment [10] that allows the use of parallel processing for running advanced application programs quickly and efficiently. This environment is called the Apollo HPC cluster. The objective now is to submit the job to Apollo to run in CosmoSIS, whereby Apollo then sends back the optimal parameters. To use Apollo, an account must be set up whereby a directory is added in your name. To access the directory on Apollo from Windows, one way is to download WinSCP which supports file transfer between local and remote computers. To submit jobs to Apollo, downloading Putty takes you to the terminal in Apollo, where jobs can be submitted via `qsub`, checking jobs via `qstat-u` and job deletion via `qdel`.

2.8.3 Submitting the Job

Apollo needs the following text files to run the job.

The Parameters File:

This contains the initial parameter values, where we have asked to find the minimum likelihood starting from three different combinations of parameters, that are simultaneously run together. *(see Appendix B.1)* It's interesting to note that the greater the offset of each parameter to their true value, the quicker the likelihood can converge to a minimum. But if the offset is too great, then it is possible for the likelihood to fluctuate away or even diverge.

The Pipeline File:

This file contains the parameters or instructions for how CosmoSIS should run. *(see Appendix B.2)* Inside the file, Emcee is the module algorithm that implements an Invariant Markov chain Monte Carlo Ensemble sampler. In order to more efficiently sample the parameter space, many samplers, called walkers, run in parallel and periodically exchange states [11]. The file also contains the algorithm parameters tolerance and maxiter. Reducing tolerance will improve the fit but will also increase the time taken. The pipeline is the sequence of calculations that computes a joint likelihood from the series of parameters [8]. `chain_out`is the file name returned by Apollo in text format and `verbosity` is the amount of data logged for debugging purposes.

The RSDPk File:

This is a text file consisting of Python code that imports CosmoSIS and all the code created throughout section 2 to compute the Likelihood. `RSDPk` also contains a Python class that imports all the data-text files (raw data and convolution/covariance matrices) and the initial parameters to be executed that are assigned to the parameter arguments in the code, that is, `H0, ombh2, omch2, As, ns, f`. *(see Appendix B.3)*

The CosmoSIS.sh File:

Finally, `CosmoSIS.sh` is the file that is submitted to Apollo. This file simply contains the paths of the other files ready for submission *(see Appendix B.4)*. Again, executing the job is done via: `qsub path\to\CosmoSIS.sh`

2.9 Plots with GetDist

After five hours' run time, the job completed and returned the `chain_out.txt` as expected. The `chain-out` file contains 10,000 combinations in a data array with columns: H_0, $\Omega_b h^2$, $\Omega_c h^2$, A_s, n_s, f and the likelihood. The best use of the data is to plot all the parameters against themselves to see how they correlate. This can be done using GetDist, a Python package used for analyzing Monte Carlo samples. Inconveniently, the format that GetDist takes is slightly different from the format of the output file from Apollo, such that GetDist needs a list of ones running all the way down in the first column, the likelihood in the second column and all other parameters shifted to the right. To fix this, we use Python to import the `chain_out` file, swap round the columns, and save it as a new file called `new_chain.txt` as shown here:

```python
chain_out = np.loadtxt('chain_out.txt')

H0  = chain_out[:,0]
Ob  = chain_out[:,1]
Oc  = chain_out[:,2]
As  = chain_out[:,3]
ns  = chain_out[:,4]
f   = chain_out[:,5]
lik = chain_out[:,6]

new_chain = []
for n in range(0,10000):
    new_chain.append([1.0, lik[n], H0[n], Ob[n], Oc[n], As[n], ns[n], f[n]])

np.savetxt('new_chain.txt', np.array(new_chain))
```

Next, the GetDist package needs to be installed and a folder created that consists of the `new_chain.txt` and another file called `new_chain.paramnames`. This file contains the list of cosmological parameter names in latex format for the graph axis labels. *(see Appendix B.5)* One way to show the plots is to run the GetDist GUI within PyCharm. To do this; open the PyCharm terminal in the bottom right hand corner, change directory to `venv\Scripts` by typing `cd venv\Scripts`. The GUI should open as a separate window where the path to the folder containing the two files can be uploaded.

The settings can be adjusted for different plots. For all the plots below, the analysis settings have been adjusted to ignore 45 percent of the chain at the start. This is because the chain converges towards the minimum likelihood, hence we want to ignore the parts of the chain further away. The plot we are most interested in is called the triangle plot and is shown below.

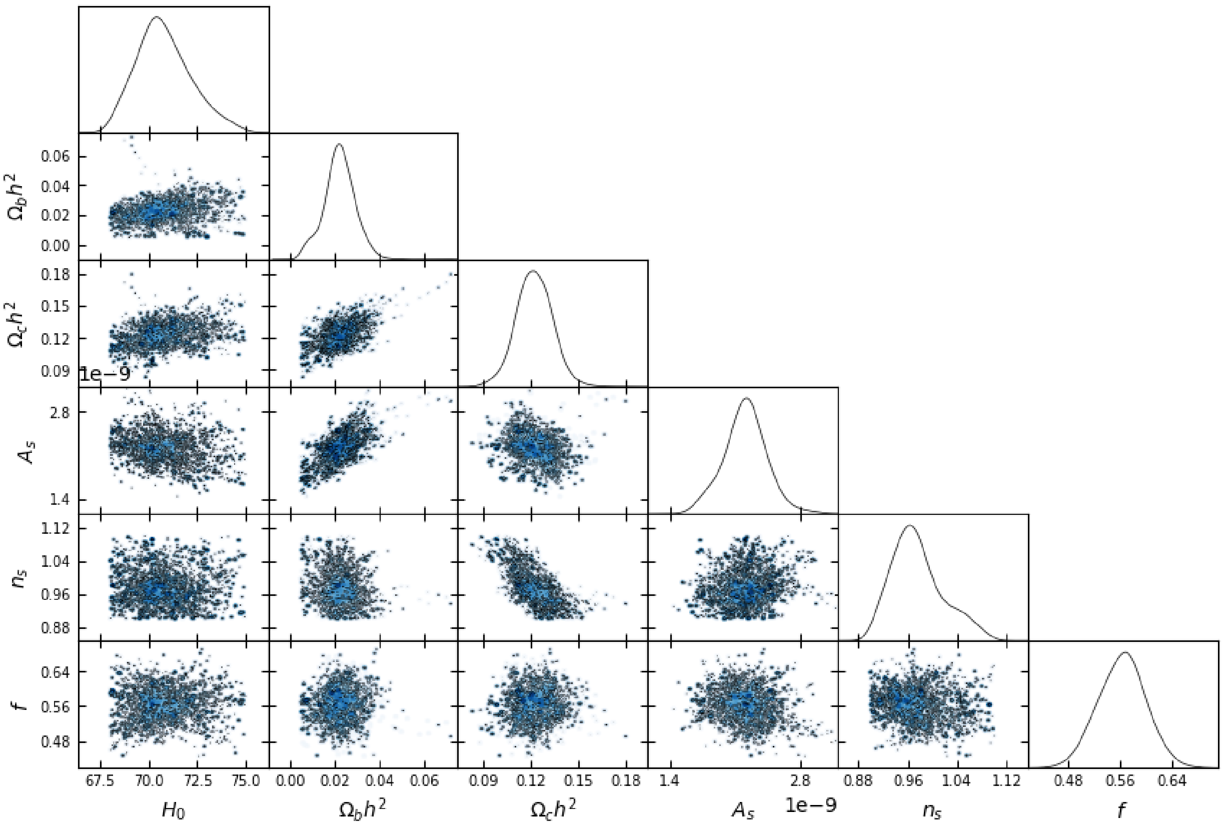

Figure 4: *Cosmological Parameters Triangle Plot*

For more plots. *see Appendix C*

3 Final Discussion

The triangle plot did not turn out in the best possible way, but this was to be expected, and the reasons why are explained in a moment. The plots should have shown multiple, defined, circular lines for each set of parameters. However, it turns out that each cosmological parameter plotted with the likelihood has mapped out nicely, where Hubble's constant has peaked at around 70.5, growth factor at 0.56, baryon density at 0.022, dark matter density at 0.122, co-moving curvature at 2.2×10^{-9} and spectral index at 0.97, all of which, are very close to their known values and with a smooth bell curve. This is also consistent with WizCOLA Suite, which uses values of $H_0 = 70.5$, $\Omega_b h^2 = 0.0456$, $\Omega_c h^2 = 0.027$, $A_s = 2.183 \times 10^{-9}$ and $n_s = 0.961$.

The reasons for the undefined plots are as follows. Firstly, the length of the chain determines the density of points on the graph. So had there been more samples run in Apollo, the plots would have turned out more defined, but only at the expense of longer run-time. Secondly, the plots are implemented by the physics brought together to create the theoretical model for the power spectrum. If the physics is incorrect or not enough physics is used, the plots will not correlate. In this paper, it is both.

What is wrong is Hubble's constant. In the continuation of Section 1, it would turn out, through more analysis, that Hubble's constant is equal to $2/3t_0$, where t_0 is the age of the universe. This is an admissible approximation, but in this model, all the other cosmological parameters do not exist, leaving inconsistencies in the model that computes the theoretical power spectrum.

In the case of where not enough physics is used in the model, one of the greatest drawbacks is that Einstein gravity has not been accounted for. From the beginning of Section 1 we remarked that it is possible to describe the universe without the need for complexity, such as general relativity. But if matter distorts the geometry of space-time then how would this affect the matter-density distribution in gravitational potential wells under the influence of a co-moving background? The answer is that the greater the complexity of the model, then the better the agreement with the convolved data and the better the results.

The purpose of this paper is to demonstrate the difficulties associated with modern Cosmology and how studies within the field are carried out. As it happens, there is no point taking a ruler out into space, so other ways have to be implemented in order to prove the next theory to be consistent with what is observed.

In this paper, we formulated a model of the density distribution against a smooth background, using vector calculus, to produce the perturbed Euler, continuity and Poison equations in an expanding background. We then used the Kaiser effect to describe the redshift distortions for the matter-density distributions to calculate the power spectrum, dependent on f and k. This power spectrum was then described by Fourier modes, evaluated under the influence of Legendre polynomials to give the power spectrum of three poles: the monopole, quadrapole and hexadecapole. This is what we called the theory model and was computed using CAMB and the Scipy interpolating spline. The data provided by WizCOLA from the WiggleZ survey was then used to construct the covariance and convolution matrices. After this, delta was constructed by combining the convolution matrix with the theory model to create a slightly biased version, from which the raw-data was then subtracted. Then we combined delta with the inverse covariance matrix and, from this, combined with delta again to create the likelihood for seven different regions in the night sky. After all this, with initial parameters given, the code was then passed to Apollo using CosmoSIS to find the optimal parameters for the minimum likelihoods. Apollo returned the chain, and GetDist was used to plot the results.

The idea was to create a model to replicate a system, in this case the power spectrum, by incorporating as much relevant physics as possible into the model. The model was then manipulated into a function which, given the right tools, can find a minimum value in order to give back the optimised parameters. The true parameter values were already known, as they were embedded into the WizCOLA simulation suite. The point was that if a new theory was incorporated into the model, and the theory is wrong, then the model is wrong and the code would give back bad results. In other words, if the theory is right, then you have discovered something new and you get a Nobel prize.

4 Acknowledgements

Sincere thanks to Dr David Seery, my supervisor, for his considerable effort throughout the year including the extra time he gave outside our weekly slots.

My thanks also to Roberto Scipioni for managing to resolve the unexpected technical difficulties with Apollo over the last month.

Finally, my thanks to Marion Rose for reading through this paper for corrections before the submission date.

Appendix A Data Extraction for the Means

```python
# import power spectrum measurements for each WiggleZ region
m1h = np.loadtxt('pkpole_wizcola_1hr_z0pt2_0pt6.dat')[1:15]
m3h = np.loadtxt('pkpole_wizcola_3hr_z0pt2_0pt6.dat')[1:15]
m9h = np.loadtxt('pkpole_wizcola_9hr_z0pt2_0pt6.dat')[1:15]
m11h = np.loadtxt('pkpole_wizcola_11hr_z0pt2_0pt6.dat')[1:15]
m15h = np.loadtxt('pkpole_wizcola_15hr_z0pt2_0pt6.dat')[1:15]
m22h = np.loadtxt('pkpole_wizcola_22hr_z0pt2_0pt6.dat')[1:15]

self.means = {'1h': np.concatenate((m1h[:, 2], m1h[:, 4], m1h[:, 6])),
              '3h': np.concatenate((m3h[:, 2], m3h[:, 4], m3h[:, 6])),
              '9h': np.concatenate((m9h[:, 2], m9h[:, 4], m9h[:, 6])),
              '11h': np.concatenate((m11h[:, 2], m11h[:, 4], m11h[:, 6])),
              '15h': np.concatenate((m15h[:, 2], m15h[:, 4], m15h[:, 6])),
              '22h': np.concatenate((m22h[:, 2], m22h[:, 4], m22h[:, 6]))}
```

Appendix B Apollo, Cosmosis and GetDist Files

B.1 Parameters File Contents

```
[cosmology]
H0 = 68 70.5 75
ombh2 = 0.005 0.0226643 0.5
omch2 = 0.05 0.113023 1.0
As = 1E-9 2.183E-9 5E-9
ns = 0.9 0.961 1.1
f = 0.3 0.56 0.7
```

B.2 Pipeline File Contents

```
[runtime]
sampler = emcee
[emcee]
walkers = 100
samples = 100
nsteps = 50
[maxlike]
tolerance = 1E-4
maxiter = 100
[pipeline]
modules = RSDPk
values = parameters.ini
likelihoods = RSDPK
[output]
filename = chain_out.txt
format = text
verbosity = debug
[RSDPk]
file = RSDPk.py
```

B.3 RSDPk File Contents

```python
from cosmosis.runtime.declare import declare_module
from cosmosis.datablock import names as section_names
import numpy as np
import chisq as chisq

class RSDPk:

    likes = section_names.likelihoods

    def __init__(self, config, name):
        # list of WiggleZ regions
        self.regions = ['1h', '3h', '9h', '11h', '15h', '22h']
        # import covariance and convolution matrices for each WiggleZ region
        d1h = np.loadtxt('covar_1hr_z0pt2_0pt6.dat')
        d3h = np.loadtxt('covar_3hr_z0pt2_0pt6.dat')
        d9h = np.loadtxt('covar_9hr_z0pt2_0pt6.dat')
        d11h = np.loadtxt('covar_11hr_z0pt2_0pt6.dat')
        d15h = np.loadtxt('covar_15hr_z0pt2_0pt6.dat')
        d22h = np.loadtxt('covar_22hr_z0pt2_0pt6.dat')

        self.inv_Cs = {'1h': chisq.Inverse_Covariance_Matrix(d1h),
                       '3h': chisq.Inverse_Covariance_Matrix(d3h),
                       '9h': chisq.Inverse_Covariance_Matrix(d9h),
                       '11h': chisq.Inverse_Covariance_Matrix(d11h),
                       '15h': chisq.Inverse_Covariance_Matrix(d15h),
                       '22h': chisq.Inverse_Covariance_Matrix(d22h)}

        self.convs = {'1h': chisq.Convolution_Matrix(d1h),
                      '3h': chisq.Convolution_Matrix(d3h),
                      '9h': chisq.Convolution_Matrix(d9h),
                      '11h': chisq.Convolution_Matrix(d11h),
                      '15h': chisq.Convolution_Matrix(d15h),
                      '22h': chisq.Convolution_Matrix(d22h)}

        # import power spectrum measurements for each WiggleZ region
        m1h = np.loadtxt('pkpole_wizcola_1hr_z0pt2_0pt6.dat')[1:15]
        m3h = np.loadtxt('pkpole_wizcola_3hr_z0pt2_0pt6.dat')[1:15]
        m9h = np.loadtxt('pkpole_wizcola_9hr_z0pt2_0pt6.dat')[1:15]
        m11h = np.loadtxt('pkpole_wizcola_11hr_z0pt2_0pt6.dat')[1:15]
        m15h = np.loadtxt('pkpole_wizcola_15hr_z0pt2_0pt6.dat')[1:15]
        m22h = np.loadtxt('pkpole_wizcola_22hr_z0pt2_0pt6.dat')[1:15]

        self.means = {'1h': np.concatenate((m1h[:, 2], m1h[:, 4], m1h[:, 6])),
                      '3h': np.concatenate((m3h[:, 2], m3h[:, 4], m3h[:, 6])),
                      '9h': np.concatenate((m9h[:, 2], m9h[:, 4], m9h[:, 6])),
                      '11h': np.concatenate((m11h[:, 2], m11h[:, 4], m11h[:, 6])),
                      '15h': np.concatenate((m15h[:, 2], m15h[:, 4], m15h[:, 6])),
                      '22h': np.concatenate((m22h[:, 2], m22h[:, 4], m22h[:, 6]))}

    def execute(self, block):
        H0 = block['cosmology', 'H0']
        ombh2 = block['cosmology', 'ombh2']
        omch2 = block['cosmology', 'omch2']
        As = block['cosmology', 'As']
        ns = block['cosmology', 'ns']
        f = block['cosmology', 'f']

        block[self.likes, 'RSDPK_LIKE'] = chisq.Chi_Square(self.means, self.inv_Cs,
                                                           self.convs, self.regions,
                                                           H0, ombh2, omch2, As, ns, f)

        return 0

    def cleanup(self):
        pass
# register this module with the cosmosis core
declare_module(RSDPk)
```

B.4 CosmoSIS.sh File Contents

```
#!/bin/bash
#$ -N CosmoSIS
#$ -pe openmp 16
#$ -q mps.q
#$ -jc mps.medium
#$ -R y
#$ -S /bin/bash
#$ -e /its/home/jb626/JackButcher/job-output/err_$JOB_NAME.$JOB_ID
#$ -o /its/home/jb626/JackButcher/job-output/out_$JOB_NAME.$JOB_ID
#####$ -M jb626@sussex.ac.uk
#######$ -m be
#$ -cwd
```

B.5 new_chain.paramnames File Contents

```
H_0 H_0
ombh_2 \Omega_b h^2
omch_2 \Omega_c h^2
A_s A_s
n_s n_s
f f
```

Appendix C Further Plots

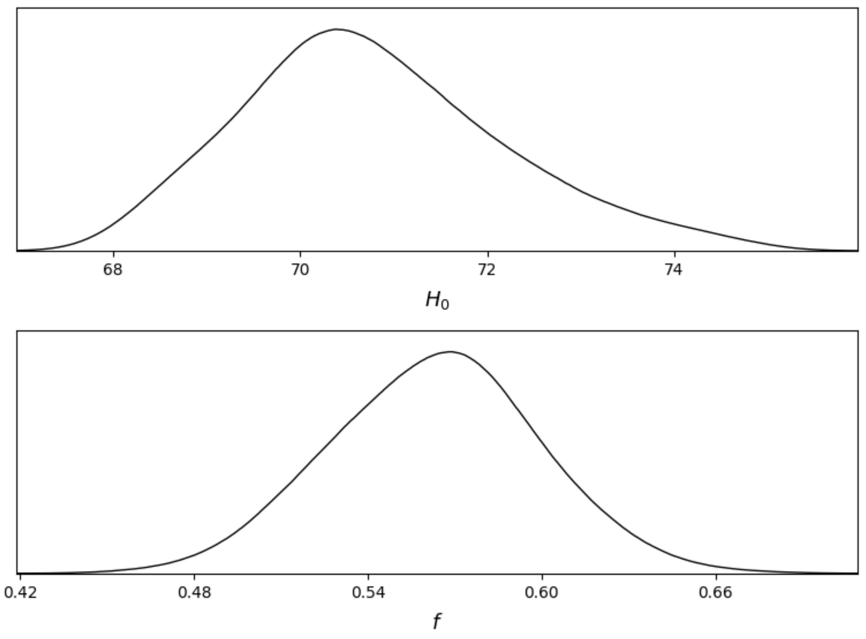

Figure 5: *Likelihood of H_0 and f*

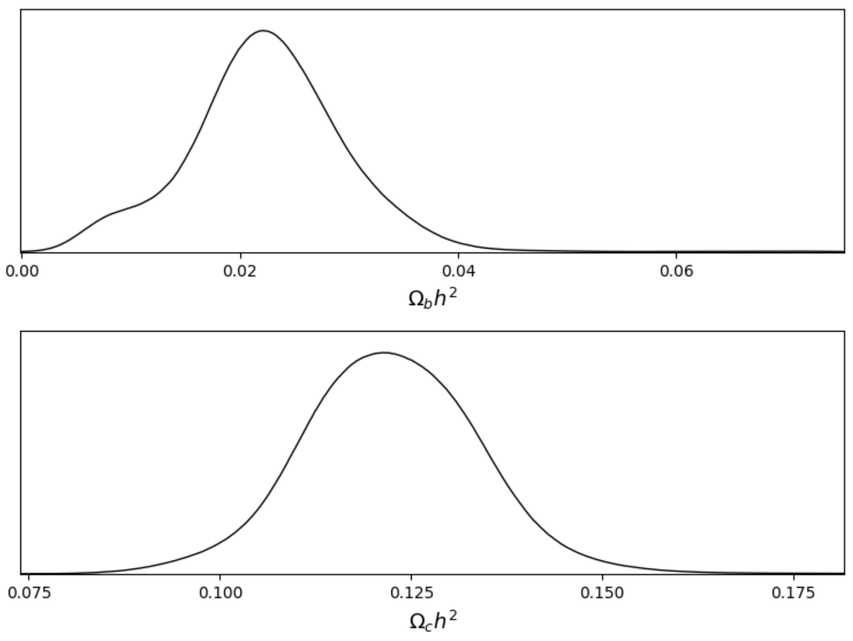

Figure 6: *Likelihood of* $\Omega_B h^2$ *and* $\Omega_C h^2$

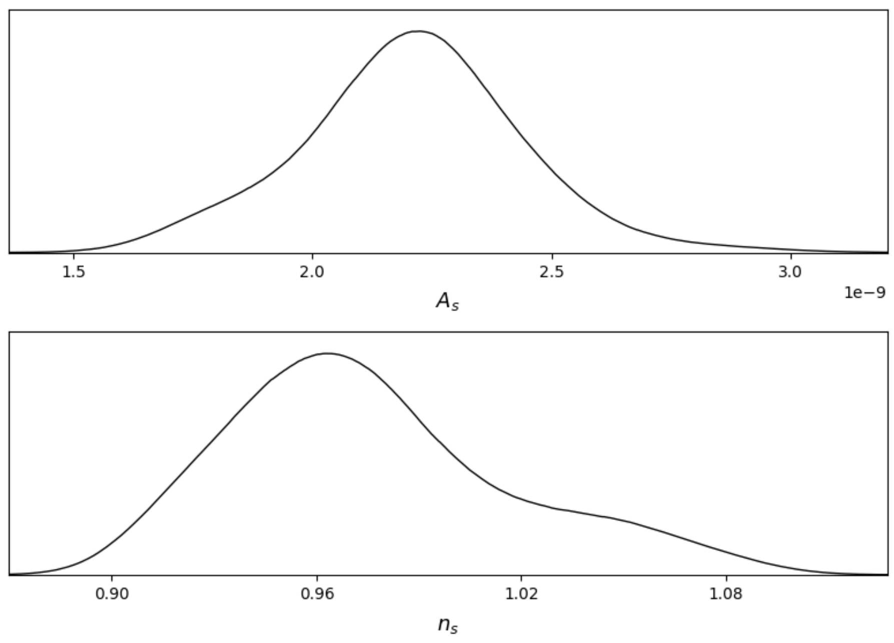

Figure 7: *Likelihood of* A_s *and* n_s

References

[1] Acheson, D. J. 1991. Elementary Fluid Dynamics. The Journal of the Acoustical Society of America, Volume 89, Issue 6. §§1.2-1.3.

[2] Peacock, John A. 1999. Cosmological Physics. Cambridge: Cambridge University Press. §§15.1-15.3

[3] Lewis.A and Challinor.A. 2017. CAMB ReadMe. [Online] Available at: `https://camb.info/readme.html`

[4] Lewis.A and Challinor.A. 2017. CAMB Notes. [Online] Available at: `https://cosmologist.info/notes/CAMB.pdf`

[5] Lewis.A and Challinor.A, 2017. camb.readthedocs. [Online] Available at: `http://camb.readthedocs.io/en/latest/index.html`

[6] Blake.C, Couch.W, Drinkwater.M and Glazebrook.K. 2011. WiggleZ Dark Energy Survey [Online] Available at: `http://wigglez.swin.edu.au/site/index.html`

[7] Blake.C, Couch.W, Drinkwater.M and Glazebrook.K. 2011. Figure: 1. [Online] Available at: `http://inspirehep.net/record/837534/plots`

[8] Joe Zuntz, Marc Paterno, Elise Jennings, Douglas Rudd, Alessandro Manzotti, Scott Dodelson, Sarah Bridle, Saba Sehrish and James Kowalkowski. 2014. CosmoSIS: modular cosmological parameter estimation. [Online] Available at: `https://arxiv.org/abs/1409.3409`

[9] Joe Zuntz, Marc Paterno, Elise Jennings, Douglas Rudd, Alessandro Manzotti, Scott Dodelson, Sarah Bridle, Saba Sehrish and James Kowalkowski. 2014. cosmosis/wiki/Home [Online] Available at: `//bitbucket.org/joezuntz/cosmosis/wiki/Home`

[10] IainStinson. 2009. University of Sussex IT Services Proposed HPC Service [Online] Available at: `http://www.sussex.ac.uk/its/pdfs/high_performance_proposal.pdf` Also see user guide: `https://info.hpc.sussex.ac.uk/hpc-guide/index.html#contents`

[11] R. Ellis, C. Burns, and R. Haynie. Carnegie Science Observatories. Carnegie Observatories, Pasadena, California, 91101 USA. Available at: `https://users.obs.carnegiescience.edu/cburns/ipynbs/Emcee.html`