

**Alma Mater Studiorum - Università di Bologna**

---

School of Science

Department of Physics and Astronomy

Master Degree Programme in Astrophysics and Cosmology

**New Machine Learning Algorithms  
to Study the Large-Scale Structure of  
the Universe**

Presented by:

**Claudia Crastolla**

Supervisor:

**Prof. Federico Marulli**

Co-supervisor:

**Dr. Francisco Villaescusa-Navarro**

---

Academic year 2024/2025

Graduation date March 27, 2026

# Abstract

The primary objective of this thesis is to propose and evaluate new artificial intelligence (AI)-enhanced methods for analyzing cosmological data. Specifically, we investigate the integration of large language models (LLMs) into the cosmological workflow to assess their capacity for automating complex research pipelines.

The LLMs utilized in this work are general-purpose and pre-trained on vast web-based data sets, without further fine-tuning on astrophysical data. By employing an agentic framework to facilitate interaction between the user and the models, the system was tasked with the execution of a clustering analysis, spanning code generation, debugging, and the critical interpretation of results for the computation of a two-point correlation function (2PCF) monopole and subsequent parameter inference from baryon acoustic oscillations (BAO).

The structure of this thesis is as follows:

Chapters 1 to 3 provide the theoretical foundations of modern cosmology and the statistical tools required to analyze the large-scale structure (LSS) of the Universe, including the 2PCF and parameter inference through the BAO peak. Chapter 4 introduces the fundamentals of LLMs and the emerging paradigm of agentic AI. Chapter 5 describes the technical implementation of the multi-agent pipeline and the prompt engineering strategies employed. Finally, Chapter 6 presents a detailed analysis of the results, comparing the performance of the autonomous agents against established cosmological benchmarks and discussing the future potential of AI-augmented research.

The results demonstrate that while currently available LLMs can rapidly implement functional software that would traditionally require significant human development time, several challenges remain regarding computational efficiency and physical rigor. While the 2PCF measurements and Markov Chain Monte Carlo (MCMC)-based parameter constraints were successfully executed, the agents encountered performance bottlenecks and produced constraints that were not always perfectly centered relative to established benchmarks.

Ultimately, this work suggests that while LLMs serve as powerful computational

engines, they require a human expert to act as a guide, providing the necessary physical intuition and theoretical validation to ensure scientific accuracy in cosmological studies.

# Contents

<b>Abstract</b>	<b>1</b>
<b>1 Chapter 1 - Introduction to Modern Cosmology</b>	<b>1</b>
1.1 A Brief Summary of Modern Cosmology . . . . .	3
1.1.1 Spacetime Geometry and the Metric Tensor . . . . .	3
1.1.2 The Expanding Universe . . . . .	5
1.1.3 Einstein's Equations . . . . .	9
1.2 Friedmann Models . . . . .	10
1.2.1 Thermal History of the Universe . . . . .	12
<b>2 Chapter 2 - Theoretical Foundation of Structure Formation</b>	<b>15</b>
2.1 Jeans Instability Theory . . . . .	15
2.2 The Correlation Function and the Power Spectrum . . . . .	17
2.2.1 The Two-Point Correlation Function . . . . .	17
2.2.2 The Power Spectrum . . . . .	19
2.3 The Einstein-Boltzmann Equations . . . . .	20
2.3.1 The Boltzmann Equation . . . . .	20
2.3.2 The Perturbed Einstein Equations . . . . .	22
2.3.3 The Einstein-Boltzmann Coupled System . . . . .	24
2.4 Modeling the Power Spectrum . . . . .	25
<b>3 Chapter 3 - Observational Probes of Large-Scale Structure</b>	<b>27</b>
3.1 Galaxy Clustering . . . . .	28
3.1.1 Galaxy Statistics . . . . .	29
3.1.2 Redshift-Space Distortions . . . . .	29
3.1.3 The Alcock-Paczynski Effect . . . . .	31
3.1.4 Baryon Acoustic Oscillations as a Standard Ruler . . . . .	31
3.2 Cosmological Parameter Estimation . . . . .	32

3.2.1	Likelihood Function . . . . .	32
3.2.2	Bayesian Analysis . . . . .	32
3.2.3	Markov Chain Monte Carlo . . . . .	33
3.3	Cosmological Simulations . . . . .	34
3.3.1	The Quijote Simulation Suite . . . . .	36
<b>4</b>	<b>Chapter 4 - Introduction to Large Language Models</b>	<b>37</b>
4.1	Machine Learning and Deep Learning Fundamentals . . . . .	38
4.1.1	What Is Machine Learning? . . . . .	38
4.1.2	Machine Learning Paradigms . . . . .	38
4.1.3	Deep Learning and Artificial Neural Networks . . . . .	40
4.2	The Transformer and Self-Attention . . . . .	42
4.2.1	Training of Large Language Models . . . . .	44
4.3	Agentic Artificial Intelligence and LangGraph . . . . .	44
<b>5</b>	<b>Chapter 5 - Cosmological Analysis with Agentic Artificial Intelligence</b>	<b>47</b>
5.1	Experimental Design . . . . .	47
5.2	Baseline Analysis . . . . .	47
5.2.1	Computation of the Two-Point Correlation Function . . . . .	48
5.2.2	Parameter Inference via Markov Chain Monte Carlo . . . . .	48
5.3	The Code . . . . .	49
5.4	Application Programming Interface for Large Language Models . . . . .	50
5.5	The Graph Architecture . . . . .	50
5.6	The Graphs . . . . .	51
5.6.1	Structured Output Schema . . . . .	51
5.6.2	The Graph State . . . . .	51
5.6.3	The Nodes . . . . .	52
5.6.4	Compiling the Graphs . . . . .	59
5.7	Prompt Engineering . . . . .	60
5.7.1	The Two-Point Correlation Function System Prompt . . . . .	61
5.7.2	The Markov Chain Monte Carlo Parameter Inference Prompt . . . . .	63
5.7.3	The Human Message Prompt . . . . .	64
<b>6</b>	<b>Chapter 6 - Results</b>	<b>69</b>
6.1	Evaluating the Agents' Performance . . . . .	69
6.1.1	The 2PCF Case Study . . . . .	69
6.1.2	Parameter Inference Analysis . . . . .	77

6.2 Discussion and Future Perspectives . . . . .	87
<b>Conclusions</b>	<b>91</b>
<b>Bibliography</b>	<b>93</b>



# 1

## Introduction to Modern Cosmology

Cosmology is the study of the cosmos on the largest scale possible, dealing with its origin, its ultimate fate, and everything in-between. Prior to the twentieth century, cosmology was primarily a philosophical discipline, until Einstein's revolutionary theory of General Relativity (GR) ([Einstein, 1917](#)) gave us a pathway for finding a scientific and testable theory of the Universe.

Our current understanding of cosmology rests on three fundamental pillars:

- The aforementioned theory of GR, which serves as the theory for gravity;
- The Cosmological Principle, stating that, on sufficiently large scales ( $\gg 100$  Mpc), the Universe is homogeneous and isotropic. This means that if we look at a sufficiently large portion of the universe, we know that any other portion of the same size will have identical properties. This gives a basic simplicity to the structure of the Universe that makes mathematical analysis possible.
- quantum field theory (QFT), describing quantum effects that underlie early-Universe phenomena such as particle production and the first stages of structure formation.

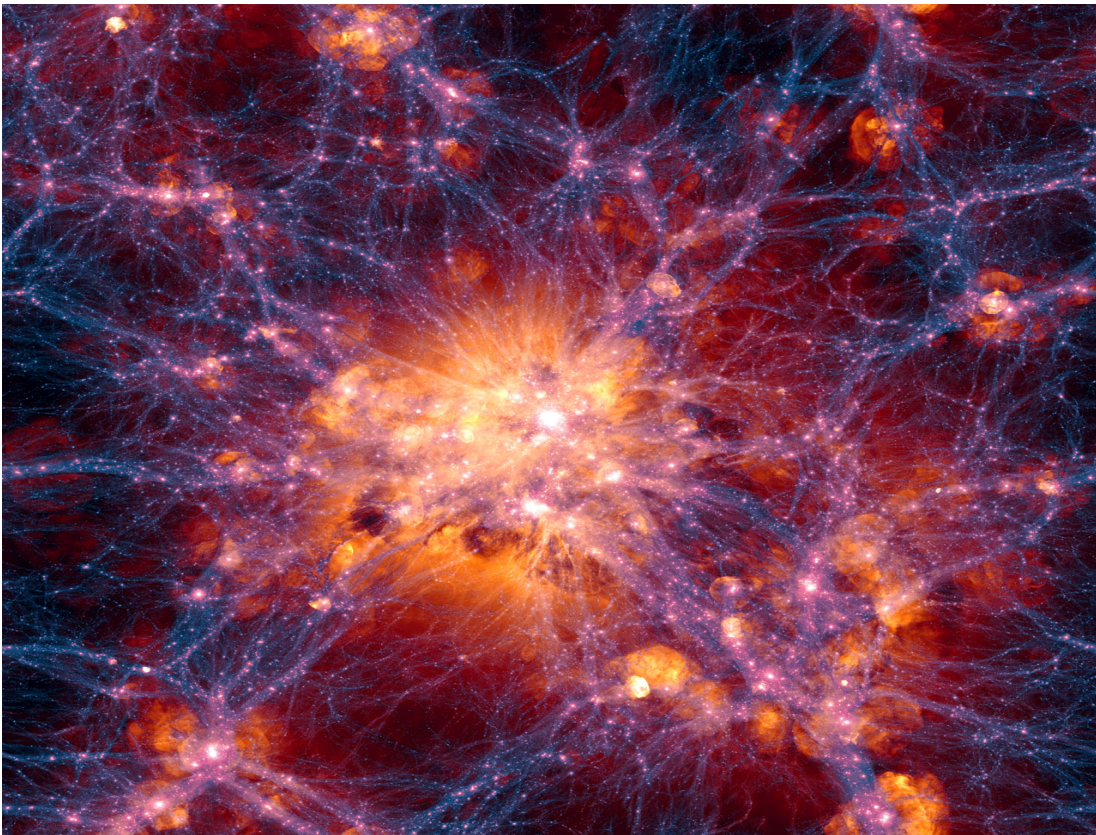
These premises underpin the Big Bang theory, which is strongly supported by observational evidence, including the Hubble diagram (demonstrating the expansion of the Universe), the primordial abundance of light elements, and the anisotropies observed in CMB, the afterglow of the Big Bang matter creation.

However, in order for this model to be consistent with observations, a few ingredients need to be introduced that go beyond the Standard Model of particle physics:

- Dark matter which, as inferred from measurable gravitational effects, constitutes the majority of matter present in the Universe;
- Dark energy, which dominates the energy budget of the Universe for most of its lifetime and determines its accelerated expansion;

- The inflation paradigm that is supposed to generate the initial perturbations from which cosmic structure formed.

Given all these ingredients, modern cosmology provides a theoretical framework that ties the early Universe to the structure we observe today through galaxy surveys and simulations: the cosmic web, a vast, filamentary network of matter as shown in Figure 1.1.



**Figure 1.1.** A detailed cosmological simulation from the Illustris project ([Vogelsberger et al., 2013](#)) showing the present-day "cosmic web" structure of the universe on large scales. The purple filaments represent the distribution of dark matter, whereas the orange patches represent the gas velocity field. The picture is centered on a massive galaxy cluster and covers a region about 300 million light-years across.

The following chapters will provide a concise overview on modern cosmology, covering its core concepts and the major unresolved problems that continue to drive current research.

## 1.1 A Brief Summary of Modern Cosmology

### 1.1.1 Spacetime Geometry and the Metric Tensor

The fundamental task in cosmology is explaining the large-scale structure of the universe. A key consideration, rooted in GR, is that distances are entirely shaped by the distribution of energy and matter. This requires us to compute proper expressions for cosmological distances, featuring a metric which should always be adapted to the case at hand.

#### The Metric Tensor

The metric tensor,  $g_{\mu\nu}$ , is used to model times and distances in the cosmological framework. In particular, it is necessary to define the infinitesimal interval between two events in spacetime:

$$ds^2 = g_{ij} dx^i dx^j, \quad (1.1)$$

which can be integrated along an object's trajectory in order to determine the total proper time or spatial distance it experiences between the events. In the context of special relativity, which describes a flat spacetime without gravity, we have the Minkowski's metric, which in Cartesian coordinates is expressed as follows:

$$g_{\mu\nu} = \eta_{\mu\nu} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (1.2)$$

by which Eq. (1.1) becomes:

$$ds^2 = -c^2 dt^2 + dx^2 + dy^2 + dz^2. \quad (1.3)$$

This interval can be of three different natures:

- Lightlike for paths followed by photons, where  $ds^2 = 0$  (null geodesics);
- Timelike when  $ds^2 < 0$ , corresponding to intervals between causally connected events;
- Spacelike when  $ds^2 > 0$ , corresponding to distances between space-separated events that are not causally connected to one another.

### The Friedmann-Lemaître-Robertson-Walker Metric

A homogeneous, isotropic, and expanding Universe can be geometrically described by the Friedmann-Lemaître-Robertson-Walker (FLRW) metric, which can be written in spherical coordinates as:

$$ds^2 = -c^2 dt^2 + a^2(t) \left[ \frac{dr^2}{1 - kr^2} + r^2(d\theta^2 + \sin^2 \theta d\phi^2) \right], \quad (1.4)$$

where  $c$  is the speed of light, and  $r$ ,  $\theta$ ,  $\phi$  are the spherical spatial coordinates. The coordinate  $t$  should be interpreted as the cosmic time, which is the time measured by a hypothetical comoving observer who is at rest with respect to the expanding universe. A particular attention should be given to the remaining two parameters:

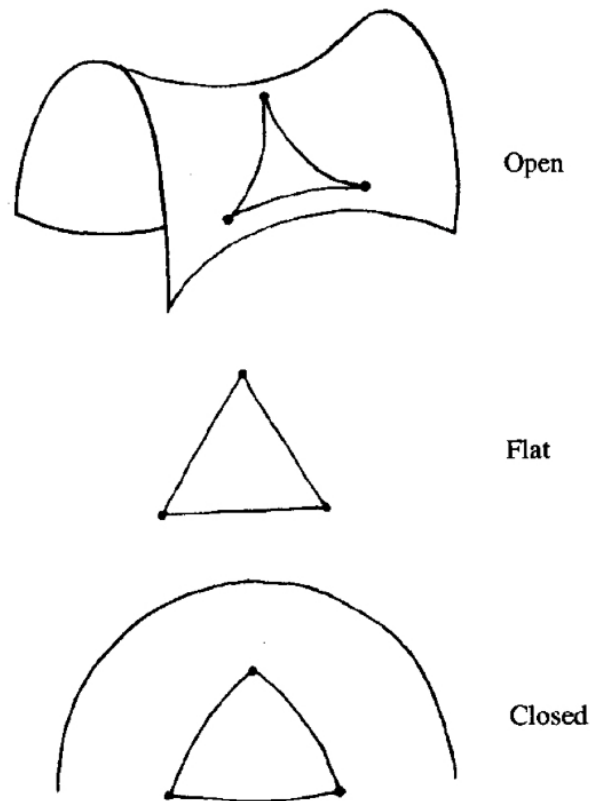
- The scale factor  $a(t)$ , which describes how distances in the universe expand over time;
- The curvature parameter  $k$ , which entirely determines the geometrical structure of the universe such that:

$k = 0$  corresponds to a flat Universe (Euclidean).

$k > 0$  corresponds to a closed Universe (spherical geometry).

$k < 0$  corresponds to an open Universe (hyperbolic geometry).

One of the most intuitive ways to illustrate different spatial geometries is through the properties of triangles, by comparing the sum of their inner angles, such as in Figure 1.2.



**Figure 1.2.** Representation of the three possible curved spaces in two dimensions. If we were to measure the sum of the internal angles of a triangle in each of these spaces, we would get exactly  $180^\circ$  in a flat space, greater than  $180^\circ$  in a closed space, and less than  $180^\circ$  in a curved one. Picture taken from (Coles & Lucchin, 2003).

### 1.1.2 The Expanding Universe

#### The Hubble Law

The measurements of time or length are complicated in GR, since spacetime is not absolute and the notion of simultaneity depends on the observer. For this reason, it is useful to consider the proper distance  $d_{pr}$ , defined as the distance between two events in the frame where they happen at the same time. Setting  $dt = 0$  and  $d\theta = d\phi = 0$  in the FLRW metric:

$$d_{pr} = a(t) \frac{dr}{\sqrt{1 - kr^2}} = a(t) f_k(r), \quad (1.5)$$

where:

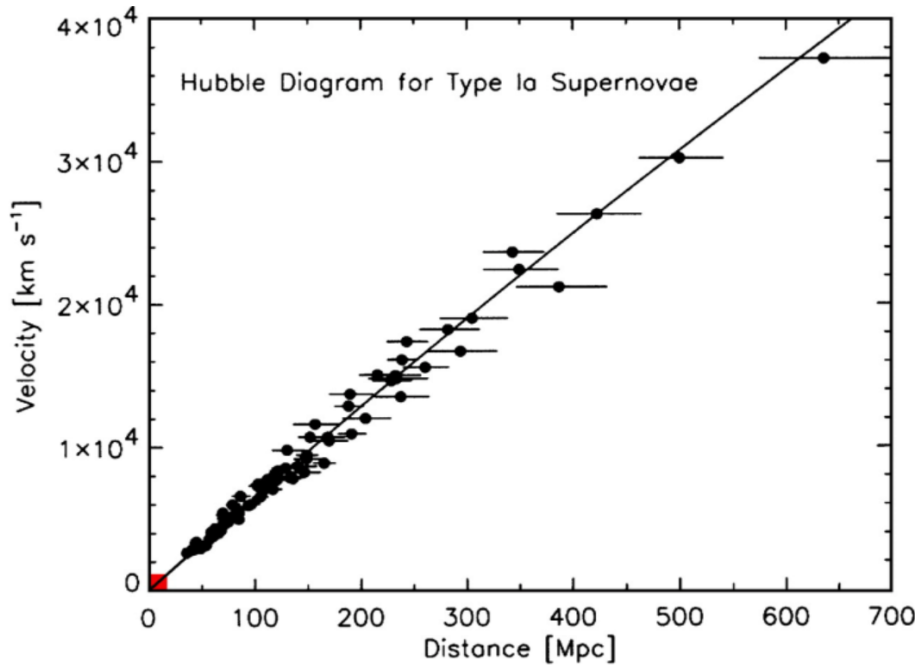
$$f(k) = \begin{cases} r & \text{when } k = 0 \\ \sin^{-1}r & \text{when } k = 1 \\ \sin^{-1}r & \text{when } k = -1 \end{cases}$$

The Hubble-Lemaître law, also known as the Hubble flow:

$$v_r = H(t)d_{pr} \quad (1.6)$$

tells us how fast an object at a given proper distance is moving away from any observer due to the expansion of space. It can be easily depicted in a Hubble diagram, like that in Fig 1.3, by plotting the velocity of astrophysical objects with respect to their distances from the observer.

The variable  $H(t)$  is known as the Hubble parameter, and the Hubble constant  $H_0$  is the present-day value for the Hubble parameter.



**Figure 1.3.** A Hubble diagram, depicting the Hubble-Lemaître law according to which the recessional velocity of large-scale structures is proportional to their distance. The straight line is calculated on a compilation of Type Ia supernovae, and its slope represents the Hubble constant. The original Hubble plot from 1929 is confined to the red rectangle on the bottom left. Picture taken from (Kirshner, 2004).

Another useful and frequently used distance variable is the comoving distance  $d_c$ , defined as the distance between two objects that remains constant through time as the objects are moving together with the Hubble flow.

It is related to the proper distance through:

$$d_c = d_{pr} \frac{a_0}{a(t)}, \quad (1.7)$$

with  $a_0$  being the scale factor at present time  $t_0$  in the comoving frame, which is conventionally taken to be unity.

### Cosmological Distances

In order to properly derive Hubble's law, it is necessary to know distances accurately. There are a few ways to define cosmological distances, beyond the ones already described (i.e. proper and comoving distances):

- The luminosity distance  $D_L$ , defined via:

$$F = \frac{L}{4\pi D_L^2}, \quad (1.8)$$

where  $F$  is the observed flux from a distant source and  $L$  is the source's intrinsic luminosity. A precise measurement for the luminosity distance can be derived from the apparent brightness of objects whose intrinsic luminosity is known (standard candles), like variable cepheid stars or Type Ia supernovae;

- The angular distance  $D_A$ , defined as:

$$D_A = \frac{l}{\theta}, \quad (1.9)$$

where  $l$  is the physical size of an object and  $\theta$  is the observed angular size on the sky.

It can be inferred from the observed angular size of an object of known physical size (standard rulers), examples being the BAO and the acoustic peaks in the CMB.

### The Cosmological Horizon

The cosmological or particle horizon is an important concept in cosmology that refers to the maximum distance in the universe from which an observer could possibly retrieve

information. This is a direct consequence of the finite speed of light as well as the finite age of the universe.

A commonly used expression for it is the particle horizon:

$$R_H(t) = a(t) \int_0^t \frac{cdt'}{a(t')}, \quad (1.10)$$

here defined as the maximum proper distance that a light particle could have traveled since the beginning of the universe up to time  $t$ .

This limit gives rise to the horizon problem: given that not all portions of the universe are causally connected with each other, it seems improbable that the Cosmological Principle would subsist.

## Redshift

Given the Hubble law, the observed wavelength of light emitted from faraway objects increases proportionally to their receding velocity, because of an effect similar to the Doppler shift. Thus, it is common to refer to faraway objects in terms of their observed redshift rather than their distance in spacetime.

Redshift is defined as:

$$z = \frac{\lambda_0 - \lambda_e}{\lambda_e}, \quad (1.11)$$

where  $\lambda_e$  is the wavelength of light at the moment of emission, while  $\lambda_0$  observed today.

In an expanding Universe described by the FLRW metric, the stretching of space affects not only the distance between galaxies, but also the wavelength of photons traveling through it along null geodesics. If a light signal is emitted at cosmic time  $t_e$ , when the scale factor was  $a(t)$ , and is observed today, then the comoving distance between the two events is:

$$\int_{t_e}^{t_0} \frac{cdt}{a(t)} = \int_0^r \frac{dr'}{\sqrt{1 - kr'^2}}. \quad (1.12)$$

If we consider another photon emitted at  $t_e + \delta t_e$  and observed at  $t_0 + \delta t_0$ , we have:

$$\int_{t_e}^{t_0} \frac{cdt}{a(t)} = \int_{t_e + \delta t_e}^{t_0 + \delta t_0} \frac{cdt}{a(t)}. \quad (1.13)$$

Subtracting the two expressions and expanding to first order gives:

$$\frac{\delta t_0}{a(t_0)} = \frac{\delta t_e}{a(t_e)}. \quad (1.14)$$

Since the wavelength of light is proportional to its period, this implies:

$$\frac{\lambda_0}{\lambda_e} = \frac{a(t_0)}{a(t_e)}. \quad (1.15)$$

Combining this with the definition of redshift, we find a useful relation between redshift and the scale factor:

$$1 + z = \frac{a_0}{a(t)}, \quad (1.16)$$

where  $a_0 \equiv a(t_0)$  is the scale factor today.

### 1.1.3 Einstein's Equations

The Universe is an expanding structure composed of massive objects with different formation histories, all of which can be modeled by GR and QFT.

One of the most important intuitions contained in GR is the unification of space and time into a single topological structure, now called spacetime, characterized by a metric tensor.

The curvature of spacetime is determined by the distribution of energy and momentum through a set of field equations, known as the Einstein's equations:

$$G_{\mu\nu} + \Lambda g_{\mu\nu} = 8\pi G T_{\mu\nu}, \quad (1.17)$$

where  $G \approx 6.67 \times 10^{-11} \text{ N} \cdot \text{m}^2/\text{kg}^2$  is the gravitational constant,  $G_{\mu\nu}$  is the Einstein tensor,  $T_{\mu\nu}$  is the stress-energy tensor,  $g_{\mu\nu}$  is the metric tensor defined in Eq. (1.1) and  $\Lambda$  is the cosmological constant.

The Einstein tensor is defined explicitly as:

$$G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu}, \quad (1.18)$$

where  $R_{\mu\nu}$  is the Ricci curvature tensor and  $R$  is the Ricci scalar, both of which depend on the metric tensor through its first and second derivatives.

The easiest way to read the Einstein equations is considering that  $G_{\mu\nu}$ , the Einstein tensor, encapsulates the gravitational effects and represents the geometric side of the equation, while the stress-energy tensor  $T_{\mu\nu}$  describes the distribution and flow of energy and momentum. Thus, the main takeaway one should get from them is that matter and energy are tightly connected to the geometry of spacetime – the content of the Universe

determines its curvature and hence its evolution.

## 1.2 Friedmann Models

In the 1920s, Alexander Friedmann derived a simple solution to Einstein's equations of GR under the assumption of the cosmological principle and of matter as a perfect fluid. The solution consists in two equations which currently serve as the main mathematical framework for studying the large-scale evolution of the Universe as a geometrical object.

### Friedmann Equations

The Einstein equations, not considering the cosmological constant, can be written as:

$$R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R = \frac{8\pi G}{c^4}T_{\mu\nu}. \quad (1.19)$$

The most relevant energy-momentum in cosmology is that of a perfect fluid:

$$T_{\mu\nu} = (p + \rho^2 c^2)U_\mu U_\nu - pg_{\mu\nu}, \quad (1.20)$$

where  $p$  is the pressure,  $\rho c^2$  is the energy density, and  $U_k$  is the fluid's four-velocity, defined as the first derivative of a particle's trajectory  $x^k(s)$ :

$$U_\mu = g_{\mu\nu} \frac{dx^\nu}{ds}. \quad (1.21)$$

Inserting Eq. (1.20) into Eq. (1.19), the two Friedmann equations can be extracted in their simplest form:

- The first Friedmann equation, which regards the expansion rate of the Universe, is given by:

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{8\pi G}{3}\rho - \frac{k}{a^2}, \quad (1.22)$$

where  $\rho$  is the total energy density of the universe;

- The second Friedmann equation, instead, describes how the expansion accelerates (or decelerates) based on the total matter content. It is given by:

$$\frac{\ddot{a}}{a} = -\frac{4\pi G}{3}(\rho + 3p). \quad (1.23)$$

### The Density Parameter

From the first Friedmann equation, one can derive the curvature of the Universe:

$$\frac{k}{a^2} = \frac{1}{c^2} \left( \frac{\dot{a}}{a} \right)^2 \left( \frac{\rho}{\rho_c} - 1 \right), \quad (1.24)$$

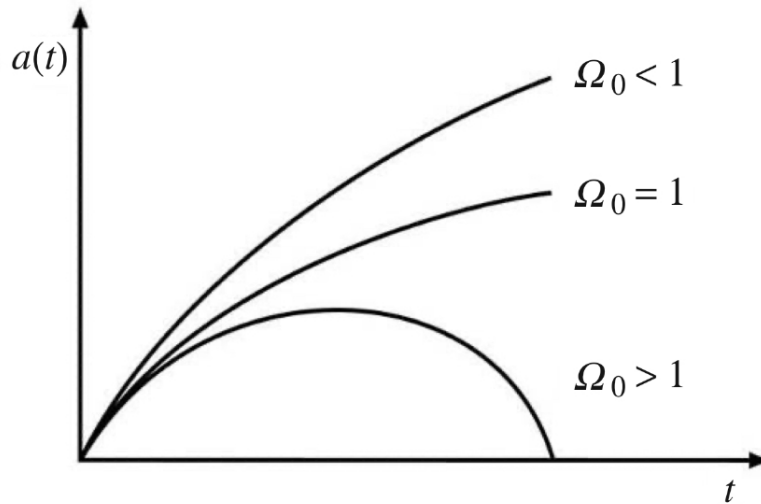
where

$$\rho_c = \frac{3}{8\pi G} \left( \frac{\dot{a}}{a} \right)^2 \quad (1.25)$$

is the critical density, which is the energy density required for a spatially flat Universe. This means that our spacetime is closed ( $k = 1$ ), flat ( $k = 0$ ) or open ( $k = -1$ ) according to whether the density parameter:

$$\Omega(t) = \frac{\rho}{\rho_c} \quad (1.26)$$

is equal to, less or greater than unity. Figure 1.4 shows the evolution of the scale factor depending on different values for the density parameter.



**Figure 1.4.** Evolution of the scale factor  $a(t)$  with three different hypothesized values for the density parameter  $\Omega_0$ , corresponding to closed, open and flat universes. In the case of a closed universe ( $\Omega_0 > 1$ ), the scale factor eventually reverses and shrinks back to zero, causing the so-called Big Crunch. Picture taken from (Coles & Lucchin, 2003).

## The Cosmological Constant

The cosmological constant can be incorporated into the Friedmann equations by modifying the energy-momentum part of the Einstein equation such that:

$$R_{ij} - \frac{1}{2}g_{ij}R = \frac{8\pi G}{c^4}\tilde{T}_{ij}, \quad (1.27)$$

where  $\tilde{T}_{ij}$  is now given by:

$$\tilde{T}_{\mu\nu} = T_{\mu\nu} + \frac{\Lambda c^4}{8\pi G}g_{\mu\nu} = -\tilde{p}g_{\mu\nu} + (\tilde{p} + \tilde{\rho}c^2)U_\mu U_\nu. \quad (1.28)$$

From this equation, we can extract the effective pressure  $\tilde{p}$  and the effective density  $\tilde{\rho}$  that we can substitute to the unmodified pressure and density terms in Friedmann equations (1.22) and (1.23), which retain their mathematical form.

The cosmological constant was first introduced by Einstein to allow for a static Universe, which was the prevailing belief at the time GR was laid down, but has now been disproven. Modern observations instead suggest that the cosmological constant is tied to a form of energy, the dark energy, that exists beyond ordinary matter and radiation. This component currently dominates the energy budget of the Universe and is thought to be the main cause of its accelerated expansion.

### 1.2.1 Thermal History of the Universe

#### Energy Density Evolution

Each component of the Universe has its own density evolution which follows the continuity equation:

$$\rho_w = \rho_{w,0} \left( \frac{a}{a_0} \right)^{3(1+w)}, \quad (1.29)$$

where  $w$  is the equation-of-state parameter, characterizing different components:

- $w = 0$  for matter;
- $w = \frac{1}{3}$  for radiation;
- $w = -1$  for the cosmological constant.

From Eq. (1.29), we see that the components' energy densities evolve differently with the scale factor, which sets the stage for a key transition in cosmic history.

This first transition, the matter-radiation equality, takes place when:

$$\rho_r(a_{eq}) = \rho_m(a_{eq}), \quad (1.30)$$

in which  $a_{eq}$  corresponds to redshift  $z_{eq} \sim 3200$ . It marks the starting point for baryonic structure formation, as matter starts to lose the radiation pressure that was first preventing gravitational collapse.

The most recent transition in the energy budget of the Universe occurs when the cosmological constant  $\Lambda$  overtakes matter. Referring to Eq. (1.29), again:

$$\rho_m(a_\Lambda) = \rho_\Lambda, \quad (1.31)$$

which happens very recently, at  $z_\Lambda \sim 0.3$ .

There is another key transition that occurs in-between the previous two, at  $z_{rec} \sim 1100$ , once the Universe cools down enough to allow for electrons and protons to combine into neutral hydrogen atoms. At this epoch, called the recombination epoch, photons decouple from matter entirely (meaning that their respective adiabatic evolution starts to be independent on the other) and begin free-streaming, creating the CMB.

### The Flatness Problem and the Inflation Paradigm

Defining the Hubble function as  $H = \left(\frac{\dot{a}}{a}\right)$ , and considering the relation between redshift and scale factor from Eq. (1.16), Eq. (1.22) can be rewritten as:

$$H^2(z) = H_0^2 \left[ \Omega_r(1+z)^4 + \Omega_m(1+z)^3 + \Omega_k(1+z)^2 + \Omega_\Lambda \right], \quad (1.32)$$

where  $\Omega_m$ ,  $\Omega_r$  and  $\Omega_\Lambda$  are the present-day parameters of matter, radiation and dark energy respectively, while  $\Omega_k = 1 - \Omega_r - \Omega_m - \Omega_\Lambda$  is the present-day curvature parameter, which essentially quantifies how far the universe is from being spatially flat.

Observational data such as that from the Planck mission ([Aghanim et al., 2020](#)) suggest that the total density parameter  $\Omega_{tot} = \Omega_r + \Omega_m + \Omega_\Lambda$  is extremely close to unity, implying a nearly flat universe. This poses the so-called flatness problem: it seems improbable, given the wide range of values  $k$  could assume, that the universe would choose to be perfectly balanced between open and closed geometries. This conundrum is one of the motivations for the cosmic inflation paradigm, first proposed by Alan Guth, according to which the universe underwent a short period of extremely rapid expansion in its early history ([Guth, 1981](#)). This expansion could have driven  $\Omega_k$  extremely close to zero, flattening the universe and preventing the emergence of a significant curvature over time. The inflation paradigm also solves the horizon problem: while true that regions in the universe are causally disconnected today, they were within each other's

horizons before inflation took place, and this early contact would explain the apparent homogeneity.

# 2

## Theoretical Foundations of Structure Formation

We have seen that a fundamental observation in modern cosmology is that the matter in the Universe is distributed in a complex pattern known as the cosmic web. This structure is the result of gravitational instability acting over billions of years: regions that began with a minuscule overdensity have since attracted more and more matter, collapsing to form the galaxies and massive clusters we see today. This process is also hierarchical, with small-scale perturbations collapsing first and then assembling into larger structures.

Since the underlying matter distribution (dominated by dark matter) cannot be observed directly, we use the distribution of galaxies and galaxy clusters as a visible tracer. This pattern is quantified statistically using tools like the correlation function and the power spectrum (which will be described in detail later). These tools provide the primary observational link between theory and data, allowing us to infer fundamental cosmological parameters.

In this chapter, I will briefly introduce the theoretical foundations that describe this entire process. First, I will give a description of the accepted theoretical models for LSS, starting with the classical Jeans Instability theory and building to the full relativistic framework of the Einstein-Boltzmann equations. Second, I will formally introduce the statistical tools used to quantify clustering, namely the two-point correlation function and the power spectrum. Finally, I will show how these pieces connect, describing how the Einstein-Boltzmann system is used to generate the final theoretical matter power spectrum,  $P(k)$ , which forms the basis for all LSS analysis.

### 2.1 Jeans Instability Theory

Studying the formation of astronomical objects such as stars and planets, astronomer James Jeans developed a theory that explains how structure naturally emerges as a result of gravitational instabilities that grow progressively with time. Starting from a homogeneous and isotropic fluid, small fluctuations in the fluid's characteristic quantities, such as density or velocity, can grow under the influence of gravity if they overcome internal

pressure forces. Once these perturbations exceed a certain critical scale—the Jeans' length  $\lambda_J$ , or equivalently  $k_J$  in Fourier terms—the gravitational collapse dominates and leads to clumping and structure formation. The simplest case is that of a collisional, non-relativistic fluid with perturbations so small ( $\delta \ll 1$ ) that any nonlinear effects can be safely neglected. Our system should satisfy a set of five equations:

- Continuity equation:  $\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0$
- Euler's equation:  $\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla) \vec{v} = -\frac{\nabla \mathcal{P}}{\rho} - \nabla \Phi$
- Poisson equation:  $\nabla^2 \Phi = 4\pi G \rho$
- Equation of state (for a perfect fluid):  $p = p(\rho, S)$
- Conservation of entropy:  $\frac{\partial S}{\partial t} + \vec{v} \cdot \nabla S = 0$

where  $\rho$ ,  $p$  and  $\vec{v}$  are, respectively, the density, pressure and velocity of the fluid,  $\Phi$  is the gravitational potential and  $S$  is the entropy. Assuming adiabatic conditions ( $\frac{dS}{dt} = 0$ ), such that the pressure is dependent only on the density ( $dp = v_s^2 d\rho$ ), we can study how the system evolves by adding small perturbations to some background solutions (indicated by the subscript "b"):

$$\begin{cases} \rho = \rho_b + \delta\rho \\ \Phi = \Phi_b + \delta\Phi \\ \vec{v} = H\vec{r} + \delta\vec{v}, \end{cases} \quad (2.1)$$

Here,  $H\vec{r}$  accounts for the background expansion of the Universe, with  $H$  being the Hubble parameter and  $\vec{r}$  the proper distance of a fluid element. Substituting Eqs. (2.1) into the governing fluid equations and linearizing by discarding all the nonlinear terms in the perturbations, it is possible to derive a wave equation for the evolution of Fourier modes of  $\delta = \frac{\delta\rho}{\rho_b}$ :

$$\ddot{\delta}_k + 2H\dot{\delta}_k + \left[ \frac{k^2}{a^2} v_s^2 - 4\pi G \rho_b \right] \delta_k = 0. \quad (2.2)$$

This is the dispersion relation governing the evolution of matter density perturbations  $\delta_k$  in comoving coordinates. It has different solutions depending on how they relate to the critical Jeans length:

- If  $k > k_J$  (or  $\lambda < \lambda_J$ ), we are in the sub-Jeans scale regime and the pressure of matter is strong enough to counteract gravitational collapse. The solution to the dispersion relation is a wave function whose amplitude does not change with time (an acoustic oscillation) and therefore does not contribute to structure formation;

- For super-Jeans scales,  $k < k_J$  (or  $\lambda > \lambda_J$ ), the pressure support is insufficient to counteract gravity and we have two linearly independent solutions for the wave equation: a growing mode and a decaying one.

Among the two super-Jeans solutions, the growing mode  $\delta_+$  (often denoted as the growth factor  $D_+(t)$ ) is of cosmological interest, as it is the one that actually contributes to structure formation. It can be written in the following way:

$$D_+(a) \propto H(a) \int_0^a \frac{da'}{(a'H(a'))^3}, \quad (2.3)$$

which is valid in a broad class of cosmological frameworks. It has no analytical solution in  $\Lambda$ CDM but can be solved numerically. For a matter-dominated (Einstein-de Sitter) Universe, it has the simple analytical solution:

$$D_+(a) \propto a \quad (\text{or } \delta_+(t) \propto t^{2/3}). \quad (2.4)$$

This result, which comes exclusively from classical gravity and fluid dynamics, and is valid for a homogeneous and isotropic Universe, constitutes the theoretical basis for how matter in the Universe clumps together to form structure. This Newtonian approach actually remains valid in GR and accurately describes matter clustering in the sub-horizon, linear regime, where it perfectly matches the particular solutions of the full Einstein-Boltzmann equations. However, the complete Einstein-Boltzmann treatment introduced in Section 2.3 becomes strictly necessary when evaluating structure formation in other regimes (such as super-horizon scales or the pre-recombination era), or when analyzing cosmologies where radiation cannot be neglected (e.g., massive neutrinos, interacting dark energy, or modified gravity).

## 2.2 The Correlation Function and the Power Spectrum

Before delving into the full relativistic description for the formation of LSS, it is appropriate to introduce the main statistical tools used to quantify and measure clustering, them being the correlation function and the power spectrum.

### 2.2.1 The Two-Point Correlation Function

In order to study clustering quantitatively, we can model astrophysical objects as discrete points distributed in 3-dimensional space. In this framework, we can investigate the likelihood of finding objects - such as galaxies, dark matter halos or quasars - within a

certain distance of another, compared to what would be expected from a completely random (Poisson) distribution of them.

The joint probability  $d^2P_{12}$  of finding an object in a small comoving volume  $dV_1$  and another in a volume  $dV_2$  at a given comoving distance  $\mathbf{r}_{12}$  is given by:

$$d^2P_{12} = n_V^2 [1 + \xi(r_{12})] dV_1 dV_2, \quad (2.5)$$

where  $n_V$  is the mean number density of the objects and  $\xi(r_{12})$  is the 2PCF. We can have three different qualitative behaviors for  $\xi(r_{12})$ :

- If  $\xi(r_{12}) > 0$  it indicates an excess probability, i.e. the objects are more clustered than in a random distribution;
- $\xi(r_{12}) < 0$  indicates a deficit within a distribution;
- $\xi(r_{12}) = 0$  indicates an absence of correlation, meaning that objects are randomly distributed on that scale.

In large-scale surveys,  $\xi(r)$  is estimated from catalogs using pair counts. A commonly used estimator is the Landy–Szalay estimator (?):

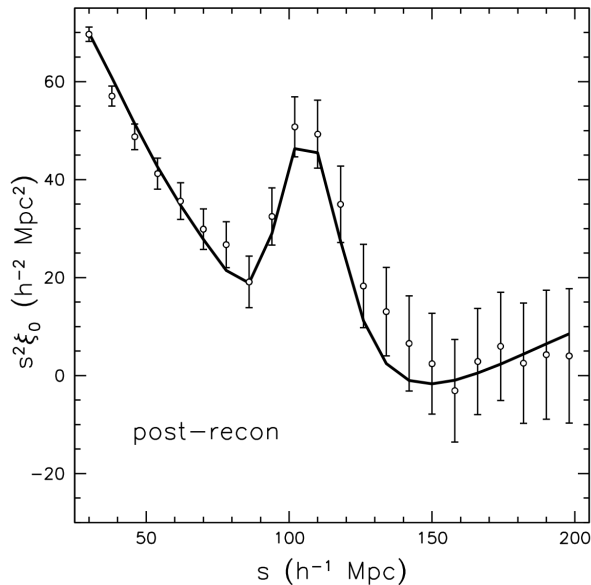
$$\hat{\xi}(r) = \frac{DD(r) - 2DR(r) + RR(r)}{RR(r)}, \quad (2.6)$$

where  $DD(r)$ ,  $DR(r)$  and  $RR(r)$  are suitably normalized (by the total number of object pairs in their respective catalogs) data–data, data–random and random–random pair counts in a given separation bin  $r$ .

Due to the isotropy of the Universe at the largest scales,  $\xi(r)$  is also isotropic (depending only on  $r$ ), however, observational effects like Redshift-Space Distortions (RSD), which will be explained more in detail later on, introduce a measurable anisotropy. This means the clustering also depends on the angle to the line of sight,  $\theta$ . To capture this, the 2PCF is decomposed into multipoles using Legendre polynomials,  $P_\ell(\mu)$ :

$$\xi(r, \mu) = \sum_{\ell=0,2,4,\dots} \xi_\ell(r) P_\ell(\mu), \quad (2.7)$$

where  $\mu = \cos(\theta)$ . The main components are the monopole ( $\ell = 0$ ), which is the angle-averaged clustering, and the quadrupole ( $\ell = 2$ ), which captures the primary anisotropy. A practical example of this measurement is illustrated in Figure 2.1, which displays the monopole  $\xi_0$  derived from survey data and clearly highlights the characteristic BAO peak.



**Figure 2.1.** An example taken from (Anderson et al., 2014) of a 2PCF monopole,  $\xi_0(s)$ , measured from a galaxy survey, plotted as  $s^2\xi_0(s)$  to enhance visibility. In this case,  $s$  denotes the observed separation of galaxies. The prominent feature near  $s \approx 100 h^{-1}\text{Mpc}$  is the BAO "standard ruler", a key probe of cosmic expansion. This specific measurement relies on the CMASS galaxy sample from the BOSS survey at an effective redshift of  $z = 0.57$ . The error bars denote the  $1\sigma$  uncertainties derived from mock galaxy catalogs.

## 2.2.2 The Power Spectrum

The power spectrum,  $P(k)$ , is another fundamental statistical tool in cosmology that indicates how much variance there is in density fluctuations on any given scale  $k$ .

Following the standard Fourier convention for the density contrast  $\delta(\vec{r}) = \delta\rho(\vec{r})/\rho_b$ :

$$\delta(\vec{k}) = \int \delta(\vec{r}) e^{-i\vec{k}\cdot\vec{r}} d^3r, \quad (2.8)$$

the power spectrum  $P(k)$  is formally defined via the ensemble average of the field's Fourier-space fluctuations:

$$\langle \delta(\vec{k}_1) \delta^*(\vec{k}_2) \rangle = (2\pi)^3 \delta_D(\vec{k}_1 - \vec{k}_2) P(k_1), \quad (2.9)$$

where  $\delta_D$  is the Dirac delta function. This definition reflects that, due to statistical homogeneity and isotropy, different  $k$ -modes are uncorrelated, and the power spectrum depends only on the magnitude  $k = |\vec{k}|$ .

The power spectrum is the Fourier transform of the 2PCF  $\xi(r)$  (a relationship known as the Wiener-Khinchin theorem):

$$\xi(r) = \int \frac{d^3k}{(2\pi)^3} P(k) e^{i\vec{k}\cdot\vec{r}} \longleftrightarrow P(k) = \int d^3r \xi(r) e^{-i\vec{k}\cdot\vec{r}}. \quad (2.10)$$

Because of isotropy, the previous 3D integral simplifies to the 1D form:

$$\xi(r) = \frac{1}{2\pi^2} \int_0^\infty k^2 P(k) \frac{\sin(kr)}{kr} dk. \quad (2.11)$$

While  $\xi(r)$  and  $P(k)$  encode the same statistical information, the power spectrum is the preferred quantity in theoretical cosmology. This is because in the linear regime, different Fourier modes  $\vec{k}$  evolve independently, making  $P(k)$  far simpler to predict than  $\xi(r)$ , which mixes information from all scales. The theoretical matter power spectrum,  $P_m(k)$ , is the direct output of solving the coupled Einstein-Boltzmann equations, which will be introduced in the next section.

## 2.3 The Einstein-Boltzmann Equations

The Jeans instability provides the classical foundation for gravitational collapse. However, to accurately describe structure formation in an expanding universe containing multiple, interacting components (like photons, baryons, and dark matter) and relativistic species (like photons and neutrinos), we must move from this Newtonian fluid description to a full general relativistic and kinetic theory. In this framework, a self-consistent description of structure formation requires following the evolution of small perturbations to both the geometry (the metric) and the matter content of the universe. In general, each species is described by a phase-space distribution function  $f(\vec{x}, \vec{p}, t)$ , which gives the number of particles per unit physical volume and momentum-space volume, and evolves with time.

### 2.3.1 The Boltzmann Equation

The Boltzmann equation describes the behavior of matter and radiation in the Universe. It is a powerful tool from statistical mechanics that tracks the time evolution of the distribution function  $f(x, p, t)$  for each particle species; more precisely, it tells you how many particles with a given momentum  $p$  are at a given position  $x$  at time  $t$ .

The most general form of the Boltzmann equation in an expanding FLRW background

is:

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{dx^i}{dt} \frac{\partial f}{\partial x^i} + \frac{dp^i}{dt} \frac{\partial f}{\partial p^i} = C[f], \quad (2.12)$$

where:

- The LHS describes how the distribution function changes as particles simply travel freely through spacetime. The paths of the particles are dictated by gravity (geodesics), so this term is where the metric ( $g_{\mu\nu}$ ) from Einstein's equations has its effects;
- The RHS  $C[f]$  is the collision term. It describes how the distribution function changes due to particle interactions, such as scattering, annihilation or decay.

In the case of a homogeneous universe, this simplifies significantly, as  $f$  depends only on the magnitude of momentum  $p$  and time  $t$ . The equation simply describes the dilution of particles by volume expansion ( $n \propto a^{-3}$ ) and the redshifting of their momentum ( $p \propto a^{-1}$ ). However, the situation gets more complicated if we were to add anisotropies in the distribution of the various particle species. Mathematically speaking, this consists in expanding the distribution functions around a static background, adding small perturbative terms to it:

$$f(\vec{x}, \vec{p}, t) = f_0(p, t)[1 + \mathcal{F}(\vec{x}, \vec{p}, t)], \quad (2.13)$$

where  $f_0(p, t)$  is the isotropic background (e.g., Bose-Einstein or Fermi-Dirac) distribution and  $\mathcal{F}$  encodes the perturbation.

In this framework, any perturbation  $\mathcal{F}(\vec{k}, \hat{p}, t)$  has a dependence on momentum-direction  $\hat{p}$  (the anisotropy). We can decompose this angular dependence by expanding it in a series of Legendre polynomials,  $P_\ell(\mu)$ , where  $\mu$  is the cosine of the angle between the wavevector  $\vec{k}$  and the particle's direction  $\hat{p}$ . There is a crucial difference between the Boltzmann treatment of baryonic matter and that of radiation, which are the two main factors that influence the LSS:

- Radiation: Photons and neutrinos are relativistic and free-streaming, making their distribution intrinsically anisotropic (direction-dependent). Consequently, higher-order moments like the quadrupole  $\ell = 2$ , which represents anisotropic stress, are non-zero and cannot be ignored unless in specific cases;
- Matter: all particles at a single point are moving with the same bulk velocity. They are "cold," so they do not have random, relativistic motions in different directions, and the only relevant moments are the monopole and the dipole, which

correspond to the standard continuity and Euler equations presented in Section 2.1.

These perturbations in the matter and radiation are directly coupled to the metric perturbations ( $\Phi$  and  $\Psi$ ) through Einstein's field equations.

### 2.3.2 The Perturbed Einstein Equations

The Boltzmann equations for each component describe how the matter and radiation fluids respond to the curvature of spacetime. However, they do not describe how spacetime itself is curved by the presence of that matter and radiation. This is the role of Einstein's field equations, first presented in Section 1.1.3.

To describe the formation and evolution of structures, one introduces small perturbations,  $h_{\mu\nu}$ , to the homogeneous and isotropic background metric,  $\bar{g}_{\mu\nu}$ :

$$g_{\mu\nu} = \bar{g}_{\mu\nu} + h_{\mu\nu}. \quad (2.14)$$

In the context of a spatially flat FLRW universe, the perturbed metric (in a general gauge) can be written as:

$$\begin{cases} g_{00}(t, x) = -1 + h_{00}(t, x) \\ g_{0i}(t, x) = a(t)h_{0i}(t, x) = a(t)h_{i0}(t, x) \\ g_{ij}(t, x) = a^2(t)[\delta_{ij} + h_{ij}(t, x)], \end{cases} \quad (2.15)$$

Just as we perturbed the distribution function, we give a similar treatment to Einstein's equations by linearizing them around a homogeneous background. This separates the equations into a "zeroth-order" part (the homogeneous background, which gives the Friedmann equations) and a "first-order" part (the perturbed equations). However, there are two key concepts we have to take in account before working on the equations.

#### The Scalar-Vector-Tensor Decomposition

The Scalar-Vector-Tensor Decomposition is a mathematical tool that allows any metric perturbation  $h_{\mu\nu}$  to be unambiguously split into three distinct types:

- **Scalar perturbations:** These describe density and pressure fluctuations and are responsible for the formation of LSS. They are characterized by scalar potentials such as  $\Phi$  and  $\Psi$ ;

- **Vector perturbations:** These correspond to rotational modes, but are not typically generated by inflation and decay with cosmic expansion, so they are typically ignored;
- **Tensor perturbations:** These correspond to gravitational waves, which are transversal and traceless and distort spacetime such that it is stretched and compressed in the plane perpendicular to their direction of propagation.

A crucial aspect of the metric perturbations is the decomposition theorem, stating that at linear order, these three types of perturbations evolve completely independently from one another. This allows us to study the evolution of structure (a scalar phenomenon) without worrying about contamination from gravitational waves (a tensor phenomenon).

### Gauge Choice

The values of the perturbation variables, like  $\Phi$  and  $\Psi$ , are not unique but depend on the choice of coordinates used to describe the perturbed spacetime. A common choice of gauge is the Conformal Newtonian Gauge, by which the perturbations to the metric can be expressed as:

$$\begin{cases} h_{00} = 2\Psi \\ h_{0i} = 0 \\ h_{ij} = 2\Phi\delta_{ij} \end{cases} \quad (2.16)$$

If we ignore the direction-dependent pressure caused by free-streaming relativistic particles (anisotropic stress), this simplifies to  $\Phi = -\Psi$ . This is a very good approximation for most of cosmic history.

Skipping the lengthy computations involved in calculating the perturbed Christoffel symbols and Ricci tensor, these previous two assumptions let us derive expressions for the perturbed Einstein equations. In particular, we find that the time-time component of the Einstein equations is (in Fourier space):

$$k^2\Phi + 3\frac{a'}{a}\left(\Phi' + \frac{a'}{a}\Psi\right) = 4\pi Ga^2 \sum_i \delta\rho_i; \quad (2.17)$$

this generalizes the Newtonian Poisson equation to a relativistic, expanding universe, and is entirely dependent on scalar perturbations, which is what this thesis concerns.

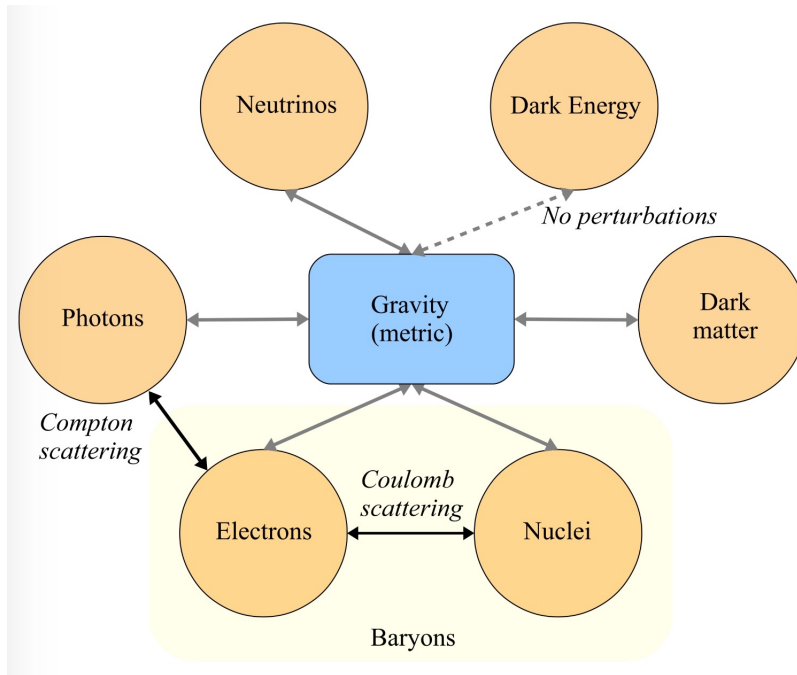
### 2.3.3 The Einstein-Boltzmann Coupled System

In order to solve the full system of linear perturbations, we need a full set of equations that tracks the evolution of each component. The complex network of interactions between these various cosmic components and the gravitational metric is visually summarized in Figure 2.2. The relevant equations for cosmological analysis are usually the monopole and dipole of the radiation distribution (on the first two lines), and the continuity and Euler equations for the cold dark matter (CDM) (on the two last lines):

$$\begin{cases} \Theta'_0 + k\Theta_1 = -\Phi' \\ \Theta'_1 + \frac{k}{3}\Theta_0 - \frac{k}{3}\Psi = -\tau'[\Theta_1 + \frac{1}{3}u_b] \\ \delta'_c + iku_c = -3\Phi' \\ u'_c + \frac{a'}{a}u_c + ik\Psi = 0 \end{cases}, \quad (2.18)$$

where  $u_c$  is the CDM velocity and  $\Theta$  denotes the photon temperature perturbation. The RHS of the radiation dipole equation (second line) is a collisional term that represents the interaction between photons and baryons;  $\tau'$  is the differential optical depth and  $u_b$  is the velocity of the baryon fluid.

In order to close this set of equations, we need an equation for the gravitational potential, for which Equation (2.17) is usually chosen.



**Figure 2.2.** A graph showing how the components in the Universe interact with each other, as well as gravity. All the connections are encoded in the coupled Einstein-Boltzmann equations. Electrons and atomic nuclei are treated as a single component, baryons, due to their tight coupling caused by Coulomb scattering.

Having closed the set of equation and defined the Einstein-Boltzmann system, we can study the evolution of all components by solving the set of five coupled equations numerically, for any given scale and time in the evolution of the Universe.

## 2.4 Modeling the Power Spectrum

Once the fundamental equations governing the evolution of cosmological perturbations (the Einstein–Boltzmann system) are established, the next step is to connect theory to observation by computing the matter power spectrum  $P(k)$ . In practice, the modern modeling of the power spectrum is generally written as:

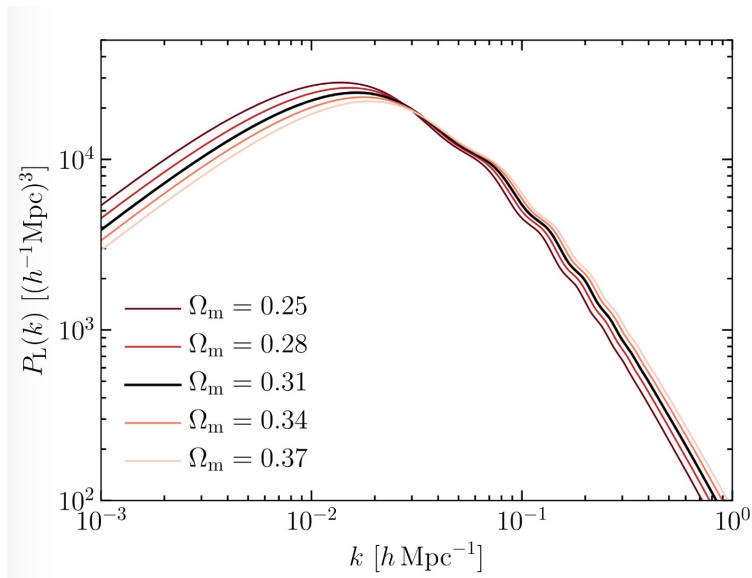
$$P(k, a) = A_s \left( \frac{k}{k_0} \right)^{n_s-1} T^2(k) D_+^2(a). \quad (2.19)$$

It requires:

- A background cosmological model (densities, curvature, cosmological constant);

- A primordial power spectrum parameterized by the scalar amplitude  $A_s$  and spectral index  $n_s$ , which sets the initial conditions;
- The transfer function  $T(k)$  and growth factor  $D_+(a)$ , which are the numerical solutions from the Einstein-Boltzmann system.  $T(k)$  encodes all the complex physics (like BAO wiggles and radiation-era suppression), while  $D_+(a)$  describes the simple, scale-independent growth at late times;
- And, if needed, nonlinear corrections from perturbation theory or  $N$ -body simulations (as described in Chapter 3).

The resulting theoretical matter power spectrum is shown in Figure 2.3, which demonstrates how altering a fundamental background parameter, such as the matter density  $\Omega_m$ , directly impacts the shape and amplitude of the spectrum.



**Figure 2.3.** The matter power spectrum at redshift  $z = 0$  in the fiducial  $\Lambda$ CDM cosmology, represented by the black line. The other lines show results when varying  $\Omega_m$  around the fiducial value: changing  $\Omega_m$  changes the epoch of equality, and consequently the shape of the matter power spectrum.

Numerical codes such as CAMB (Code for Anisotropies in the Microwave Background) and CLASS (Cosmic Linear Anisotropy Solving System) perform exactly this task. They evolve the Einstein–Boltzmann system in Fourier space for each mode  $k$ , tracking the evolution of metric and matter perturbations over time. The resulting outputs include the linear matter power spectrum  $P_L(k, a)$  and the predicted CMB temperature and polarization anisotropies, providing the final theoretical prediction that can be compared to observational data.

# 3

## Observational Probes of Large-Scale Structure

In the previous chapter, we had a quick overview of the theoretical models used to describe the properties of the LSS of the universe - that is, the way that matter in the Universe clumps into a vast network of filaments which is the cosmic web. Theory's value, however, is realized only when confronted with observations. Unfortunately, obtaining direct observational evidence on cosmological scales is notoriously difficult.

Our best probes are galaxy redshift surveys, which provide three-dimensional maps of the galaxy distribution. From these maps, we are able to estimate statistics such as the 2PCF and its Fourier transform, the power spectrum. While these observables are powerful tools in cosmology, interpreting them requires a deep understanding of how the observed galaxy positions in redshift space relate to their true positions in real space. Two main physical effects complicate this relationship: RSD and geometrical distortions induced by adopting an incorrect fiducial cosmology. Although these effects “distort” the observed galaxy distribution, they also encode valuable information about fundamental cosmological parameters and can therefore be exploited rather than simply corrected for.

Another valuable tool, although not strictly observational, is the use of cosmological simulations. These simulations act as a sort of virtual laboratory in which different models can be tested under controlled conditions. In this sense, simulations complement observations by providing a setting where theoretical predictions can be assessed and where the statistical properties of the LSS can be quantified with high precision.

In this chapter, I will introduce the main observational probes of LSS, the distortions that affect them, and the simulation tools that allow us to interpret these measurements and connect them back to cosmological theory. Most of the discussion follows the treatment in *Modern Cosmology* ([Dodelson & Schmidt, 2020](#)). Regarding simulations, a particular focus will be given to the Quijote simulation suite ([Villaescusa-Navarro et al., 2020](#)), which provides the numerical backbone for the analyses presented in this work.

### 3.1 Galaxy Clustering

Our best probes of the LSS are galaxy redshift surveys, which provide a direct measurement of how galaxies are distributed across the Universe at different redshifts. One way to study the Universe's matter distribution is to analyze how galaxies clump together, which is done by computing the power spectrum of the galaxy distribution. When we carry out such an analysis, we must take into consideration that the observed positions of galaxies (and thus their power spectrum) are subjected to distortions by:

- The difference between the fiducial cosmology and the parameters of the real Universe;
- RSD due to neglecting peculiar velocities when converting redshift into distances.

If a given galaxy is at a real comoving distance  $\chi(z)$  relative to us, its observed position in the sky will have components:

$$x_{obs}(z, \theta, \phi) = \chi(z) \hat{n}(\theta, \phi), \quad (3.1)$$

where the unit vector is  $\hat{n} = \vec{x}_{obs}/|\vec{x}_{obs}|$ . This presents two main complications. First, assuming a fiducial cosmology, the distance-redshift relation differs from the real one by a  $\delta\chi(z)$  amount:

$$\chi_{fid} = \chi(z) + \delta\chi(z). \quad (3.2)$$

Second, the observed redshift  $z$  is affected by the galaxy's peculiar velocity  $u_{||}$  (the component along the line of sight  $\hat{n}$ ):

$$1 + z = \frac{1}{a_{em}} [1 + u_{||}]$$

The change in the inferred distance at linear order is:

$$\Delta_{\text{RSD}} = \frac{\partial\chi}{\partial z} \Delta z \approx \frac{1}{aH} u_{||}. \quad (3.3)$$

Combining these two effects, we obtain the observed position  $\vec{x}_{obs}$  in terms of the true position  $\vec{x}$ :

$$\vec{x}_{obs} = \vec{x} + \left( \delta\chi(z) + \frac{1}{aH} u_{||}(\vec{x}) \right) \hat{n}, \quad (3.4)$$

where  $\vec{x}$  is the true three-dimensional position of the galaxy.

### 3.1.1 Galaxy Statistics

To relate the observed power spectrum to the true one, we use the fact that the number of galaxies is independent of the coordinates we use:

$$n_{g,\text{obs}}(\vec{x}_{\text{obs}})d^3x_{\text{obs}} = n_g(\vec{x})d^3x. \quad (3.5)$$

The infinitesimal volume elements are related by the Jacobian of the coordinate transformation,  $J = |d^3x/d^3x_{\text{obs}}|$ . In the “distant-observer” approximation (where  $\hat{n}$  is constant), this simplifies. The relation between the observed density perturbation,  $\delta_{g,\text{obs}}$ , and the true one,  $\delta_g$ , becomes:

$$\delta_{g,\text{obs}}(\vec{s}) \approx \delta_g(\vec{x}) - \frac{1}{aH} \frac{\partial u_{\parallel}}{\partial r_{\parallel}}, \quad (3.6)$$

where  $r_{\parallel}$  is the true line-of-sight distance. This equation is the foundation of the observed distortions.

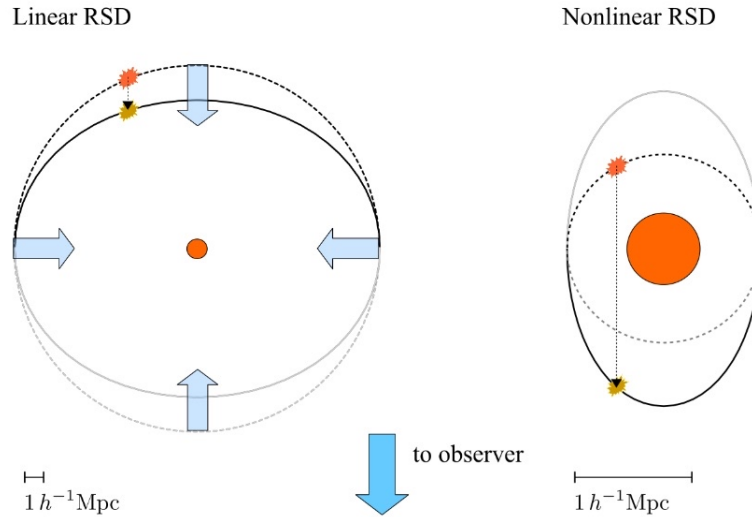
RSDs lead to an angular dependence ( $\mu_k$ ) in the power spectrum, which holds valuable information on the structure formation history. On the other hand, geometric distortions, also known as the Alcock-Paczynski distortions, caused by an incorrect distance-redshift relation (i.e.,  $\delta\chi \neq 0$ ), are useful to infer distance and the Hubble rate e.g. via the BAO standard ruler. We will take a closer look at these two types of distortions and how they are used to constrain cosmological parameters in the following paragraphs.

### 3.1.2 Redshift-Space Distortions

Peculiar velocities have an effect on galaxy clustering that differs between large-scale and small-scale structures:

- Large-scale (Kaiser effect): overdense regions appear “squashed” in redshift space, introducing an apparent anisotropy in an otherwise circular overdensity. This effect arises because galaxies are falling coherently toward the overdensity: those moving toward us along the line of sight appear closer in redshift space, while those moving away appear farther, leading to an apparent compression of structures along the line of sight, represented by the  $\partial u_{\parallel}/\partial r_{\parallel}$  term.
- Small-scale (Fingers of God): peculiar velocities displace galaxies more drastically than in the large-scale case, due to the fact that peculiar velocities are larger and the galaxies’ own motions ( $u_{\parallel}/aH$ ) become significant with respect to their

relative distance. This causes both an elongation in the overall overdensity and a sign inversion in the quadrupole moment of the clustering.



**Figure 3.1.** Illustration of the two primary effects of RSD, taken from (Dodelson & Schmidt, 2020). The left panel shows the Kaiser effect (squashing) due to coherent bulk flows on large scales. The right panel shows the Fingers of God effect (stretching) due to random motions on small, clustered scales. The dashed gray line represents the true spatial distribution of an overdensity, while the solid black line represents the observed distribution in redshift space.

The two effects are shown in Figure 3.1.

To compute the power spectrum, we relate the velocity  $u_{\parallel}$  to the density field  $\delta_m$  using the linear continuity equation, which gives  $u_{\parallel}(\vec{k}) \propto f/k\mu_k\delta_m(\vec{k})$ . Substituting this into the Fourier transform of the density relation yields the famous Kaiser formula:

$$\delta_{g,\text{RSD}}(\mathbf{k}) \propto [b_1 + f\mu_k^2] \delta_m(\mathbf{k}), \quad (3.7)$$

where  $b_1$  is the linear bias factor relating galaxy clustering to matter clustering,  $f$  is the linear growth rate of structure ( $f = d \ln D_+ / d \ln a$ ), and  $\mu_k$  is the cosine of the angle between the wavevector  $\mathbf{k}$  and the line of sight. The dependence on this last parameter means that the observed power spectrum becomes anisotropic, depending not only on the magnitude of  $k$  but also on its direction.

By measuring the magnitude and anisotropy of the power spectrum, RSDs allow for the determination of the crucial cosmological parameter  $f\sigma_8$  (the growth rate multiplied by the matter fluctuation amplitude), which gives us information on structure formation processes.

### 3.1.3 The Alcock-Paczynski Effect

If the assumed fiducial cosmology is incorrect, the galaxy distribution is distorted, leading to an apparent anisotropy in the galaxy power spectrum. This geometrical effect is known as the Alcock-Paczynski (AP) effect.

The distortion in the 3D galaxy field can be characterized by two different parameters evaluated at the mean redshift  $\bar{z}$  of the galaxy survey slice:

- Transverse distortion ( $\alpha_{\perp}$ ): proportional to the ratio of the true and fiducial comoving distances ( $\chi(\bar{z})/\chi_{\text{fid}}(\bar{z})$ );
- Line-of-sight distortion ( $\alpha_{\parallel}$ ): proportional to the ratio of the true and fiducial Hubble parameters ( $H_{\text{fid}}/H(\bar{z})$ ).

By measuring these distortion parameters,  $\alpha_{\perp}$  and  $\alpha_{\parallel}$ , observers can infer the true distance ( $\chi(\bar{z})$  or the angular diameter distance  $D_A(\bar{z})$ ) and the true Hubble rate  $H(\bar{z})$  at that redshift.

### 3.1.4 Baryon Acoustic Oscillations as a Standard Ruler

One of the most efficient tools to exploit the AP effect is by using the BAO as a “standard ruler”.

The BAO is a small, characteristic oscillatory modulation imprinted in the matter power spectrum by ripples in the early baryon-photon plasma. This modulation corresponds to a specific, comoving scale  $r_s$ , the sound horizon at recombination. Since the value of  $r_s$  is very precisely determined by CMB data, measuring the apparent size of this feature in galaxy surveys allows for a precise measurement of distance and expansion history.

The parameters  $\alpha_{\perp}$  and  $\alpha_{\parallel}$  shift the location of the BAO feature in the observed power spectrum, enabling highly precise constraints on  $D_A(\bar{z})/r_s$  and  $H(\bar{z})r_s$ . The full equation describing the observed galaxy power spectrum, incorporating both RSD and AP distortions, is:

$$P_{g,\text{obs}}(k_{\text{obs}}, \mu_{\text{obs}}) = \frac{[b_1 + f\mu_k^2]^2}{\alpha_{\perp}^2 \alpha_{\parallel}} P_m(k) + P_N, \quad (3.8)$$

where  $k$  and  $\mu_k$  on the right side are the true values, which are functions of the observed  $k_{\text{obs}}$ ,  $\mu_{\text{obs}}$ , and the distortion parameters  $\alpha_{\perp}$  and  $\alpha_{\parallel}$ .  $P_N$  is the shot noise, an error term that accounts for the discreteness of galaxies as tracers of the matter density field.

## 3.2 Cosmological Parameter Estimation

Having established how observational effects distort the galaxy power spectrum, we now turn to the question of how cosmological parameters can be extracted from the data. The basic idea is to compare a theoretical model for the observed power spectrum,  $P_{g,obs}(k, \mu)$ , with measurements from a galaxy survey, and determine the set of parameters that best reproduces the data. The parameters of interest typically include:

- The bias  $b_1$ ;
- The growth rate  $f\sigma_8$ ;
- The AP parameters  $\alpha_{\parallel}$  and  $\alpha_{\perp}$ ;
- Additional nuisance parameters (e.g., the shot noise).

To infer these quantities, we construct a statistical model describing the probability of observing a given data set by assuming a specific choice of parameters. This probability is encoded in the likelihood function, a fundamental component of Bayesian analysis, which forms the core of parameter estimation in cosmology.

### 3.2.1 Likelihood Function

To extract cosmological information from the observed galaxy power spectrum, we compare the data to theoretical predictions through a likelihood function. Let  $\hat{P}$  denote the measured power spectrum multipoles, and let  $P_{th}(\theta)$  be the theoretical model for a set of parameters  $\theta$ . In the simplest approach, we assume that the measured multipoles are Gaussian-distributed around their theoretical expectations. The likelihood for  $\theta$  is then:

$$\mathcal{L}(\theta) \propto \exp \left[ -\frac{1}{2} \left( \hat{P} - P_{th}(\theta) \right)^T \mathbf{C}^{-1} \left( \hat{P} - P_{th}(\theta) \right) \right], \quad (3.9)$$

where  $\mathbf{C}$  is the covariance matrix encoding the uncertainties and correlations in the observations. In this work, we will not discuss the detailed construction of  $\mathbf{C}$ , but treat it as a known input.

### 3.2.2 Bayesian Analysis

The standard approach for estimating cosmological parameters is Bayesian inference, which is built upon Bayes' theorem. This model does not require us to assume a

hypothesis to be true a priori, but lets us update our convictions about model parameters, gradually, as more and more data becomes available. For a set of parameters  $\theta$  and a data set  $D$ , the theorem states:

$$P(\theta|D) = \frac{P(D|\theta) \cdot P(\theta)}{P(D)}, \quad (3.10)$$

where:

- $P(\theta|D)$  is the posterior probability of the parameters  $\theta$  to be correct given the observed data  $D$ . This is our updated conviction on the accuracy of the hypothesized parameters, which is the quantity we aim to calculate;
- $P(D|\theta)$  is the likelihood, defined as the probability of observing the data  $D$  if the parameters  $\theta$  were accurate. This term quantifies how compatible data is with the hypothesis;
- $P(\theta)$  is the prior, containing any previous knowledge we have on the parameters before observing the data  $D$ . This could derive from previous experiments, established theories or even our ignorance of the parameters;
- The evidence,  $P(D)$ , is the probability of observing the data  $D$  independently from any specific hypothesis. This serves as a normalization factor and is crucial for comparing between different models.

Our goal is to map the posterior probability  $P(\theta|D)$  to find the most likely parameter values and their uncertainties.

Maximizing this likelihood (or, equivalently, the posterior when the prior is significant) gives the best-fit parameters, while sampling the posterior with Monte-Carlo methods, often used when the likelihood is non-Gaussian, yields the distributions used for parameter inference.

### 3.2.3 Markov Chain Monte Carlo

The true posterior distribution  $P(\theta|D)$  is often non-Gaussian in the parameters (exhibiting, for example, “banana-shaped” degeneracies) and is high-dimensional. Brute-force calculation by mapping this space on a grid is computationally impossible in typical cosmological analyses.

It is therefore convenient to use a sampling algorithm, typically Markov Chain Monte Carlo (MCMC), which performs a random walk through the high-dimensional parameter space in this fashion:

- It starts at a random point in the parameter space  $\theta_i$ ;
- Proposes a "step" to a new point  $\theta_{i+1}$ ;
- Calculates the posterior  $\mathcal{P}$  at this new point;
- Decides whether to accept or reject the step based on the ratio  $\mathcal{P}(\theta_{i+1})/\mathcal{P}(\theta_i)$ . It always accepts a step to a higher-likelihood region and sometimes accepts a step to a lower-likelihood one.

By repeating this process several times, the algorithm preferentially spends its time in regions of high probability. After an initial “burn-in” phase, during which the chain stabilizes and approaches the target posterior distribution, the chain of accepted samples  $\{\theta_i\}$  forms a fair statistical representation of the true posterior distribution.

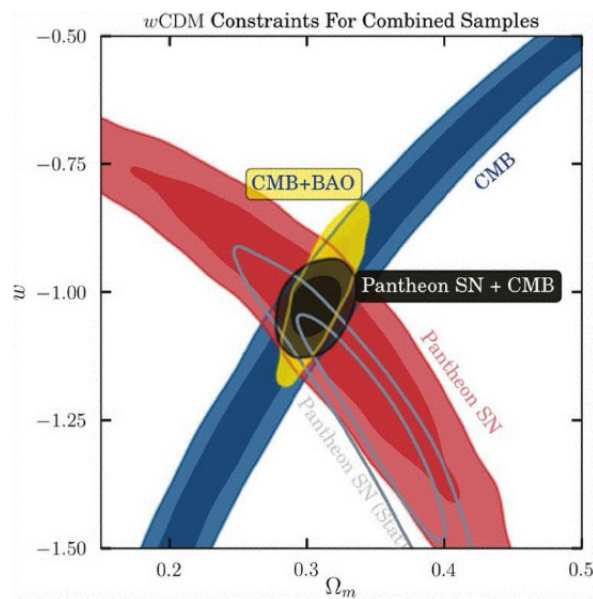
What we expect to obtain from such analysis are corner plots such as that in Figure 3.2, which show the probability distribution function (PDF) of the inferred parameters. Corner plots are useful for visualizing high-dimensional data by showing all pairwise relationships between variables in a grid, with histograms on the diagonal cells, representing the 1D distribution of each parameter, and scatter plots in the off-diagonal cells showing the 2D correlations between parameter pairs. In these off-diagonal panels, the samples naturally reproduce the familiar curved, “banana-shaped” confidence regions that are typical of cosmological parameter degeneracies.

### 3.3 Cosmological Simulations

Cosmological simulations act as a virtual laboratory for the study of LSS. Unlike observations, which provide only one realization of the Universe, simulations allow us to explore how the matter distribution evolves under different physical and cosmological assumptions in a fully controlled setting, allowing us to bridge the gap between fundamental theory and complex observables measured by galaxy surveys.

Cosmological simulations are essentially N-body simulations that evolve a set of dark matter particles under gravity from early times to the present day, providing a detailed map of the matter distribution in the Universe. The core of these simulations is a gravitational solver, which computes the forces between particles efficiently while capturing the growth of structure on a wide range of scales.

Codes such as GADGET ([Springel, 2005](#)), use hybrid methods that combine large-scale efficiency with high-resolution small-scale accuracy, making them ideal for studying both the linear and non-linear regimes of structure formation.



**Figure 3.2.** Example of a corner plot illustrating parameter constraints on the dark energy equation of state ( $w$ ) and matter density ( $\Omega_m$ ) for a flat  $w$ CDM cosmology. The colored contours represent the confidence regions derived from independent observational probes: Type Ia supernovae (red), the CMB (blue), and BAO (yellow). The joint constraints (black) demonstrate how orthogonal data sets effectively break parameter degeneracies. Adapted from (Scolnic et al., 2018).

### 3.3.1 The Quijote Simulation Suite

The cosmological analyses presented in the following chapters use data sets from the Quijote simulations (Villaescusa-Navarro et al., 2020), one of the most extensive N-body simulation suites designed for cosmological parameter studies. They evolve dark matter particles under gravity from the early Universe to today, effectively serving as a virtual laboratory to study LSS formation. The suite comprises 44 100 independent simulations, including:

- 15 000 realizations at the fiducial cosmology;
- Over 7000 simulations exploring variations in several cosmological parameters;
- Box sizes of  $1 \text{ Gpc} h^{-1}$  with resolutions ranging from  $512^3$  to  $2048^3$  particles;
- Outputs including particle snapshots, halo catalogs, and summary statistics such as power spectra, bispectra, PDF, and void catalogs.

The large cosmic volume of the Quijote simulations ensures that BAO scales are well resolved and that RSD can be accurately studied. For this thesis, these simulations provide a high-fidelity numerical representation of the LSS, allowing direct comparisons between theoretical predictions and observations that can be used for galaxy statistics and parameter inference.

# 4

## Introduction to Large Language Models

AI has intrigued humanity for decades, from ancient myths like Talos the bronze automaton to modern computing. The field of AI formally began in the 1950s, with the Dartmouth Summer Research Project on AI in 1956 being widely cited as the founding event of the field. Today, AI spans many approaches, but a dominant trend is the use of machine learning (ML), and in particular neural networks (NNs), to learn from data.

In astrophysics and cosmology, the necessity of ML has been driven by the increasing size and complexity of data sets generated by surveys like (e.g.) the Dark Energy Survey ([The Dark Energy Survey Collaboration, 2018](#)) and, in the near future, the Euclid surveys ([Mellier et al., 2025](#)). The sheer volume of data demands automated, efficient, and sophisticated methods for analysis that go beyond traditional techniques. This is a case where ML techniques, particularly Deep Learning (DL), can offer a solution. DL has already been successfully applied to critical tasks such as building fast emulators for cosmological simulations. In this work, however, we will investigate a slightly different angle, focusing on the potential use of LLMs in enhancing cosmological analysis.

The LLM revolution was initiated by the advent of the Transformer architecture ([Vaswani et al., 2023](#)), which ushered in the era of models that power conversational agents such as ChatGPT<sup>1</sup>, Gemini<sup>2</sup>, or Claude<sup>3</sup>. While initially focused on natural language processing, the underlying principles of LLMs are highly relevant to scientific workflows, particularly for synthesizing complex information and generating code.

In this chapter, we provide a conceptual overview of how these models function. We begin by defining the fundamental building blocks of DL and then we briefly introduce the specific architecture of the Transformer that allows computers to “speak” and reason autonomously.

---

<sup>1</sup><https://chatgpt.com>

<sup>2</sup><https://gemini.google.com>

<sup>3</sup><https://claude.ai>

## 4.1 Machine Learning and Deep Learning Fundamentals

Although this work does not involve the training of new LLMs, a foundational understanding of their architecture is necessary to appreciate their reasoning capabilities and their potential applications to cosmological analysis. Accordingly, the following section is dedicated to introducing a theoretical framework of NNs and the Transformer architecture that underlie modern LLMs.

### 4.1.1 What Is Machine Learning?

ML represents a fundamental shift from traditional programming. In the classical paradigm, a human operator provides both the data and the explicit, hard-coded rules for processing the data, with the computer’s role being simply to execute those rules deterministically. In ML, this relationship is reframed: the human provides the computer with the input data and the desired output, with the goal that the algorithm itself finds the rules that map the former to the latter.

To put it simply, an ML algorithm is one that “is able to learn from data” (Mitchell, 1997), a process that relies on three main components:

- **The Task ( $T$ ):** The objective, such as predicting the next word in a sentence, classifying a picture or winning a game of chess;
- **The Experience ( $E$ ):** The data set used for training. This works as a sort of “textbook” that the algorithm learns from in order to carry out a given task precisely;
- **The Performance Measure ( $P$ ):** A metric that quantifies how well the model approximates the desired function.

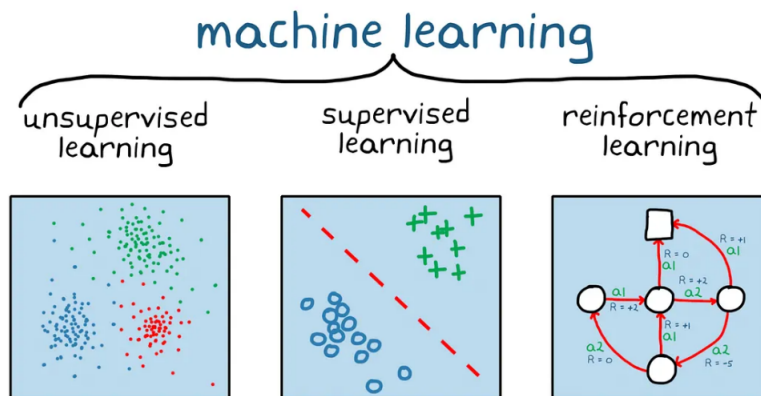
According to Mitchell (Mitchell, 1997): “A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .”

### 4.1.2 Machine Learning Paradigms

We have said that  $E$  is the bank of data from which the model learns information. However, the specific ways in which the model learns from  $E$  are different, with some algorithms being better suited for some tasks than others.

The learning algorithms are broadly categorized into three macro-categories (see Figure 4.1):

- **Supervised Learning:** The experience consists of labeled examples. This means that to each input,  $x$ , corresponds a target  $y$ , and the algorithm is supposed to learn the mapping function  $f$  that correctly approximates  $y = f(x)$ . This supervision provides a direct feedback signal: each input should produce one correct output. This learning algorithm couples well with tasks like classification, or fitting a function to data;
- **Unsupervised Learning:** The data set is not labeled. This means that the algorithm is only given the inputs,  $x$ , with the goal of finding some  $p(x)$  that describes interesting correlations or structure within the data. This is often used for tasks such as dimensionality reduction (such as Principal Component Analysis) or grouping similar data points (Clustering);
- **Reinforcement Learning (RL):** A paradigm where an agent learns to make optimal decisions by interacting with an environment and receiving feedback in the form of rewards or penalties.



**Figure 4.1.** A schematic overview of the three main ML paradigms. Left (Supervised): The model learns to draw a decision boundary to separate data points into distinct categories based on known labels. Center (Unsupervised): The algorithm detects inherent structure in unlabeled data, grouping points into clusters based on their proximity. Right (Reinforcement): An agent optimizes its behavior by interacting with an environment, learning through a feedback loop of actions, states, and rewards. Image taken from [The MathWorks Inc. \(2024\)](#).

LLMs are trained using a combination of these learning strategies. The most critical paradigm for understanding LLMs is called Self-Supervised Learning (SSL).

In this framework, the model starts with unlabeled data and generates its own supervision labels from the data itself, effectively transforming the unsupervised data

corpus into a massive supervised learning task. Moreover, RL is sometimes used in later stages of a model’s training, for example through Reinforcement Learning through Human Feedback (RLHF), in which a human operator manually gives feedback to the model to adjust the model’s internal parameters.

At this stage, it is worth noting that LLMs do not process raw text directly. Instead, they operate on processing units called “tokens”—small words, sub-words, or characters—that are represented as numerical vectors, making them readable to a computer.

A significant problem with this task and learning strategy is that the amount of data needed to carry this out is massive and difficult to manage without the use of NNs.

### 4.1.3 Deep Learning and Artificial Neural Networks

The systems capable of handling the enormous scale and complexity required for modern LLM algorithms are particular ML structures based on DL. DL uses NNs, mathematical functions loosely inspired by the structure of the human brain. They are comprised of very simple processing units, called neurons, organized into multiple layers; the term “deep” refers to networks having multiple hidden layers between the input and the output layers that allow these algorithms to process increasingly abstract information.

Each single neuron has to work on a large set of internal numerical values that determine how the model transforms an input into an output. Essentially, each neuron  $j$  in the network takes in an input vector  $\mathbf{x}$  and produces a linear combination as an output:

$$z_j = \mathbf{w}_j \cdot \mathbf{x} = \sum_i w_{ji}x_i + b_j, \quad (4.1)$$

where  $w_{ji}$  and  $b_j$  are the internal parameters for that neuron, specifically the weights and the bias terms. The weights  $w$  determine the strength of connection between neurons, resembling synaptic connections in biological brains, while the bias  $b$  provides an adjustable offset, effectively increasing or decreasing the probability of the neuron activating independently of the input signal. The collection of all weights and biases constitutes the state, or the “knowledge” of the network, often denoted generally as  $\theta$ .

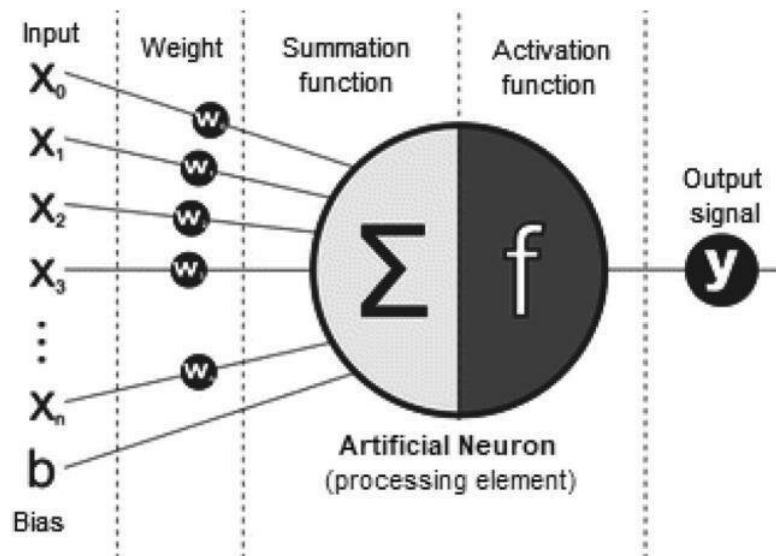
The linear combination  $z_j$  is always passed through a nonlinear activation function:

$$a_j = f(z_j), \quad (4.2)$$

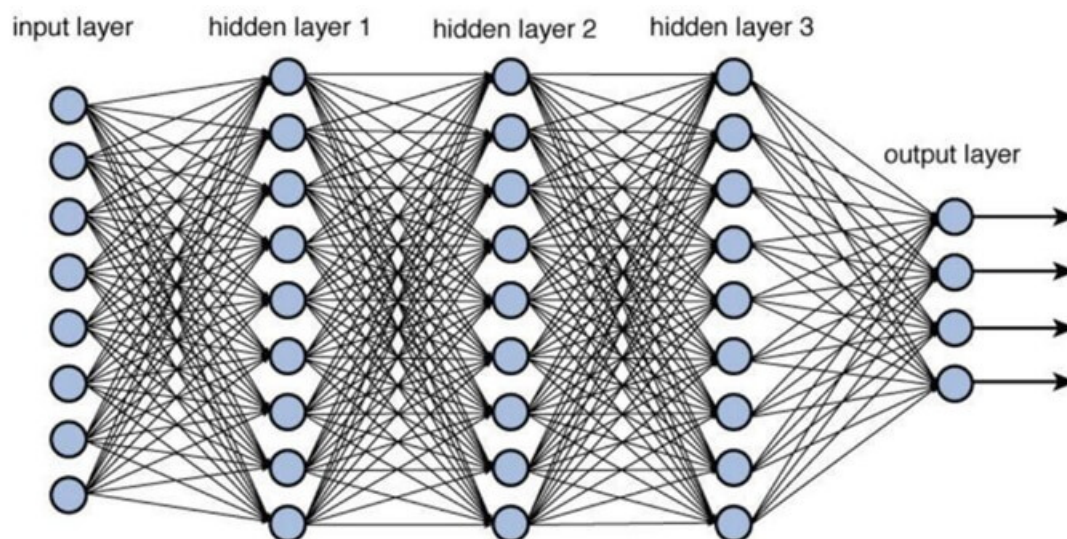
where  $a_j$  represents the final output of a given neuron.

The result of this activation function is determining which next neurons will activate

in the network before walking to the final output. This is analogous to only a selected number of neurons activating in the human brain when dealing with a specific task. In the context of ML it is necessary because, if NNs consisted only of linear operations, a network of any depth would be mathematically equivalent to a single linear layer (since the composition of linear functions is just another linear function), and it would not be possible for the network to distinguish and learn any complex information. A visual representation of a single neuron and of a Deep NN architecture are given respectively in Figures 4.2 and 4.3.



**Figure 4.2.** Schematic representation of a single artificial neuron, taken from [Emerging Information & Technology Conference](#). The inputs  $x_i$  are multiplied by their respective weights  $w_i$ , summed together with a bias term  $b$ , and passed through a non-linear activation function  $f$  to produce the output  $y$ .



**Figure 4.3.** Figure from [Ramachandra \(2018\)](#) showing the architecture of a Deep NN. Information flows from the input layer on the left, through multiple hidden layers where abstract features are extracted, to the final output layer on the right.

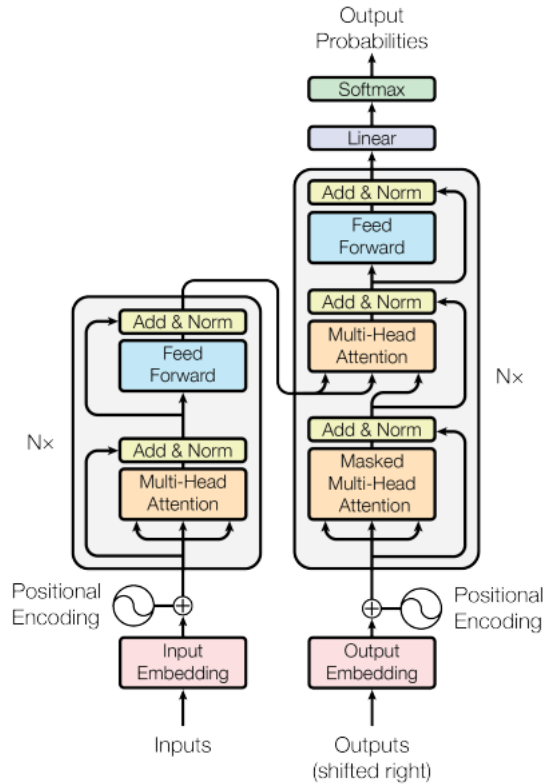
## 4.2 The Transformer and Self-Attention

The computational engine that makes modern LLMs possible is the Transformer architecture, introduced by [Vaswani et al. \(2023\)](#). Before this innovation, language models processed text sequentially (word by word), much like a human reading a sentence. This approach had two major flaws: it was computationally slow, and the models were forgetful, often losing the context of a sentence after just a few words.

The Transformer, shown in Fig. 4.4, revolutionized this by introducing the process of Self-Attention, by which every token in a sentence is able to look at every other token to calculate their reciprocal relevance. This is done by computing a correlation matrix that stores numerical scores quantifying how strongly related any two tokens are, regardless of their distance in the text. For example, in the sentence “The star collapsed because **it** was too massive”, the model must resolve the ambiguity of the “it” pronoun. It does this by assigning a high attention score between “it” and “star”, the noun it is referring to. Because the attention scores are calculated simultaneously for entire sequences, rather than one by one, this architecture allows for massive parallelization on modern hardware, which makes it possible to train on very vast data sets such as the entire internet.

Before processing, each token is converted into a numerical vector (Embedding) including its positional information, allowing the model to distinguish word order

despite parallel processing. After passing through the Transformer layers, including the Self-Attention layers, the final vector is projected back into the vocabulary space (Unembedding) to calculate the probability distribution for the next token.



**Figure 4.4.** The originally proposed Transformer architecture (Vaswani et al., 2023), showing an Encoder on the left and a Decoder on the right. The data flow is hierarchical: input text is converted into numerical vectors (Embeddings), passed through the Self-Attention mechanism to gather context, and processed by a standard NN layer as a final step. Note that modern LLMs utilize a Decoder-only architecture, effectively using only the right-hand stack of this diagram.

The original Transformer architecture is composed of two distinct blocks: an Encoder, which processes and interprets the input sequence, and a Decoder, which generates the output sequence based on that interpretation. Modern Generative AI models (such as GPT) utilize a slightly modified, Decoder-only version of the original Transformer architecture. This means they are structurally constrained to look only at past tokens to predict the future, rather than analyzing the full sentence simultaneously as an Encoder would, which makes them able to generate text.

### 4.2.1 Training of Large Language Models

As mentioned previously, LLMs are trained through SSL, where the data given as input to the model provides its own supervision signals. The model treats the actual next tokens in a sentence as the ground truth, creating a feedback loop where predictions are instantly validated against the original text.

The training process can be conceptualized as a massive, high-speed “guessing game.” The model is presented with a sequence of text derived from internet data (books, articles, code repositories), but the final part of the sequence is hidden. The model’s sole objective is to predict the immediate next token based on the context provided so far. For example, given the input “The expansion of the Universe is”, the model calculates the probability distribution over its entire vocabulary to determine the most likely next word (e.g., “accelerating”).

During the training phase, the model reads trillions of words, playing the “guessing game” over and over. If it guesses wrong, it slightly nudges its internal weights via the optimization process to make the correct answer more likely the next time. By repeating this process trillions of times, the model is able to compress the statistical structure of language and abstract information into its own weights.

Essentially, the model’s capabilities rest on a sophisticated ability to mix and match linguistic components, reconstructing new sequences based on the patterns it memorized during training.

## 4.3 Agentic Artificial Intelligence and LangGraph

The standard interaction between a human operator and an LLM is a prompt-and-response cycle: the user asks questions to the LLM, and the model generates a probability-based answer. While powerful, this mechanism has limitations when it comes to doing rigorous scientific work:

- **Hallucinations:** LLMs always produce a plausible answer, but it might be filled with incorrect facts or faulty, broken code;
- **No iteration cycles:** A standard LLM is unable to check its own work or run the code by itself to verify its functioning;
- **Static nature:** The model is unable to access external tools or libraries directly.

To move beyond these limitations, Agentic AI frameworks, also called AI agents, were introduced.

Algorithmically speaking, an AI agent is a software tool built around repeated, sequential calls to an LLM. Rather than relying on a single prompt and a single response, the agent operates in a continuous loop where, at each step, the software queries the LLM to determine the next appropriate action, such as generating a snippet of Python code, reading a file, or calling a specific library.

Because the LLM is highly skilled at recognizing patterns and parsing context, the agent can evaluate which computational tools to use, reflect on its own errors and operate with a certain degree of autonomy. Essentially, it could be visualized as a chatbot programmed to act both as the human user and the AI assistant simultaneously, dialoguing with itself to arrive at a desired solution.

We implement this type of architecture using LangGraph (Wang & Duan, 2024), a Python library designed to build multi-agent applications by representing the agent’s logic as graphs, consisting of the following main components:

- **State:** A shared data structure that represents the internal “memory” of the agent. It persists across all steps of the agent’s computations;
- **Nodes:** Python functions that represent a unit of work;
- **Edges:** Connections between nodes that define the logical flow. Conditional Edges are particularly important, as they work like the `if` operator and can route the agent’s internal state to different nodes based on the outcome of the previous step.

With the theoretical framework of LLMs and the structural logic of Agentic AI established, we can now proceed to the practical implementation. The pipeline developed for this work, demonstrating how these abstract concepts are translated into a functional tool for automated cosmological analysis, will be detailed in the following chapter.



# 5

## Cosmological Analysis with Agentic Artificial Intelligence

The introduction of LLMs into scientific workflows represents a potential paradigm shift in how computational physics is conducted, enhancing the computer’s autonomy in conducting rigorous, end-to-end analyses.

The primary objective of this thesis work is to build an autonomous agentic system, evaluating its capacity to derive meaningful physics insights from cosmological data with as minimal human intervention as possible. This was implemented entirely within a Python environment, leveraging Jupyter Notebook for execution and the LangGraph framework for state management.

This chapter details the experimental design, describing the graph-based architecture of the agents, the prompt engineering required to enforce scientific accuracy, and the validation mechanisms used to ensure the AI’s results are sound.

### 5.1 Experimental Design

This thesis employs a comparative analysis methodology. A traditional, human-driven cosmological analysis was first performed to establish a ground truth benchmark. An autonomous AI agent was then tasked with performing the exact same analysis from first principles, receiving only the raw data and a natural language prompt. The results from both methods were then quantitatively and qualitatively compared to assess the reliability, accuracy, and scientific understanding of the AI.

The analysis was divided into two distinct, sequential tasks:

- **Task 1:** 2PCF monopole computation;
- **Task 2:** Parameter inference via MCMC.

### 5.2 Baseline Analysis

This section details the human-driven analysis performed on a snapshot from the Quijote simulations (Villaescusa-Navarro et al., 2020) at redshift  $z = 0$ . It serves as the reference

standard for evaluating the AI.

### 5.2.1 Computation of the Two-Point Correlation Function

The ground truth 2PCF was computed using the CosmoBolognaLib (CBL) framework (Marulli et al., 2016), a highly optimized C++/Python set of libraries.

- **Data Preparation:** A dark matter halo data catalog was loaded and a corresponding random catalog was generated using the CBL framework.
- **Data Export:** The halo Cartesian coordinates ( $X, Y, Z$ ) were exported to plain-text files (`catalog.dat` and `random.dat`). These raw text files served as the absolute sole input provided to the AI agent, ensuring it had no access to any pre-computed CBL objects, methods, or internal helper functions.
- **2PCF Computation:** Using the internal CBL objects, the `TwoPointCorrelation_multipolesDirect` tool was used to compute the benchmark 2PCF monopole. The measurement was configured for 30 linear bins from 1 to 150 Mpc  $h^{-1}$ . The resulting monopole, with the associated Poisson errors, was saved to a file, which represents the correct answer for the first task.

After the benchmark solution was established, the agents were set up to independently carry out a similar analysis. The workflow is as follows:

- An elaborate prompt was devised to pass the instructions to the agents as unambiguously as possible. The specific phrasing was developed iteratively by observing the agent’s initial outputs and continuously steering its behavior toward the correct scientific baseline;
- The prompt was given to the agents, who devised a strategy to carry out the task independently;
- The resulting code and its outputs were compared to the benchmark results to evaluate accuracy.

### 5.2.2 Parameter Inference via Markov Chain Monte Carlo

The second, more tentative task of the analysis consists of a parameter inference run. While the monopole  $\xi_0(r)$  integrates over angular dependencies and cannot fully constrain dynamic and geometric distortion parameters, it serves as a sufficient test to evaluate the statistical reasoning capabilities of the agent. We followed this procedure:

- **Theoretical Model:** To model the BAO into the 2PCF we fit the measured monopole with a theoretical template based on linear theory. Specifically, the analytical model is defined as:

$$\xi_{model}(r) = (b\sigma_8)^2 \xi_{fid}(\alpha r) + A_0 + \frac{A_1}{r} + \frac{A_2}{r^2}. \quad (5.1)$$

Here,  $\alpha$  is the geometric scaling parameter that captures shifts in the BAO peak relative to the fiducial cosmology. The amplitude of the clustering signal is parametrized by  $b\sigma_8$ , which jointly captures the linear galaxy bias ( $b$ ) and the matter fluctuation amplitude ( $\sigma_8$ ).  $A_0$ ,  $A_1$ , and  $A_2$  are broadband nuisance parameters designed to marginalize over nonlinear structure formation and theoretical inaccuracies, effectively isolating the pure BAO geometric signal.

- **Likelihood Analysis:** A standard Gaussian likelihood was employed. The data vector consisted of the monopole computed in the previous step, while the covariance matrix was assumed to be diagonal, derived directly from the Poisson errors of the pair counts.
- **Sampling:** The parameter space was explored using uniform priors for all model parameters ( $\alpha, b\sigma_8, A_0, A_1, A_2$ ). These priors were identical to those later provided to the AI agent to ensure a fair comparison. The sampler was initialized with 1000 walkers. A burn-in phase was performed to stabilize the chains, followed by a production run of 10 000 steps.

Again, a benchmark solution was computed through the CBL, and the agents were then set up to carry out a similar task independently.

### 5.3 The Code

The proposed solution is a hierarchical multi-agent system that automates the computation of the 2PCF and parameter inference. We use the LangGraph framework (Wang & Duan, 2024), which allows us to model the agent not as a linear chain of commands, but as a graph of state transitions.

The architectural pattern chosen for this work is the Reflexion agent. As described in Shinn et al. (2023), it is a framework that uses verbal reinforcement to induce self-improvement. Rather than simply retrying a failed task blindly, the agent generates a verbal critique of why it failed, stores this in memory, and uses it to guide the subsequent attempt. This mimics the human debugging process, where an error message leads to a hypothesis about the bug, which then informs the fix.

In particular, this project’s design is heavily inspired by the AlphaCodium application of the Reflexion agent (Ridnik et al., 2024). This approach encourages coding practices such as bullet point prompts and modular code generation through sub-functions in order to facilitate debugging.

## 5.4 Application Programming Interface for Large Language Models

The agents developed in this work rely on proprietary LLMs, in particular GPT-4o (Achiam et al., 2023) and Claude 3.5 Sonnet (Anthropic, 2024), which are not executed locally due to their immense computational requirements. Instead, they are accessed remotely via an Application Programming Interface (API).

Consequently, reproducing this analysis requires an active API key from the respective provider (e.g., OpenAI or Anthropic). It is important to note that these are commercial services; users are typically charged a fee based on the number of tokens processed by the model during the generation and reflection cycles.

To manage these credentials, the pipeline utilizes the `os` and `getpass` Python modules. This setup allows the user to input their paid API keys securely at runtime as environment variables.

It should be noted that since the framework developed in this thesis was specifically evaluated using Claude 3.5 Sonnet and GPT-4o, comparable accuracy and convergence stability cannot be guaranteed for other models. While free, open-weights models, such as Llama 3 (Dubey et al., 2024) or Mistral (Jiang et al., 2023) exist and can be run locally, they are generally reported as not as powerful and might fail to converge on the correct solution without extensive fine-tuning.

## 5.5 The Graph Architecture

The architecture consists of two distinct layers:

- **Subgraphs (Agents):** Two independent graphs, one for the 2PCF computation and another for parameter inference (MCMC).
- **Parent Graph (Orchestrator):** A higher-level graph that manages the workflow and passes data (such as the computed 2PCF) from one agent to the next.

By reducing the complexity of the task into smaller, modular subgraphs, we effectively constrain the search space for the LLM and optimize the reasoning trajectory of the

agents. This prevents the model from becoming overwhelmed by a too broad context (which increases the risk of hallucinations) and allows for granular control and validation at each intermediate step of the cosmological analysis.

## 5.6 The Graphs

### 5.6.1 Structured Output Schema

One of the primary challenges in using LLMs for code generation is ensuring that the output can be parsed reliably by a computer. To address this, we use the Pydantic library to enforce a strict output schema (Colvin et al., 2025), forcing the agents to return a clearly structured object containing three specific fields, as shown in Listing 5.1.

**Listing 5.1.** Pydantic schema for Agent Output

---

```
class CodeOutput(BaseModel):
    prefix: str = Field(description="Description of the problem and approach")
    imports: str = Field(description="Library import statements")
    code: str = Field(description="The executable Python code block")
```

---

The forced structure helps the agents compartmentalize the task. The `prefix` field encourages Chain-of-Thought reasoning, prompting the model to first describe its logical plan before execution. This ensures a deeper understanding of constraints before generation begins. The separation of `imports` and `code` simplifies the execution process in the Python sandbox, allowing us to parse dependencies and install missing libraries dynamically before the main execution block runs.

### 5.6.2 The Graph State

The core of the functioning of the graph lies in the `GraphState`, a typed dictionary that acts as the short-term memory of the agent. It carries information across the processes of generation, execution, and validation (see Listing 5.2).

**Listing 5.2.** Definition of the GraphState

---

```
class GraphState(TypedDict):
    error: str # Flag for execution errors
    messages: List # Chat history (User/AI/System)
    generation: str # The generated Python code
    iterations: int # Counter for self-correction loops
```

---

---

```

last_step_approved: bool    # Flag for human feedback
xi: List[float]           # The computed 2PCF result
validated: bool           # Flag for numerical validation

```

---

The most important aspect of this design is that it allows the graph to be cyclic; if an error occurs, the `error` field is updated, and the graph routes back to the generation node with the error context preserved in the `messages` list. This allows the agent to remember its previous mistake and attempt a different solution in the next iteration.

### 5.6.3 The Nodes

Once the memory state is defined, the code defines all the sub-functions that are used by the agent to carry out the specific task it is designed to undertake. These correspond to the single nodes in the agent's workflow.

#### The Generation Node

The `generate` function acts as the primary code generator. It operates by constructing a prompt based on the current state and invoking the LLM. The logic within this node is recursive; if a code check fails in a later step, the graph routes back to this node. Upon re-entry, the function increases the iteration index by one and is invoked again to retry the generation. This loop continues up to a designed maximum number of iterations. The output of this node is a structured code object, which is appended to the messages history, as demonstrated in the generation logic in Listing 5.3.

**Listing 5.3.** The functioning of the generation node.

---

```

if error == "yes":
    messages += [
        (
            "user",
            "Now, try again. Invoke the code tool to structure the output
with a prefix, imports, and code block:",
        )
    ]

    # Solution
    code_solution = code_gen_chain.invoke(
        {"messages": messages}
    )

```

```

# Enforcing Structured Output (Prefix vs Code)
messages += [
    (
        "assistant",
        f"{code_solution.prefix} \n Imports: {code_solution.imports} \n
Code: {code_solution.code}".rstrip(),
    )
]

# Increment
iterations = iterations + 1
return {"generation": code_solution, "messages": messages, "iterations":
iterations}

```

The initial conditional block (`if error == "yes"`) is bypassed during the initial generation phase, as no error state exists yet. It can only be triggered by a subsequent iteration after a failed execution check, ensuring the agent explicitly addresses the faults of the previously generated code before producing a new one.

### The Execution Node

The code check function (implemented as `check_twop` and `check_chains` in the code) is responsible for executing the generated script in a controlled environment. It verifies whether both the imports and the code run smoothly without runtime errors. The outcome of this node determines the subsequent flow of the graph, with three possible scenarios:

- If the execution is successful, the graph transitions to the human feedback node, where the user can review the results (see Listing 5.4).

**Listing 5.4.** Code execution success; the graph's internal parameters are updated to proceed with the next step.

```

print("---NO CODE TEST FAILURES---")

return {
    "generation": code_solution,
    "messages": messages,
    "iterations": iterations,
    "error": "no",
}

```

- 
- If the execution is unsuccessful but the iteration index has not yet reached the maximum limit, the graph transitions to the reflection node. This triggers a self-correction cycle where the agent attempts to fix its own errors, as shown in Listing 5.5.

**Listing 5.5.** Code execution failure; a message is appended to the list and used for the next iteration.

---

```

try:
    exec(imports + "\n" + code, {})
except Exception as e:
    print("---CODE BLOCK CHECK: FAILED---")
    error_message = [("user", f"Your solution failed the code
execution test: {e}")]
    messages += error_message
    return {
        "generation": code_solution,
        "messages": messages,
        "iterations": iterations,
        "error": "yes",
    }

```

---

- If the execution fails and the maximum number of iterations has been reached, the graph halts the loop. In this case, the broken code is returned along with the error message, which is then presented to the user for manual inspection in the human feedback stage. This function updates the `error` variable in the graph state, which acts as the control signal for the decision node.

For the 2PCF computation, there is an additional step that validates the results against a previously computed CBL benchmark, which can be activated through the `flag_validate` variable. If this validation is not successful, it appends a message in the message list and sends it back to generation for a retry (Listing 5.6).

**Listing 5.6.** The 2PCF validation process.

---

```

if flag_validate == "validate":

    print("Running validation...")

```

```
same = np.allclose(
    computed_xi,
    benchmark_xi,
    rtol=0.5,
    atol=0.5
)

if same:
    print("Arrays match! Stopping early.")

    return {
        "generation": code_solution,
        "messages": messages,
        "iterations": iterations,
        "error": "no",
        "xi": computed_xi,
        "validated": True
    }

else:
    print("Validation failed. Continuing execution...")
    messages.append(
        ("user", f"Validation failed: xi does not match the
benchmark. Write a better code in order to reach the \
benchmark: \n{benchmark_xi}\n .")
    )
```

---

## The Reflection Node

The `reflect` node implements the self-correction mechanism inspired by the Reflexion architecture. If the previous generation has failed, this node is activated to analyze the error. It involves an LLM call, taking the standard error output from the execution node and prompting the LLM to verbally diagnose the cause of the failure, as demonstrated in Listing 5.7.

**Listing 5.7.** The reflection process.

---

```
reflections = code_gen_chain.invoke(
    {"messages": messages})
```

```
messages += [("assistant", f"Here are reflections on the error:
{reflections}")]
return {
    "generation": code_solution,
    "messages": messages,
    "iterations": iterations
}
```

---

For example, the model might identify a shape mismatch in a numpy array or a memory overflow. It then formulates a correction plan, which is added to the `messages` history. This node always routes the workflow back to the generation node, ensuring that the new insight is immediately applied to the next attempt.

### The Decision Node

The `decide_to_finish` function acts as a logic gate that manages the control flow of the graph based primarily on the state of the `error` variable. It checks if the code executed successfully, if the maximum retries have been exhausted and if the produced results are sound. If either condition is met, it routes the graph to the next step in the workflow. If neither is met (meaning an error occurred but retries are still available), it redirects the flow back to the generation or reflection nodes to continue the loop (see Listing 5.8).

**Listing 5.8.** The functioning of the decision node.

---

```
if iterations >= max_iterations:
    print(f"---DECISION: MAX ITERATIONS REACHED ({iterations}) ->
FINISH---")
    return "vision_chains"

if error == "no":
    print("---DECISION: SUCCESS (Validated) -> FINISH---")
    return "vision_chains"

print(f"---DECISION: FAILURE (Error: {error}) -> RETRYING---")

if flag == "reflect":
    return "reflect"
else:
    return "generate"
```

---

## The Vision Node

To further enhance the validation capabilities, a vision node was integrated into the pipeline. This node loads the local image file generated by the code (e.g., the plot of the 2PCF) and creates a `HumanMessage` containing the image data. It does so by converting the binary image into a Base64 encoded string and embedded into a structured message object for a model with vision capabilities, such as GPT-4o, as shown in Listing 5.9. Ideally, this returns an exhaustive physical analysis of the cosmological graph, parsing every detail and giving suggestions on what can be done to improve the analysis.

**Listing 5.9.** The functioning of the vision node.

---

```
image_data = image_to_base64(image_path)

vision_message = HumanMessage(
    content=[
        {"type": "text", "text": "You are a cosmologist. I will show you
a graph containing cosmological data. \
    You will describe everything you see; what they are, what they
imply from a physical \
    perspective, if they are in accordance with each other, how they
can be improved. Every detail you can parse."},
        {
            "type": "image_url",
            "image_url": {"url": f"data:image/png;base64, {image_data}"}
        }
    ]
)

print("Analyzing results...")

llm_oai.invoke([vision_message]).pretty_print()
```

---

This mimics the role of a senior cosmologist reviewing a plot, allowing the agent to “see” if the results are physically meaningful.

## The Human Feedback Node

This node serves as the final validation gate and is designed to give the user control over the scientific validity of the output. It activates when the reflection loop is deactivated or when the maximum number of iterations has been extinguished. At this stage, the

user is presented with the final result of the generation process, including the generated plot and console output.

The user interacts with a chatbot interface to provide feedback. This interaction has two possible outcomes:

- If the user presses ENTER, the input is interpreted as an approval of the code, and the graph moves to its END state;
- If the user provides any text input, it is interpreted as a refusal. In this case, the graph resets the iteration counter to zero and routes back to the generation node. The user's text input is stored in the `messages` variable, injecting new instructions or constraints into the agent's memory for the next generation attempt, as implemented in Listing 5.10.

This function also automatically resets the iteration count to zero, meaning it will re-run the entire generation process if the human operator decides to refute the result.

---

**Listing 5.10.** The functioning of the human feedback node.

---

```
iterations = 0

user_input = input("Your command: ").strip()

# Add user message if any
if user_input != "":
    messages.append(("user", f"Here are the user's considerations:
{user_input}"))

# Return updated state
return {
    "last_step_approved": user_input == "",
    "messages": messages,
    "iterations" : iterations,
}
```

---

The human feedback node also includes another subfunction, `should_continue`, responsible for evaluating the approval state to determine whether to conclude the workflow or trigger a complete reiteration (see Listing 5.11).

---

**Listing 5.11.** Implementation of the graph's loop termination condition.

---

```
if state.get('last_step_approved', False):
```

```
        return "end" # approved
    else:
        return "retry" # rejected
```

---

This adds a final security measure, allowing for human guidance in the case that the agents are misinterpreting their task or producing incorrect results.

#### 5.6.4 Compiling the Graphs

The following snippet (Listing 5.12) shows the compilation of the 2PCF subgraph. By registering the individual sub-functions as nodes and explicitly defining the conditional edges, it establishes the precise execution logic, routing, and cyclical reflection pathways of the agentic workflow.

**Listing 5.12.** Compilation of the 2PCF subgraph.

---

```
workflow.add_node("generate", generate)
workflow.add_node("check_twop", check_twop)
workflow.add_node("reflect", reflect)
workflow.add_node("vision", vision)
workflow.add_node("human_feedback", human_feedback)

# Build graph
workflow.add_edge(START, "generate")
workflow.add_edge("generate", "check_twop")

workflow.add_conditional_edges(
    "check_twop",
    decide_to_finish,
    {
        "vision": "vision",
        "reflect" : "reflect",
        "generate" : "generate",
    },
)

workflow.add_edge("reflect", "generate")
workflow.add_edge("vision", "human_feedback")

workflow.add_conditional_edges(
    "human_feedback",
```

```

        should_continue,
        {
            "end": END,                # If 'approve', end the graph
            "retry": "generate" ,      # If 'reject', go back to the start
        })

twop_app = workflow.compile()

```

---

Following an identical methodology, a parallel subgraph was compiled to handle the MCMC parameter inference.

With both dedicated agents fully constructed, they were subsequently embedded as distinct operational nodes within a higher-level parent graph. This orchestrator manages the entire pipeline, ensuring the workflow successfully transitions from the 2PCF computation directly into the parameter inference stage, as shown in Listing 5.13.

---

**Listing 5.13.** Compilation of the parent graph orchestrator.

---

```

parent_graph = StateGraph(ParentState)

parent_graph.add_node("twop_computation", twop_computation)
parent_graph.add_node("chain_computation", chain_computation)

parent_graph.add_edge(START, "twop_computation")
parent_graph.add_edge("twop_computation", "chain_computation")
parent_graph.add_edge("chain_computation", END)

parent_app = parent_graph.compile()

```

---

## 5.7 Prompt Engineering

The most important aspect when interacting with LLMs is prompt engineering, which refers to the strategic formulation of text inputs required to effectively translate a complex human objective into a computational task the model can execute.

A critical component of this engineering is the enforcement of structured output formats. By constraining the model's response to a specific schema, we significantly reduce the probability of hallucination and ensure that the generated code can be reliably executed and provide sensible results. In this work, the final prompts were constructed using a multi-layered approach, combining static System Prompts to establish each agent's persona and role in the workflow, and dynamic Human Message prompts that

can be used to assign the agents specific jobs with specific instructions.

### 5.7.1 The Two-Point Correlation Function System Prompt

The prompt used for the first agent (see Listing 5.14) instructs the model to use the Landy-Szalay estimator (Landy & Szalay, 1993) and handle edge cases like normalization.

It is important to note that this final configuration was reached after a significant process of trial and error. We started from a much simpler set of instructions, but early iterations frequently led to physical and technical errors: common issues included incorrect normalization factors, double-counting of pairs, skipping counts to make the computation faster (but too approximated to be scientifically reliable), and frequent array length mismatches when handling data. Consequently, the prompt had to be refined iteratively, adding specific constraints to address these issues one by one, until the instructions were unambiguous enough to guarantee a stable execution each time.

**Listing 5.14.** 2PCF Prompt

```
"""You are tasked with calculating the 2-point correlation function (2PCF)
    monopole on a dark matter halo catalog and a relative random catalog.

The user will give you these parameters as input:
1) catalog: filename of the catalog, containing the comoving coordinates of
    the particles (X, Y, Z) in Mpc/h units;
2) random_catalog: the associated random catalog;
3) minimum_separation: minimum distance between galaxies across which the
    2PCF is computed;
4) maximum_separation: maximum distance between galaxies across which the
    2PCF is computed;
5) nbins: the number of radius bins used in the computation;
6) shift: the shift with respect to the bin center;

Follow these steps:

1. Load Data
   - Load the catalog and random catalog from the filenames stored in
     "catalog" and "random_catalog".
   - Keep in mind that they're in comoving (X, Y, Z) coordinates, in Mpc/h
     units, within the fiducial LambdaCDM cosmology.

2. Compute the 2PCF function monopole
   - Use the two catalogs to compute an estimate for the correlation function.
```

- It could be any of the multipoles depending on what is requested by the user.
- Use the Landy-Szalay estimator.
- Keep in mind that the random catalog is larger than the catalog by some amount (check it).
- DO NOT SKIP ANY COUNT!!!
- Ensure all pair counts are normalized correctly and that no spurious signal arises due to double-counting, self-pairs, or incorrect normalization. Check that the mean of the largest separations is near zero. If it is offset (e.g. near one), re-normalize xi.
- Print a message when you're done measuring each of the counts.

### 3. Compute Poissonian errors

- Compute Poissonian errors (shot noise) on the correlation function. They should be computed as:  $(1 + \xi)/\text{np.sqrt}(DD)$ , where DD is the (NOT NORMALIZED) number of data-data pairs in a given bin.

### 4. Plot the Results

- Plot the 2PCF monopole with r in Mpc/h units on the x-axis (bin centers) and xi (r) on the y-axis.
- It should be a lineless scatter plot with the associated errors (use `plt.errorbar()`). Show the graph when you're done!

Save the plot as 'graph.png' in the working directory.

All input vectors x and y can be any length.

Do not assume a fixed length. Always handle arrays dynamically based on the actual length of the input. Don't use Greek characters in order to avoid unicodeescape errors. Explain what you're doing step by step.

After computing the 2PCF, you MUST store the list/array of xi(r) values in the 'xi' field of your structured output. Make sure the 'xi' field contains a plain Python list of floats.

ALSO, use the time module to compute the runtime of the 2PCF computation process."""

---

Complementing the task-specific human prompts, a foundational system prompt was deployed to enforce global instructions. This system prompt establishes the agent's baseline persona and operational constraints, ensuring rigid adherence to structured

outputs and safe coding practices regardless of the specific mathematical task at hand.

### 5.7.2 The Markov Chain Monte Carlo Parameter Inference Prompt

For the second task, the agent was provided with the computed correlation function and asked to infer the cosmological parameters. Again, this specific prompt was selected through process of trial-and-error, where simpler prompts were given the agents at first, and later modified depending on the agents' misconceptions on the task, resulting in the final prompt shown in Listing 5.15.

**Listing 5.15.** MCMC Inference prompt

---

```

"""You are tasked with performing cosmological parameter inference using a
    2-point
correlation function monopole, provided as a file. You will calculate
    posteriors
on several parameters by running an MCMC analysis and comparing the data set
    to a
theoretical model prediction.

The user will provide the following inputs:
1) twop_file: filename of the 2PCF monopole. One column contains the
    separation
    in Mpc/h, the other contains the values of xi.
2) covariance_matrix: the associated covariance matrix.
3) parameter_list: the set of parameters to infer, with uniform prior ranges.
    These parameters are: alpha, b_sigma8, A0, A1, A2.
4) xmin: the minimum scale included in the fit.
5) xmax: the maximum scale included in the fit.
6) chain_size: the total number of samples in the MCMC chain.
7) nwalkers: the number of walkers to use in the MCMC.

Follow these steps:

1. Load the data:
    - Read the 2PCF monopole file and store the separation values and xi values
      in arrays, applying the xmin and xmax cuts.

2. Construct the model:
    - You need a 2PCF model computed at each step of the chain.
    - The theoretical model for the 2PCF monopole is defined as:
      xi_model(r) = (b_sigma8)**2 * xi_template(alpha * r) + A0 + A1/r +

```

```
A2/r**2,
  where xi_template should be computed with a proper Einstein-Boltzmann
  solver.
```

3. Set up the Bayesian analysis:
  - Assume a Gaussian likelihood.
  - Use uniform priors for all parameters: alpha, b\_sigma8, A0, A1, A2. The prior limits are provided by the user.
4. Compute the posteriors:
  - Run an MCMC using the chain\_size and nwalkers values provided by the user.
  - Return the full posterior distribution for all parameters.
5. Print the results:
  - Print the posterior means and credible intervals for all inferred parameters.

Always produce a single, self-contained script where all functions, imports, and variables are defined in the same scope, in the correct order, and without relying on any external or prior definitions. Do not split definitions across cells or assume they persist. Provide the full script as one coherent block.

Do not use Greek characters anywhere in the code to avoid unicode escape issues.  
 Explain all steps clearly as you generate the code."""

---

This part of the code requires as input the monopole computed in the previous step, and a covariance matrix that was computed with the CBL, in this case.

### 5.7.3 The Human Message Prompt

The final component of the prompting strategy is the dynamic user message. This prompt serves as the execution trigger, providing the agent with the exact parameters of the physical problem, the relevant data set file paths, and any targeted algorithmic instructions required for that specific run.

In the context of this thesis, the human messages were meticulously crafted to guide the agent away from known computational pitfalls. By embedding explicit “hints”

derived from prior failed iterations, we successfully steered the model toward more stable and scientifically valid solutions. The final invocation payload, which passes these critical constraints to both agents and produced the results analyzed in Chapter 6, is presented in Listing 5.16. It involves explicit instructions to use the Eisenstein-Hu model (Eisenstein & Hu, 1998) through the `colossus` library (Diemer, 2018) for computing the theoretical power spectrum, and then `mcfit` (Li, 2019) to integrate the power spectrum and obtain the 2PCF template. These libraries, initially proposed by the agents during prior iterations, were explicitly embedded into the final prompts to ensure a highly stable and reproducible computational workflow.

**Listing 5.16.** The final invocation payload. The “HINT FROM PREVIOUS ITERATIONS” tags denote explicit human constraints added to prevent computational errors observed in earlier test runs.

---

```

result = parent_app.invoke({
    "twop_message": [
        ("user", """Hi. You are tasked with doing 2PCF monopole computation
on a catalog of dark matter halos.

I will give you the parameters needed for such computation:
- catalog: 'halos_0.dat'
- random_catalog: 'halos_0_random.dat'
- minimum_separation: 1
- maximum_separation: 150
- nbins: 30
- shift: 0.5

HINT FROM PREVIOUS ITERATIONS: Use Corrfunc for blazing fast computation.
HINT FROM PREVIOUS ITERATIONS: Set periodic=False!

Good luck!""")
    ],
    "chain_message": [
        ("user", """Hi. You are tasked with doing parameter inference using
the MCMC method.

These are the initial data and priors for the parameters:
- twop_file = "monopole.dat"
- covariance_matrix = 'covariance_square.txt'
- priors: alpha = [0.8, 1.2], b_sigma8 = [1, 5], A0 = [-100, 100],
A1 = [-100, 100], A2 = [-1000, 1000]

```

```
- xmin = 1.0, xmax = 150.0
- chain_size = 10000, nwalkers = 100, burn in: 20% of chain_size
```

HINT FROM PREVIOUS ITERATIONS: Do NOT use a power law approximation for the model. You should use an appropriate Einstein-Boltzmann solver or an approximation of it.

HINT FROM PREVIOUS ITERATIONS: I recommend you use the colossus library with the EisensteinHu model, which should be fast and light enough to not cause kernel crashes. Assume a standard value for sigma\_8 for the fiducial Planck18 cosmology at redshift 0.

HINT FROM PREVIOUS ITERATIONS: You should import cosmology from colossus.cosmology, define a cosmo model, and then use that object to compute the power spectrum through cosmo.matterPowerSpectrum(). Pre-compute this template globally outside the MCMC loop so the sampler stays fast. Assume a Planck18 cosmology.

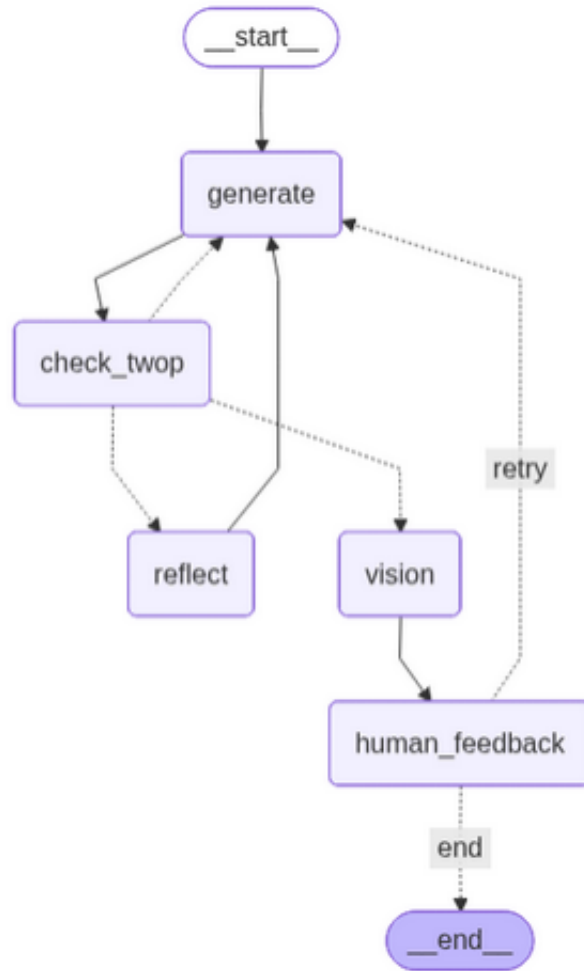
HINT FROM PREVIOUS ITERATIONS: CRITICAL PERFORMANCE CONSTRAINT: You MUST pre-compute the theoretical correlation function template globally, OUTSIDE and BEFORE the likelihood function and the MCMC loop. Inside the log\_likelihood function, you should ONLY perform the algebraic scaling (applying alpha, b\_sigma8, and the polynomials) to this pre-computed template. Do NOT call the cosmological model or FFTLog inside the MCMC steps, or the process will take too long.

HINT FROM PREVIOUS ITERATIONS: Use mcfit (FFTLog) to carry out the Hankel transform integration, in log space.

Please, explain each step of your solution in detail and show me the corner plots when you're done.

```
Give me the executable code now!""")
    ]
})
```

---



**Figure 5.1.** The state graph of the AI agent, visualized using the built-in drawing tools of LangGraph.

It is important to highlight that while the agent was granted autonomy in devising the coding implementation for the MCMC sampling, it was constrained to operate within a specific theoretical framework. The agent was forced to adhere to a previously computed template and a definitive set of priors, as preliminary testing demonstrated that allowing the model complete freedom in defining the parameter space often led to chaotic behavior and failure to converge.

With the agentic machinery now fully defined, the following chapter presents a final evaluation of this system, where we effectively compare the AI-derived 2PCF and MCMC posteriors against the human-driven baseline based on CBL, evaluating the framework’s computational efficiency and its ability to recover the ground truth cosmology of the Quijote simulations.



# 6

## Results

The implementation of the multi-agent pipeline described in the previous chapters yielded a functional system capable of devising strategies to perform cosmological analysis and carry it out autonomously. However, the path to convergence was not linear. While the system demonstrated remarkable adaptability, the results highlight significant blind spots relative to the agents' capabilities when left entirely to their own devices.

This final chapter analyzes the performance of the multi-agent system, focusing on three critical dimensions: the computational efficiency of the produced code, the sensitivity of the LLM to prompt engineering, and the agents' capacity for self-correction.

Regarding the latter, while the agents possess significant coding capabilities when powered by a model such as Claude-3.5 or GPT-4, they still require substantial guidance by the human operator to produce scientifically rigorous results. We propose that while the agents act as a powerful engine for code generation and data processing, the human operator must function as the steering wheel, refining the scientific direction and correcting model hallucinations.

### 6.1 Evaluating the Agents' Performance

The following sections provide a detailed analysis of two distinct pipeline outputs: the calculation of the 2PCF and the subsequent cosmological parameter inference. It should be noted that due to the stochastic nature of generative models, specific code implementations and intermediate outputs may vary across different execution runs, even when initialized with identical instructions. The results presented here illustrate a typical performance outcome of the system.

#### 6.1.1 The 2PCF Case Study

To evaluate the agents' coding efficiency, we tasked the system with computing the 2PCF on a reduced data set of 10 000 halos extracted from a Quijote simulation snapshot<sup>1</sup>, containing a total of 406 660 halos, at redshift  $z = 0$ . The underlying cosmological framework is a flat  $\Lambda$ CDM model, with parameters extracted directly from the simulation

---

<sup>1</sup><https://quijote-simulations.readthedocs.io/en/latest/>

header:  $\Omega_m = 0.3175$ ,  $\Omega_\Lambda = 0.6825$ , and a dimensionless Hubble parameter  $h = 0.6711$ . The spatial catalog represents a cubic volume of  $1000 \text{ Mpc } h^{-1}$  on a side, yielding an effective number density of  $10^{-5} (h/\text{Mpc})^3$ . For the 2PCF computation, the pair counts were evaluated across 30 linear bins from a minimum separation of  $1 \text{ Mpc } h^{-1}$  up to a maximum separation of  $150 \text{ Mpc } h^{-1}$ .

As for the baseline analysis, this was carried out through the CBL framework, utilizing an implementation analogous to the example codes provided in CBL's documentation<sup>2</sup>.

All computations for this performance test were executed within CBL's Ubuntu-based virtual environment<sup>3</sup>, configured in VirtualBox with 2 vCPUs and 11 GB of allocated RAM. Under these conditions, the baseline CBL computation completed in 5.0 seconds.

To test the agents' self-correcting capabilities, the initial prompt omitted the explicit constraints (the "HINTS") discussed in the previous chapter (see Section 5.2.1). The agents' initial response, as shown in Listing 6.1, was to generate a standard Python script using nested `for` loops to calculate pair counts. The logic was mathematically sound, utilizing the `scipy.spatial.cKDTree` library (Virtanen et al., 2020) for neighbor queries. However, they defaulted to a textbook definition of the problem, iterating through points pair-by-pair rather than utilizing more efficient standard numerical optimization techniques.

**Listing 6.1.** Initial naive 2PCF implementation generated by the agents using nested loops.

```
import numpy as np
from scipy.spatial import cKDTree

# ... [Setup code omitted] ...

# Step 4: Compute DD (data-data pairs)
print("Computing DD pairs...")
DD_counts = np.zeros(nbins)

for i in range(len(data)):
    # Query neighbors for each point individually
    indices = tree_data.query_ball_point(data[i], rmax)

    for j in indices:
        if j > i: # Avoid double counting
            dist = np.linalg.norm(data[i] - data[j])
```

<sup>2</sup>[https://colab.research.google.com/drive/1tpQ1Tj06RRwOfZtxYXPC1Sr\\_Hdb679w8](https://colab.research.google.com/drive/1tpQ1Tj06RRwOfZtxYXPC1Sr_Hdb679w8)

<sup>3</sup><https://federicomarulli.github.io/CosmoBolognaLib/Doc/html/index.html>

```

if dist >= rmin and dist < rmax:
    bin_idx = int((dist - rmin) / (rmax - rmin) * nbins)
    DD_counts[bin_idx] += 1

```

While the resulting 2PCF matched the CBL benchmark, the computational efficiency was poor, with a runtime of 21.07 seconds, against CBL's 5.0 seconds.

Utilizing the Human-in-the-Loop interface, we provided feedback to the agents regarding the execution time. The agents correctly identified the bottleneck as nested loops and generated another code that employed vectorization via the NumPy library (Harris et al., 2020). Instead of querying neighbors individually, the agents utilized `tree.query_pairs` to retrieve all pairs simultaneously, as seen in Listing 6.2.

**Listing 6.2.** Vectorized implementation. The agents replaced the nested loops with numpy broadcasting and efficient tree queries.

```

# Optimized Step 4: Compute DD using Vectorization
print("Computing DD pairs...")

# 1. Get all pairs within rmax at once (returns set of tuples)
# This avoids the Python loop overhead
pairs = tree_data.query_pairs(rmax, output_type='ndarray')

# 2. Vectorized distance calculation
# Extract coordinates for all pairs simultaneously
p1 = data[pairs[:, 0]]
p2 = data[pairs[:, 1]]
diff = p1 - p2
distances = np.linalg.norm(diff, axis=1)

# 3. Binning using Histogram (replaces manual binning loop)
hist, _ = np.histogram(distances, bins=bin_edges)
DD_counts += hist

```

This refinement reduced the runtime to 1.51 seconds, a 14× speedup over the code in Listing 6.1. This highlights the agents' ability to self-correct and improve their code when guided by the user.

To further test the agents' capabilities, we requested a final iteration asking if the execution could be sped up further. Rather than performing low-level optimization, the agents demonstrated their ability to efficiently search for and integrate existing high-performance codes. They autonomously selected the `Corrfunc` library (Sinha & Garrison, 2020), a highly optimized set of OpenMP-parallelized C routines for clustering statistics. Given this new prompt, the agents successfully generated the integration

code shown in Listing 6.3.

**Listing 6.3.** Final optimized code using the C-based Corrfunc library.

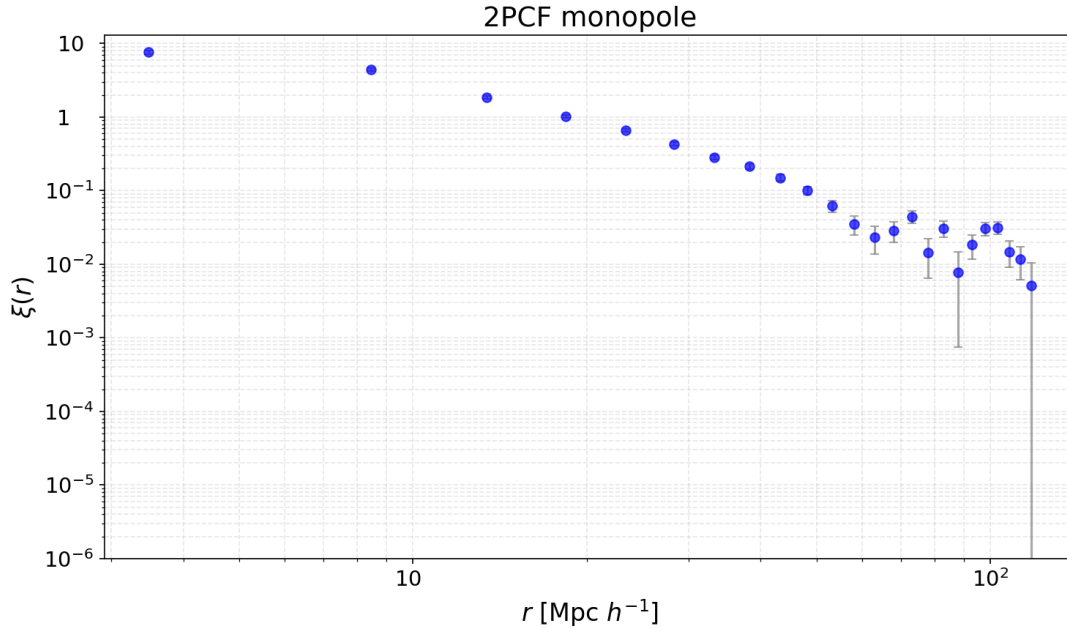
```
from Corrfunc.theory.DD import DD

# ... [Setup code omitted] ...

# Use Corrfunc for high-performance counting
# nthreads=4, autocorr=1 (true)
print("Computing DD pairs with Corrfunc...")
results_DD = DD(1, 4, bin_edges, X_data, Y_data, Z_data,
                periodic=False, verbose=False)

# Extract counts directly from the C-struct
DD_counts = np.array(results_DD['npairs'], dtype=float)
```

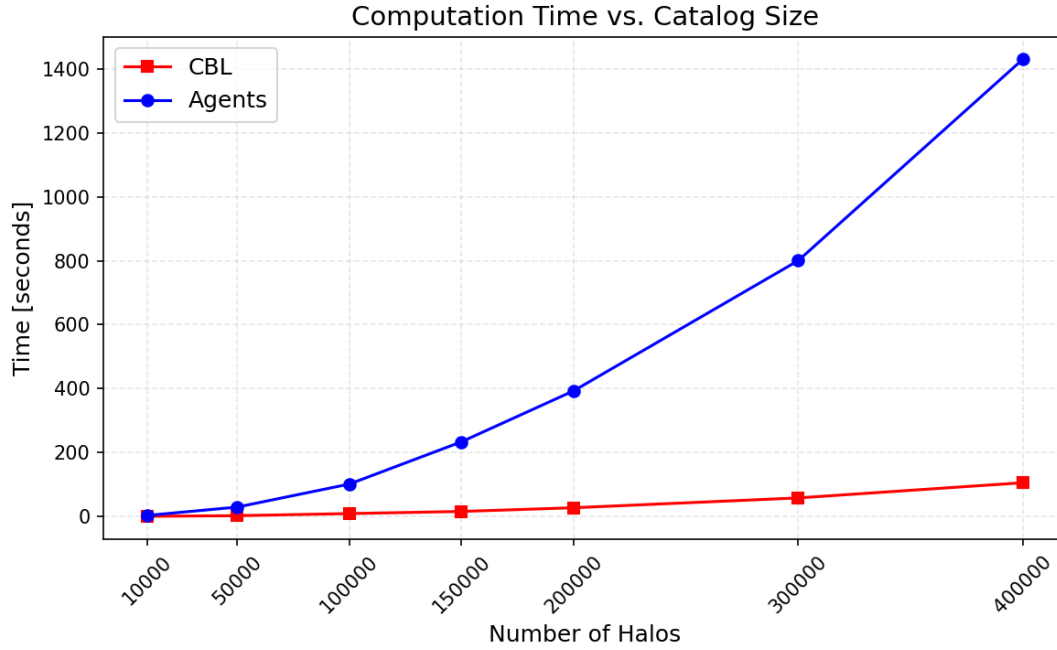
The transition to Corrfunc yielded a result perfectly in line with the CBL benchmark but with a runtime of only 0.12 seconds, representing a  $\sim 175\times$  speedup over the code in Listing 6.1 for a catalog of 10 000 halos. The resulting 2PCF is shown in Figure 6.1.



**Figure 6.1.** The 2PCF monopole computed by the autonomous agents on a catalog of 10 000 halos. The error bars represent Poissonian uncertainties, calculated by the agents as  $(1 + \xi)/\sqrt{DD}$ .

This represents a large improvement over the initial naive attempt. It shows the agents' ability to not only interpret the physical problem correctly but also to actively seek out and use high-performance computational tools when prompted by the user.

To demonstrate the scalability and performance of the agent-generated solution, a scaling analysis was performed by increasing the number of halos  $N$ . The sub-samples were obtained through direct random partitioning of the original catalog, effectively reducing the number density while maintaining the full simulation volume; while this naive partitioning is unsuitable for a true clustering analysis (as it disrupts the LSS), it still served as a robust scaling test. As shown in Table 6.1 and Figure 6.2, the agent's final solution maintains efficient scaling, significantly outperforming the CBL baseline for larger halo counts.

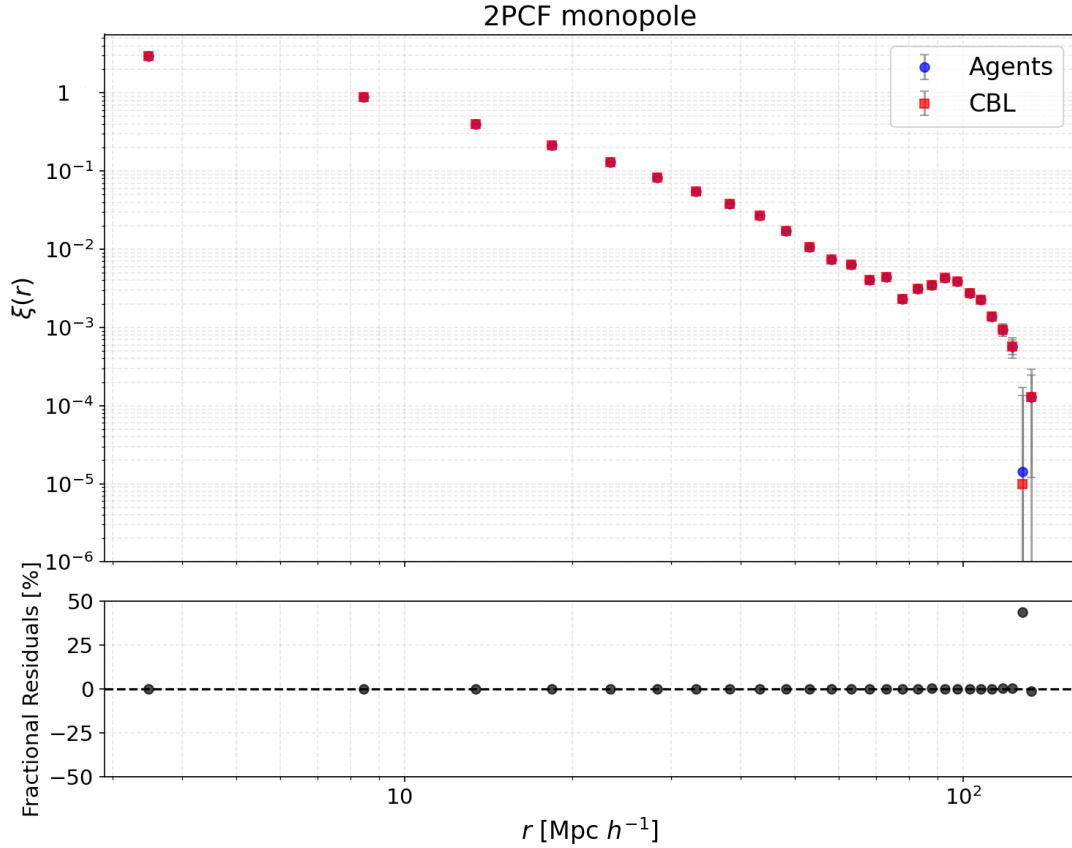


**Figure 6.2.** Scaling analysis comparing the computational time of the agents' optimized solution using `Corrfunc` (blue) versus the CBL benchmark 2PCF (red).

**Table 6.1.** Performance benchmark (runtime in seconds) for the 2PCF calculation across varying halo counts  $N$ . The AI agent utilized the `Corrfunc` implementation, while the baseline was computed using the CBL framework.

N (Halos)	CBL Runtime [s]	AI Agent Runtime [s]	Speedup Factor
10 000	3.0	0.3	9.4×
50 000	29.0	2.3	13.4×
100 000	101.0	8.8	11.5×
150 000	233.4	15.5	15.1×
200 000	393.0	27.0	14.5×
300 000	800.2	57.8	13.8×
400 000	1429.8	105.3	13.6×

Following the small-scale tests, the `Corrfunc`-based pipeline was applied to the full data set, containing 406 660 halos, and validated against the CBL results. As seen in Figure 6.3, except for small discrepancies at large scales caused by binning differences between the codes, the residuals between the two implementations are negligible, confirming that the agent's faster solution did not compromise accuracy.



**Figure 6.3.** Validation of the 2PCF computed by the agents (blue) against the CBL benchmark (red) on a catalog of 406 660 halos. The error bars represent Poissonian uncertainties, calculated by the agents as  $(1 + \xi)/\sqrt{DD}$ , consistent with the statistical approach used in the CBL baseline. The bottom panel shows the fractional residuals, calculated as  $(\xi_{Agent} - \xi_{CBL})/\xi_{CBL} \times 100$  [%].

The final step of the pipeline involves an example of the Vision Node analyzing a plot generated by the agents (such as the one in Figure 6.1). Listing 6.4 shows an output of the Vision Node.

**Listing 6.4.** Visual analysis of a 2PCF graph carried out by the Vision Node.

```

===== Ai Message =====

The graph you provided represents the 2-Point Correlation Function (2PCF) Monopole,
which is a key tool in cosmology for understanding the distribution of galaxies in
the universe. Here's a detailed breakdown of the graph:

### Axes
- X-axis (r [Mpc/h]): This axis represents the distance scale, measured in megaparsecs
  per unit of the Hubble constant (h). The range extends from 0 to 150 Mpc/h,
  indicating distances between galaxies or galaxy clusters.

- Y-axis (xi(r)): This axis shows the value of the 2PCF Monopole, denoted as xi(r). It
  quantifies the excess probability of finding a pair of galaxies at a distance r
  compared to a random distribution. The values range from approximately 0 to 3.

### Data Points
- The data points are plotted as blue dots, showing the correlation function values at
  various distances. The highest value appears at small distances (around 0 Mpc/h)
  and decreases rapidly as the distance increases.

### Trends and Implications
1. High Correlation at Small Distances: The initial high value of xi(r) at small
   distances suggests a strong clustering of galaxies. This is expected, as galaxies
   tend to form groups or clusters due to gravitational attraction.

2. Rapid Decline: The rapid decline in xi(r) as r increases indicates that the
   likelihood of finding pairs of galaxies decreases significantly with distance.
   This behavior is typical in large-scale structures of the universe, where galaxies
   are more clustered at smaller scales and more uniformly distributed at larger
   scales.

3. Asymptotic Behavior: The function approaches zero as r increases, suggesting that
   at very large distances, the distribution of galaxies becomes random, consistent
   with the cosmological principle that the universe is homogeneous and isotropic on
   large scales.

### Conclusion
Overall, the graph effectively illustrates the 2PCF Monopole, highlighting the
clustering behavior of galaxies and providing valuable insights into the

```

large-scale structure of the universe.

As seen in the agents' results, the pipeline is highly stable and scientifically sound. The Vision Node successfully identifies the axes, parses the asymptotic behavior, and accurately links the initial peak to gravitational clustering.

However, the pipeline remains entirely dependent on the precision of human language. To achieve the final stable, fast result, the user had to explicitly guide the agent away from its naive assumptions and actively prompt the inclusion of high-performance packages such as `Corrfunc`.

Moreover, while the Vision Node provides a generally good description of the image's features, it exhibits a significant lack of domain-specific knowledge. For instance, it erroneously attributes the 2PCF signal to galaxy clusters rather than dark matter halos. A human researcher would immediately recognize that the data represents a halo distribution, for example by generating a theoretical dark matter correlation function  $\xi_{dm}(r)$  and extracting the linear bias  $b$  by computing the ratio  $b^2 = \xi_{halo}/\xi_{dm}$ . This suggests that while the multimodal node is a powerful tool for parsing visual data like axes and slopes, it currently acts as an encyclopedic descriptor rather than a specialized analytical partner.

It is important to clarify that the system did not intrinsically optimize the computational complexity of the code, and that the agents' success was entirely driven by their ability to identify, select, and appropriately call pre-existing, highly optimized external libraries. A potential future development for this agentic framework could involve moving beyond library integration to attempt direct algorithmic optimization. Specifically, the agents could be tasked with analyzing and refining the low-level source code of established frameworks, such as the CBL libraries, to identify bottlenecks and implement novel computational optimizations.

### 6.1.2 Parameter Inference Analysis

After the successful measurement of the 2PCF, the agents were tasked with extracting cosmological parameters via Bayesian inference. Specifically, the analysis focused on the geometric scaling parameter,  $\alpha$ , and the clustering amplitude,  $b\sigma_8$ , which represents the product of the linear tracer bias and the normalization of the matter power spectrum (as formally defined in Eq. 5.1).

As noted in Chapter 5, the theoretical complexity of parameter inference initially confused the agents. Left unconstrained, earlier iterations frequently hallucinated naive power-law models instead of deploying an accurate Einstein-Boltzmann solver, requiring

the user to explicitly specify the library and the theoretical model to use. Additionally, as a time-saving measure, we instructed the agents to pre-compute the theoretical 2PCF template only once, before the MCMC loop.

Guided by the final prompt payload, the agent successfully generated the architectural syntax for the MCMC. It automatically decided to compute the theoretical model using `colossus` (Diemer, 2018) and `mcfits` (Li, 2019), as shown in Listing 6.5.

**Listing 6.5.** Agent-generated code for computing the theoretical 2PCF template. The agent correctly initialized the Eisenstein-Hu power spectrum and performed the log-space Hankel transform outside of the MCMC loop.

```
# Initialize Planck18 cosmology
cosmo = cosmology.setCosmology('planck18')

# Define k range for power spectrum (in h/Mpc)
k_min, k_max = 1e-4, 10.0
k_array = np.logspace(np.log10(k_min), np.log10(k_max), 512)

# Compute matter power spectrum at z=0
P_k = np.array([cosmo.matterPowerSpectrum(k) for k in k_array])

# Use mcfits (FFTLog) to transform P(k) -> xi(r)
p2xi = P2xi(k_array, l=0, lowring=True)
r_template, xi_template = p2xi(P_k)

# Interpolate template to match observed r values
xi_template_interp = np.interp(r_obs, r_template, xi_template)
xi_fid = xi_template_interp.copy()
r_fid = r_obs.copy()
```

As explained in the official `colossus` documentation<sup>4</sup>, the `matterPowerSpectrum` method defaults to the Eisenstein-Hu model (Eisenstein & Hu, 1998), which the agents selected as requested by the user.

With the template pre-computed, the agents were then able to define the theoretical model and the corresponding Gaussian likelihood. As shown in Listing 6.6, the agents successfully translated the physical model into Python syntax, correctly applying  $\alpha$ ,  $b\sigma_8$ , and the broadband polynomial nuisance parameters as defined in Eq. (5.1).

**Listing 6.6.** Agent-generated theoretical model. The agent successfully interpolated the pre-computed template to apply the geometric shift ( $\alpha$ ) and algebraically combined it with the amplitude bias and nuisance polynomials.

<sup>4</sup>[https://bdiemer.bitbucket.io/colossus/cosmology\\_\\_cosmology.html](https://bdiemer.bitbucket.io/colossus/cosmology__cosmology.html)

```

def model_xi(params, r, xi_template):
    alpha, b_sigma8, A0, A1, A2 = params

    # Scale distances by alpha
    r_scaled = r / alpha

    # Interpolate template at scaled distances
    xi_scaled = np.interp(r_scaled, r_fid, xi_template,
                          left=xi_template[0], right=xi_template[-1])

    # Apply bias scaling
    xi_model = (b_sigma8**2 / sigma8_fid**2) * xi_scaled

    # Add polynomial correction
    poly_correction = A0 + A1 / r + A2 / (r**2)
    xi_model += poly_correction

    return xi_model

```

Finally, the agent integrated these functions into the `emcee` (Foreman-Mackey et al., 2019) framework. To ensure convergence across the large uniform prior ranges, the agent decided to initialize the walkers in a tight Gaussian ball around a sensible physical baseline and dynamically clip them to respect the prior boundaries (Listing 6.7).

**Listing 6.7.** Initialization of the MCMC walkers using the `emcee` library.

```

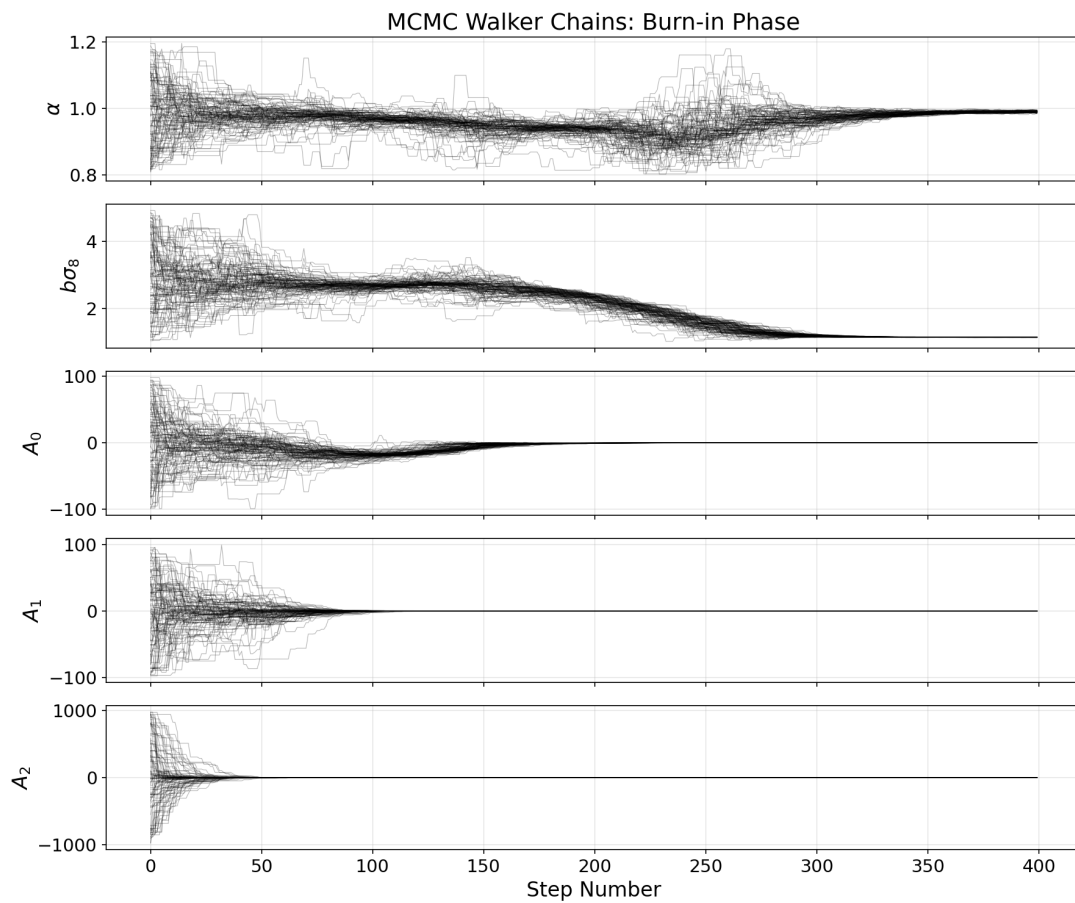
# Initialize walkers near center of prior ranges
initial_params = np.array([1.0, 2.0, 0.0, 0.0, 0.0]) # alpha, b_sigma8, A0, A1, A2
pos = initial_params + 1e-3 * np.random.randn(nwalkers, ndim)

# Ensure initial positions are within priors
pos[:, 0] = np.clip(pos[:, 0], 0.8, 1.2) # alpha
pos[:, 1] = np.clip(pos[:, 1], 1.0, 5.0) # b_sigma8
pos[:, 2] = np.clip(pos[:, 2], -100, 100) # A0
pos[:, 3] = np.clip(pos[:, 3], -100, 100) # A1
pos[:, 4] = np.clip(pos[:, 4], -1000, 1000) # A2

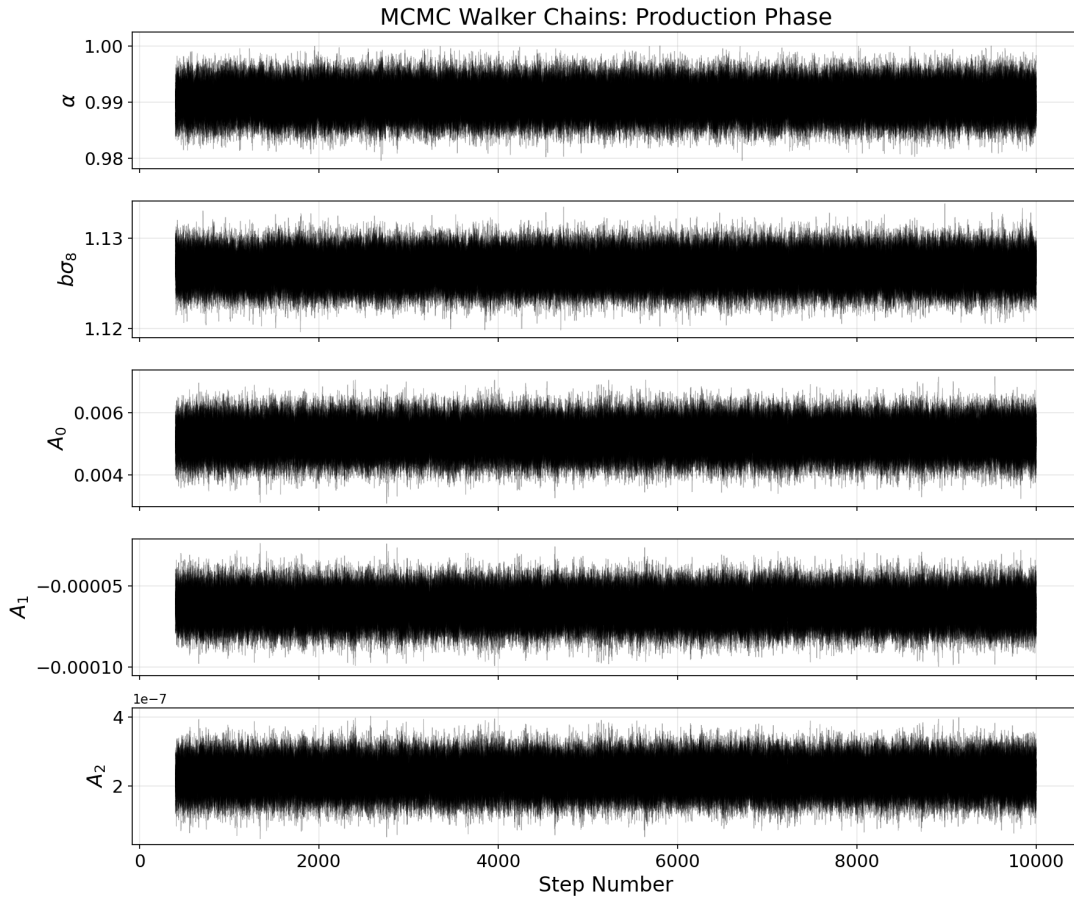
# Create sampler
sampler = emcee.EnsembleSampler(
    nwalkers, ndim, log_probability,
    args=(r_obs, xi_obs, inv_cov, xi_fid)
)
sampler.run_mcmc(pos, chain_size)

```

As observed in Figure 6.4, the walkers reached the high-probability region of the parameter space within the first few hundred steps, demonstrating an efficient burn-in process. Following this initial convergence, Figure 6.5 shows the production phase where the chains oscillate around a stable mean. This overlapping, dense behavior is characteristic of an MCMC sampler that is effectively exploring the posterior distribution.



**Figure 6.4.** Trace plots showing the initial burn-in phase of the MCMC walkers. The vertical movement at the start illustrates the transition from the initial Gaussian ball toward the region of maximum likelihood.



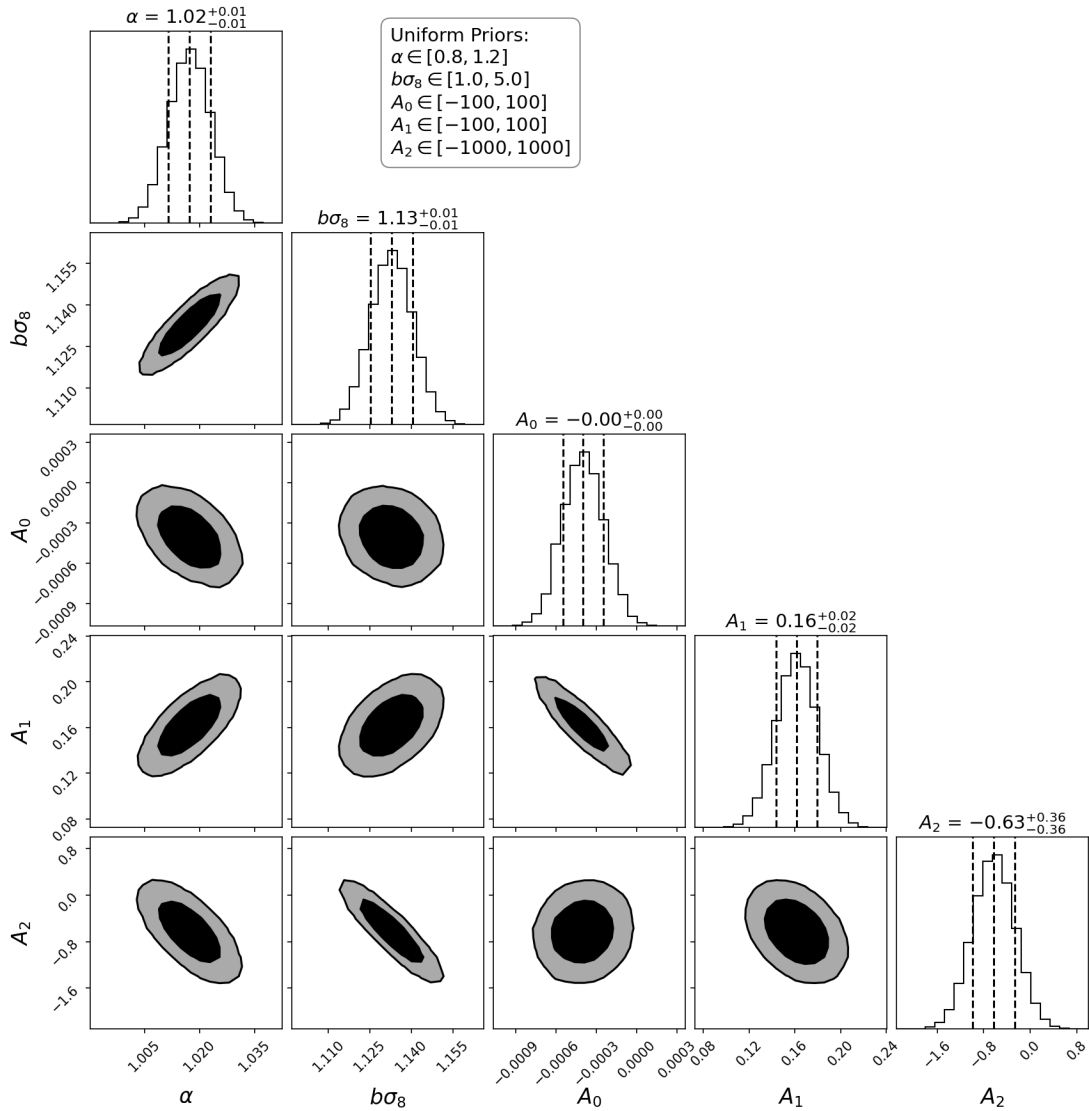
**Figure 6.5.** Trace plots showing the production phase of the MCMC walkers. The stationary, “fuzzy caterpillar” appearance indicates that the chains have reached a state of successful convergence.

For the baseline comparison, a similar analysis was carried out using the CBL. This implementation relied directly on two CBL example codes: the first runs the MCMC analysis with a 2PCF monopole BAO model<sup>5</sup>, while the second analyzes the output chains and visualizes the posterior distributions<sup>6</sup>.

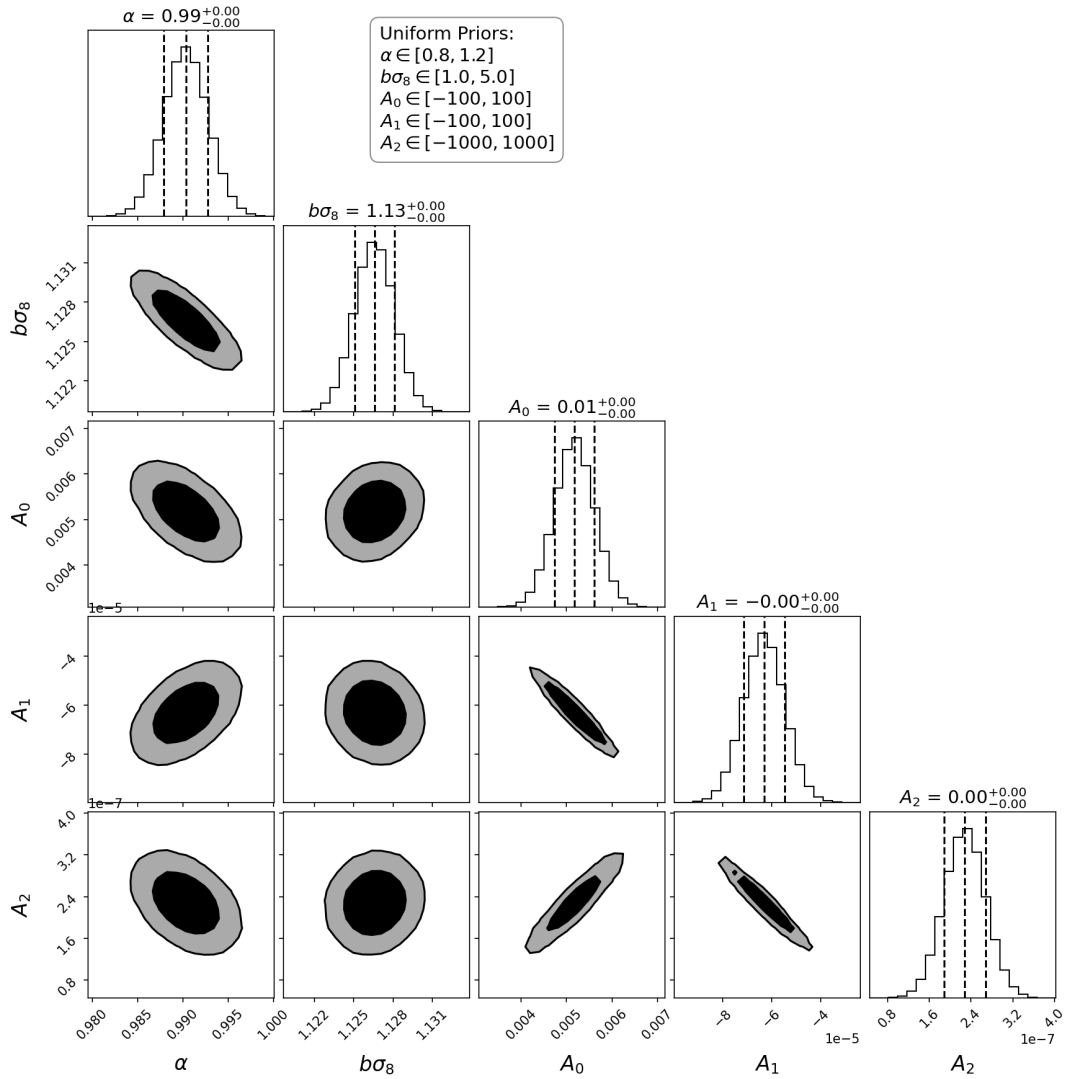
The full multi-dimensional parameter space is shown as a corner plot in Figure 6.6 for CBL’s results and Figure 6.7 for the agents’, while Table 6.2 presents the marginalized posterior constraints for both the CBL ground truth and the agents.

<sup>5</sup>[https://federicomarulli.github.io/CosmoBolognaLib/Doc/html/dc/d7d/model\\_2pt\\_monopole\\_BAO\\_8cpp-example.html](https://federicomarulli.github.io/CosmoBolognaLib/Doc/html/dc/d7d/model_2pt_monopole_BAO_8cpp-example.html)

<sup>6</sup>[https://federicomarulli.github.io/CosmoBolognaLib/Doc/html/db/dca/analyzeChains\\_8ipyb-example.html](https://federicomarulli.github.io/CosmoBolognaLib/Doc/html/db/dca/analyzeChains_8ipyb-example.html)



**Figure 6.6.** Marginalized posteriors and 2D contours for the CBL baseline. Wide uniform priors were applied:  $\alpha \in [0.8, 1.2]$ ,  $b\sigma_8 \in [1.0, 5.0]$ ,  $A_0, A_1 \in [-100, 100]$ , and  $A_2 \in [-1000, 1000]$ , as shown in the legend on the top right. The vertical dashed lines within the 1D histograms indicate the median and the 16th and 84th percentiles. The 2D graphs show the 68% and 95% credible regions for the parameters. The inferred constraints are tightly bounded well within the non-informative priors used.



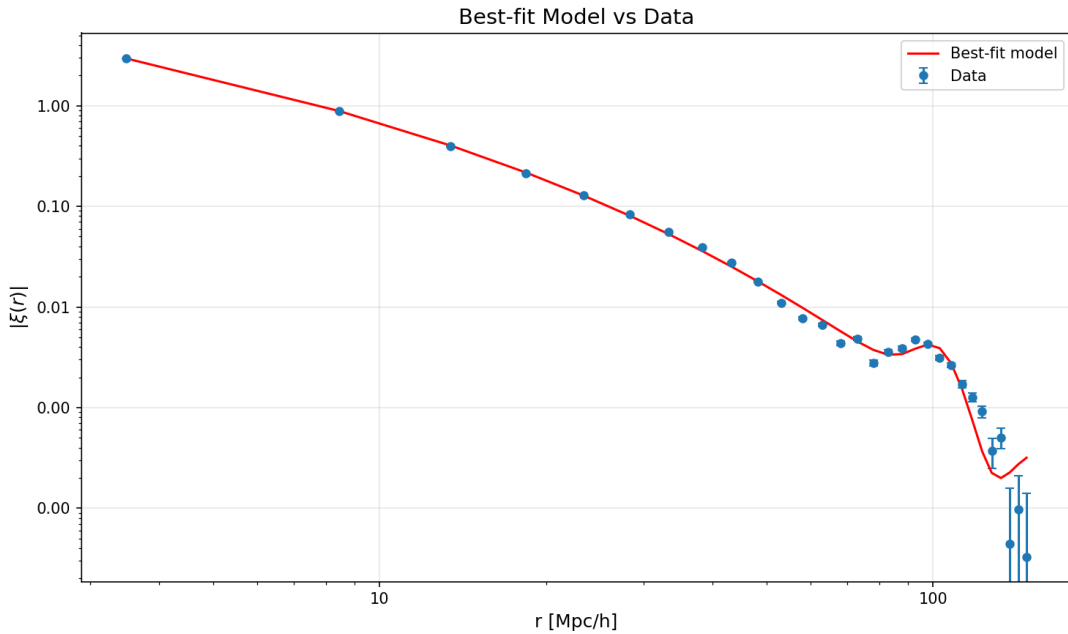
**Figure 6.7.** Marginalized posteriors and 2D contours for the agents, with a similar format to that of Figure 6.6. A comparison with the CBL results reveals slight systematic shifts across most parameter constraints.

While the MCMC chains converged to mathematically tight constraints, a rigorous quantitative comparison against the CBL baseline reveals a slight, systematic physical discrepancy that requires further investigation.

**Table 6.2.** Comparison of the posterior parameter constraints obtained from the CBL baseline analysis and the autonomous AI agent. The results report the median values and the 68% credibility intervals (16th and 84th percentiles).

Parameter	CBL (Baseline)	AI Agent
$\alpha$	$1.02 \pm 0.02$	$0.99 \pm 0.00$
$b\sigma_8$	$1.13 \pm 0.01$	$1.13 \pm 0.01$
$A_0$	$-0.00 \pm 0.00$	$0.01 \pm 0.00$
$A_1$	$0.16 \pm 0.02$	$-0.00 \pm 0.00$
$A_2$	$-0.63 \pm 0.36$	$0.00 \pm 0.00$

Additionally, the best-fit model computed by the agents is shown in Figure 6.8.



**Figure 6.8.** The best-fit theoretical model (solid line) generated by the agent, overlaid on the computed 2PCF data points. The error bars represent Poissonian errors.

The goodness-of-fit was assessed by calculating the reduced  $\chi^2$  statistic, defined as  $\chi^2_{\nu} = \chi^2/\text{dof}$ . With 30 data points and 5 free parameters ( $\alpha, b\sigma_8, A_0, A_1, A_2$ ), the degrees of freedom are  $\text{dof} = 25$ . The resulting value was  $\chi^2_{\nu} = 22.91$ . This high result can be attributed to the use of a simplified diagonal covariance matrix. By neglecting the strong off-diagonal correlations between  $r$ -bins and relying purely on Poissonian shot noise, the statistical uncertainties are likely underestimated, which leads to an inflated  $\chi^2$  value despite the close visual agreement between the model and the data.

points seen in Figure 6.8.

Following the methodology applied to the 2PCF measurement, the resulting corner plot was saved as a static image and passed to the Vision Node. It is important to emphasize that at this stage, as with the 2PCF case, the multimodal agent has no access to the numerical chain data or the previously computed statistics; its analysis is based solely on the visual interpretation of the plot, simulating the process of a researcher reviewing figures in a scientific publication. The agent generated the following contextual critique:

**Listing 6.8.** Output generated by the Vision Node analyzing the MCMC corner plot.

```

===== Ai Message =====

The graph you provided appears to be a corner plot, commonly used in cosmology and
statistics to visualize the posterior distributions of parameters from a Bayesian
analysis. Here's a detailed breakdown of what can be inferred from the plot:

### Structure of the Plot
1. Diagonal Histograms: Each diagonal entry shows the marginal distribution of a
parameter. The parameters represented are:
- alpha
- b_sigma8
- A0
- A1
- A2

Each histogram provides insights into the distribution of the parameter values,
including their central tendency and spread.

2. Off-Diagonal Contour Plots: The off-diagonal plots show the joint distributions
between pairs of parameters. The contours indicate the density of points sampled
in the parameter space, with darker regions representing higher density.

### Parameter Values and Uncertainties
- Each parameter has a value with associated uncertainties (e.g.,  $\alpha = 0.99^{+0.00}_{-0.00}$ ). This indicates the best estimate of the parameter along
with the 68% credible interval.

### Implications from a Physical Perspective
- Parameter Relationships: The correlations between parameters can be observed in
the off-diagonal plots. For instance, if your_label parameters show a strong
correlation, it may imply that they are related in the underlying physical model.

- Consistency: The values of the parameters should be checked against theoretical

```

```
predictions or previous measurements. If they align well, it strengthens the
validity of the model being tested.
```

```
### Prior Ranges
```

- ```
- The prior ranges provided in the legend indicate the bounds within which the
  parameters were constrained during the analysis. This is important for
  understanding the assumptions made in the modeling process.
```

```
### Improvements
```

- ```
1. Sampling Density: Increasing the number of samples could provide a more refined
  estimate of the parameter distributions, especially in regions of lower density.

2. Parameter Space Exploration: Exploring a wider range of prior distributions
  could help in understanding the robustness of the results. If the posterior
  distributions are sensitive to the priors, this should be reported.

3. Model Comparison: Including additional models or parameters could help in
  assessing the fit of the current model against alternatives, providing a more
  comprehensive understanding of the data.

4. Error Analysis: A more detailed error analysis could be beneficial, especially
  if systematic uncertainties are not accounted for.
```

The Vision Node analysis is encouraging, demonstrating an ability to read numerical distributions and constraints directly off the plot with significant accuracy. It correctly parses the correlation between parameters and accurately extracts the numerical means and standard deviations.

However, its critique remains highly encyclopedic, as it suggests generic statistical improvements without going into much detail. A human cosmologist reviewing this plot would instantly spot, for example, that the  $\alpha$  constraint diverged from an expected value of  $\alpha = 1$  and investigate the model's assumptions; the Vision Node, lacking this physical intuition, is unable to do that at the current version, and without further human help.

## 6.2 Discussion and Future Perspectives

The results presented in this chapter illustrate that high-performance LLMs can autonomously replicate the pipeline of a standard cosmological analysis, from data processing to parameter inference. The progression from the naive script in Listing 6.1 to the highly optimized routine in Listing 6.3 highlights the profound potential for AI-assisted coding to dramatically accelerate research workflows and lower the barrier to entry for complex numerical methods.

However, the blind spots encountered (particularly during the theoretical setup of the MCMC chains) underscore a critical issue: autonomous agents currently require substantial hand-holding from the user. While they excel at a few tasks such as library integration and optimization, they lack the intrinsic physical intuition required to evaluate their own systematic offsets. The LLM acts as a highly capable engine for code generation, but it still requires the human operator to act as the steering wheel, providing rigid physical constraints to prevent mistakes.

In order to transition these systems from assisted-coding tools into fully autonomous scientific researchers, future work must focus on expanding the agent’s contextual awareness and flexibility:

- **Integration of Retrieval-Augmented Generation:** By granting the agent access to a vector database of published cosmological papers and specialized library documentation, the model could autonomously research necessary physical corrections and numerical safeguards rather than relying solely on generalized pre-trained weights;
- **Automated Algorithmic Optimization:** Future implementations could focus on expanding the agent’s capabilities to perform direct optimization on established tools. For example, the system could be tasked with analyzing and refining the C++ source code of the CBL libraries to implement more efficient data structures or parallelization strategies, effectively acting as an automated software engineer for cosmological frameworks;
- **Modularization of the Graph Architecture:** The current pipeline could be expanded into a more granular ecosystem of specialized micro-agents. This might allow for more complex analyses within a highly controlled environment, where specialized nodes provide rigorous validation for each stage of the cosmological workflow;
- **Implementation of Robust Covariance Estimators:** Expanding the statistical pipeline to autonomously implement rigorous spatial resampling methods, such as Jackknife or Bootstrap techniques, to properly capture off-diagonal correlations and large-scale statistical errors;
- **Multipole Expansion and Anisotropic BAO Analysis:** Extending the analytical pipeline beyond the 2PCF monopole to compute higher-order multipoles (such as the quadrupole and hexadecapole). Equipping the agents to perform

full anisotropic BAO analysis in order to break parameter degeneracies and yield significantly more precise cosmological constraints;

- **Testing Frontier Models:** As the capabilities of generative AI evolve, benchmarking this exact LangGraph architecture against newer models will be critical to determine if the deficit in physical intuition is a fundamental flaw of LLMs or simply a limitation of current-generation training.

This experiment shows that while we are not quite ready to crown these agents as independent “AI cosmologists”, they are already extremely useful tools. As long as a human expert is there to properly direct them, they can take a massive amount of heavy lifting out of the research workflow, particularly when it comes to coding, debugging, or drafting text, and hold real promise for the future of cosmology.



# Conclusions

In this work, we have analyzed the potential of LLMs as tools for automating cosmological analysis, focusing specifically on the clustering of dark matter halos. By constructing a multi-agent pipeline, we tested the boundaries of what current generative AI models can achieve in a highly specialized and mathematically rigorous scientific field.

A central aspect of this research was the use of generic, pre-trained models. These agents were tasked with conducting a full research workflow, from data processing and statistical measurement to parameter inference, without prior fine-tuning on cosmological data sets. By employing an agentic architecture based on the LangGraph framework, we evaluated the system's ability to serve as a stable, autonomous research assistant.

The experiment yielded several key insights:

- **Generative Capabilities:** The agents demonstrated a remarkable ability to write, execute, and debug code for 2PCF measurements and Bayesian inference via MCMC. The speed at which these models can generate functional, optimized software, often selecting high-performance libraries like `Corrfunc` and `emcee` autonomously, suggests that LLMs will likely remain the foundation for research automation in the future;
- **Scientific Accuracy and Intuition:** While the mathematical procedures were correctly translated into Python syntax, the results were not always perfectly centered on the expected physical values. This underscores a current deficit in physical intuition; the models can replicate the logic of a calculation but lack the intrinsic ability to evaluate the scientific validity of their outputs against established cosmological standards;
- **Self-Correction and Optimization:** The agents proved to be highly responsive to iterative feedback and user input when it came to improving the codes' performance. This suggests a future where AI agents could be utilized to autonomously optimize established frameworks, such as the CBL, to meet the demands of

next-generation, data-intensive surveys.

Ultimately, this thesis research proves that while current LLMs are capable of implementing complex scientific workflows with high efficiency, they are not yet ready to function as independent AI cosmologists. At this stage, they serve as a powerful engine for code generation and data processing, but the human researcher remains an essential component of the loop, providing the expert oversight and physical constraints necessary to guide the model away from hallucinations and toward scientifically rigorous results.

As the capabilities of generative AI continue to evolve, the integration of specialized knowledge through retrieval-augmented generation and modular agentic structures holds the promise of a future where AI can transition from a computational tool into a truly collaborative partner in the exploration of the LSS of the Universe.

# Bibliography

- Achiam, J., Adler, S., Agarwal, S., et al. 2023, arXiv preprint arXiv:2303.08774
- Aghanim, N., Akrami, Y., Ashdown, M., et al. 2020, *Astronomy and Astrophysics*, 641, A6, doi: [10.1051/0004-6361/201833910](https://doi.org/10.1051/0004-6361/201833910)
- Anderson, L., Aubourg, , Bailey, S., et al. 2014, *Monthly Notices of the Royal Astronomical Society*, 441, 24–62, doi: [10.1093/mnras/stu523](https://doi.org/10.1093/mnras/stu523)
- Anthropic. 2024, The Claude 3 Model Family: Opus, Sonnet, Haiku. [https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model\\_Card\\_Claude\\_3.pdf](https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf)
- Coles, P., & Lucchin, P. 2003, *Cosmology: The Origin and Evolution of Cosmic Structure* (Wiley). <https://books.google.it/books?id=BGYcivB1EtMC>
- Colvin, S., Jolibois, E., Ramezani, H., et al. 2025, Pydantic Validation, 2.12.5. <https://github.com/pydantic/pydantic>
- Diemer, B. 2018, *The Astrophysical Journal Supplement Series*, 239, 35, doi: [10.3847/1538-4365/aace8c](https://doi.org/10.3847/1538-4365/aace8c)
- Dodelson, S., & Schmidt, F. 2020, *Modern Cosmology*, 2nd edn. (Academic Press)
- Dubey, A., Jauhri, A., Pandey, A., et al. 2024, arXiv preprint arXiv:2407.21783
- Einstein, A. 1917, *Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften*, 142. <https://ui.adsabs.harvard.edu/abs/1917SPAW.....142E>
- Eisenstein, D. J., & Hu, W. 1998, *The Astrophysical Journal*, 496, 605–614, doi: [10.1086/305424](https://doi.org/10.1086/305424)
- Emerging Information & Technology Conference. ????, Mathematics for Machine Learning and Data Science, EITC. <http://>

[eitc.org/research-opportunities/new-media-and-new-digital-economy/  
ai-machine-learning-deep-learning-and-neural-networks/  
mathematics-for-ai-ml-and-data-science/mathematics-for-machine-learning-data-science](https://eitc.org/research-opportunities/new-media-and-new-digital-economy/ai-machine-learning-deep-learning-and-neural-networks/mathematics-for-ai-ml-and-data-science/mathematics-for-machine-learning-data-science)

Foreman-Mackey, D., Farr, W., Sinha, M., et al. 2019, *Journal of Open Source Software*, 4, 1864, doi: [10.21105/joss.01864](https://doi.org/10.21105/joss.01864)

Guth, A. H. 1981, *Phys. Rev. D*, 23, 347, doi: [10.1103/PhysRevD.23.347](https://doi.org/10.1103/PhysRevD.23.347)

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, *Nature*, 585, 357

Jiang, A. Q., Sablayrolles, A., Mensch, A., et al. 2023, arXiv preprint arXiv:2310.06825

Kirshner, R. P. 2004, *Proceedings of the National Academy of Science*, 101, 8, doi: [10.1073/pnas.2536799100](https://doi.org/10.1073/pnas.2536799100)

Landy, S. D., & Szalay, A. S. 1993, , 412, 64, doi: [10.1086/172900](https://doi.org/10.1086/172900)

Li, Y. 2019, mcfits: multiplicatively convolutional fast integral transforms, <https://github.com/eelregit/mcfits>, GitHub

Marulli, F., Veropalumbo, A., & Moresco, M. 2016, *Astronomy and Computing*, 14, 35–42, doi: [10.1016/j.ascom.2016.01.005](https://doi.org/10.1016/j.ascom.2016.01.005)

Mellier, Y., Abdurro'uf, Acevedo Barroso, J. A., et al. 2025, *Astronomy and Astrophysics*, 697, A1, doi: [10.1051/0004-6361/202450810](https://doi.org/10.1051/0004-6361/202450810)

Mitchell, T. M. 1997, *AI Magazine*, 18, 11, doi: [10.1609/aimag.v18i3.1303](https://doi.org/10.1609/aimag.v18i3.1303)

Ramachandra, V. 2018, *Deep Learning for Causal Inference*. <https://arxiv.org/abs/1803.00149>

Ridnik, T., Kredo, D., & Friedman, I. 2024, *Code Generation with AlphaCodium: From Prompt Engineering to Flow Engineering*. <https://arxiv.org/abs/2401.08500>

Scolnic, D. M., et al. 2018, *The Astrophysical Journal*, 859, 101, doi: [10.3847/1538-4357/aab9bb](https://doi.org/10.3847/1538-4357/aab9bb)

Shinn, N., Cassano, F., Berman, E., et al. 2023, *Reflexion: Language Agents with Verbal Reinforcement Learning*. <https://arxiv.org/abs/2303.11366>

Sinha, M., & Garrison, L. 2020, *Monthly Notices of the Royal Astronomical Society*, 491, 3022

- Springel, V. 2005, Monthly Notices of the Royal Astronomical Society, 364, 1105, doi: [10.1111/j.1365-2966.2005.09655.x](https://doi.org/10.1111/j.1365-2966.2005.09655.x)
- The Dark Energy Survey Collaboration. 2018, The Astrophysical Journal Supplement Series, 239
- The MathWorks Inc. 2024, What Is Reinforcement Learning? <https://www.mathworks.com/discovery/reinforcement-learning.html>
- Vaswani, A., Shazeer, N., Parmar, N., et al. 2023, Attention Is All You Need. <https://arxiv.org/abs/1706.03762>
- Villaescusa-Navarro, F., et al. 2020, The Astrophysical Journal Supplement Series, 250, 2, doi: [10.3847/1538-4365/aba8fe](https://doi.org/10.3847/1538-4365/aba8fe)
- Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261, doi: [10.1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2)
- Vogelsberger, M., et al. 2013, Illustris simulations project
- Wang, J., & Duan, Z. 2024, Agent AI with LangGraph: A Modular Framework for Enhancing Machine Translation Using Large Language Models. <https://arxiv.org/abs/2412.03801>

