An Analysis of Cosmological Models

Project work submitted by

Madhava Krishnan N S (22PGP17)

Shyam Chaand R S (22PGP30)

Under the Guidance of

Mr. B. Rajesh,

Assistant Professor,

Department of physics,

The American College, Madurai 625002

Department of PG Physics,

The American College,

(An autonomous institution affiliated to Madurai Kamaraj University)

Madurai – 625002

April 2024

Certificate

This is to certify that this project work entitled **"An Analysis of Cosmological Models"** is carried out under the guidance of **Mr. B. Rajesh**, Assistant Professor at Department of PG Physics by **Madhava Krishnan N S** (22PGP17) and **Shyam Chaand R S** (22PGP30) in partial fulfilment of the requirement for the award of the degree of Master of Science in Physics during the period 2022-2024.

……

Dr. (Mrs.) S. Paul Mary Deborrah,
Associate Professor &
Head of the Department (PG & Research),
The Department of Physics,
The American College, Madurai 625002

……..

Mr. B. Rajesh, (Guide)
Associate Professor,
The Department of physics,
The American College, Madurai 625002

Declaration

We hereby declare that this project work for the degree of Master of Science in Physics entitled **"An Analysis of Cosmological Models"** is our original work and submitted by us to the department of PG Physics, The American College, for the award of the degree of Master of Science in Physics during the period 2022-2024. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this or any other institute or university.

Place: The American College, Madurai – 625002 ……

Date: (Madhava Krishnan N S)

……

(Shyam Chaand R S)

Acknowledgements

I extend my heartfelt gratitude to **Dr. (Mrs.) S. Paul Mary Deborrah**, Associate Professor & Head of the Department (PG & Research) in the Department of Physics at The American College, Madurai, for providing me with the invaluable opportunity to undertake this project.

I am sincerely grateful for **Mr. B. Rajesh**, Assistant Professor in the Department of Physics at The American College, Madurai, for his unwavering guidance and support throughout this project.

I would also like to express my deep appreciation for all other esteemed faculty members and every individual involved in this project, whose significant contributions and support have been pivotal to its success.

Their constant support, insightful guidance, and unwavering encouragement have been instrumental in the successful completion of this thesis, and I extend my deepest gratitude for their invaluable assistance and mentorship.

Abstract

This project aims to conduct a thorough analysis of cosmological models using Type Ia supernova (SN Ia) luminosity distance data. The primary focus is on estimating essential parameters such as the Hubble parameter (H_0), matter density parameter (Ω_{mo}), and equation of state parameter (ω) within the framework of the Flat-ΛCDM and Flat-wCDM models. Utilizing both chi-square minimization and Bayesian estimation techniques, we anticipate obtaining parameter estimates along with their associated uncertainties. Additionally, we expect the estimates of Ω_{mo} to reveal insights into the predominant composition of matter in the universe, aligning with theoretical predictions. In the context of the Flat-wCDM model, we anticipate uncovering intriguing insights into the nature of dark energy. Through a comparative analysis of results obtained from different estimation methods, we anticipate demonstrating the robustness of our findings, further enhanced by the consideration of confidence intervals. Ultimately, this study is poised to make significant contributions to our understanding of cosmological models and their implications for the evolution of the universe.

TABLE OF CONTENTS

CHAPTER – 1

INTRODUCTION

The cosmological model based on Cold Dark Matter (CDM), Friedmann equations, and the shape of the universe represents a fundamental framework for understanding the evolution, structure, and dynamics of the universe on cosmological scales. This model, grounded in the principles of General Relativity and observational cosmology, provides a comprehensive theoretical framework for describing the large-scale behavior of the cosmos and interpreting observational data from astronomical surveys and experiments.

At its core, the model relies on the Friedmann equations, a set of differential equations derived from Einstein's field equations of General Relativity, which describe the expansion of the universe over cosmic time. These equations govern the evolution of the cosmic scale factor, which quantifies the relative size of the universe, as well as the energy density and pressure of the various components of the cosmic fluid, including matter, radiation, and dark energy.

In the context of the CDM cosmological model, Cold Dark Matter (CDM) plays a central role as the dominant form of matter in the universe, accounting for the majority of the total mass content. CDM is characterized by its non-relativistic, collision less nature, which makes it an ideal candidate for explaining the observed large-scale structure of the universe, including the distribution of galaxies, galaxy clusters, and cosmic voids. The gravitational influence of CDM drives the growth of structure through gravitational collapse and hierarchical merging, leading to the formation of galaxies and galaxy clusters over cosmic time.

The shape of the universe, as described by the geometry of space-time, is another key aspect of the cosmological model. According to the Friedmann-Lemaître-Robertson-Walker (FLRW) metric, which serves as the mathematical framework for describing the universe on large scales, the shape of the universe can take on three possible geometries: flat, open, or closed. The determination of the universe's geometry has profound implications for its ultimate fate, expansion dynamics, and global topology.

1.1 Lambda cdm

The ΛCDM (Lambda Cold Dark Matter) cosmological model represents the prevailing paradigm for understanding the large-scale structure and evolution of the universe. In this model, Λ (Lambda) denotes the cosmological constant, which represents dark energy, while CDM stands for Cold Dark Matter, a form of matter that does not interact electromagnetically and is thus "cold" in the sense that its constituent particles move slowly relative to the speed of light.

One aspect of the ΛCDM model involves the concept of a transition redshift, which marks the epoch at which the expansion of the universe transitions from decelerating to accelerating. This transition is attributed to the dominance of dark energy, represented by the cosmological constant Λ, which drives the accelerated expansion of the universe. Understanding the transition redshift is crucial for probing the nature of dark energy and its influence on the cosmic expansion.

The Hubble parameter, denoted as H(z), quantifies the rate of expansion of the universe as a function of redshift (z). In the context of the ΛCDM model, the Hubble parameter is a key parameter that governs the evolution of the universe according to the Friedmann equations. By measuring the Hubble parameter as a function of redshift, cosmologists can constrain the cosmological parameters of the ΛCDM model and probe the expansion history of the universe.

The matter density parameter, Ω_{mo}, represents the fraction of the total energy density of the universe that is attributed to matter, including both baryonic matter and dark matter. In the ΛCDM model, Ω_{mo} plays a crucial role in determining the dynamics of structure formation and the evolution of cosmic large-scale structure. By constraining Ω_{mo} through observational data, such as galaxy surveys and cosmic microwave background observations, cosmologists can infer the distribution and properties of matter in the universe.

Luminosity distance, denoted as D_L(z), is a fundamental quantity in observational cosmology that characterizes the apparent brightness of astronomical sources as a function of redshift. In the ΛCDM model, the calculation of luminosity distances is essential for interpreting

observations of distant galaxies, supernovae, and other standard candles used to probe the cosmic expansion history. By comparing observed luminosity distances with theoretical predictions based on the ΛCDM model, cosmologists can test the consistency of the model and constrain its parameters.

The equation used to estimate cosmological parameters in the ΛCDM model is provided below:

$$d_L = c(1+z)\int_0^z \frac{dx}{H(x)}$$

Here,

d_L is the luminosity distance, c is the speed of light, z is the redshift

$H = H_0\sqrt{\Omega_{m0}(1+z)^3 + 1 - \Omega_{m0}}$

1.2 wCDM

The wCDM (w parameter Cold Dark Matter) cosmological model extends the ΛCDM model by introducing a time-varying equation-of-state parameter, w, to describe the properties of dark energy. In the context of the wCDM model, the equation-of-state parameter w characterizes the ratio of the pressure to the energy density of dark energy, allowing for a more flexible description of its dynamics compared to the ΛCDM model.

One key aspect of the wCDM model is the concept of a transition redshift, which marks the epoch at which the expansion of the universe transitions from decelerating to accelerating. This transition is influenced by the behavior of dark energy, as described by the equation-of-state parameter w. Understanding the transition redshift in the context of the wCDM model is crucial for probing the nature of dark energy and its influence on the cosmic expansion history.

The Hubble parameter, denoted as H(z), remains a fundamental quantity in the wCDM model, quantifying the rate of expansion of the universe as a function of redshift (z). Similar to the ΛCDM model, the Hubble parameter plays a central role in determining the evolution of the universe according to the Friedmann equations. By measuring the Hubble parameter as a function of redshift, cosmologists can constrain the cosmological parameters of the wCDM model and probe the expansion history of the universe.

The matter density parameter, Ω_{mo}, retains its significance in the wCDM model, representing the fraction of the total energy density of the universe attributed to matter, including both baryonic matter and dark matter. In the context of the wCDM model, Ω_{mo} influences the dynamics of structure formation and the evolution of cosmic large-scale structure. By constraining Ω_{mo} through observational data, such as galaxy surveys and cosmic microwave background observations, cosmologists can infer the distribution and properties of matter in the universe.

Luminosity distance, denoted as D_L(z), remains a crucial quantity in observational cosmology within the wCDM model, characterizing the apparent brightness of astronomical sources as a function of redshift. The calculation of luminosity distances is essential for interpreting observations of distant galaxies, supernovae, and other standard candles used to probe the cosmic expansion history within the wCDM framework. By comparing observed luminosity distances with theoretical predictions based on the wCDM model, cosmologists can test the consistency of the model and constrain its parameters.

The equation used to estimate cosmological parameters in the wCDM model is provided below:

$$d_L = c(1+z)\int_0^z \frac{dx}{H(x)}$$

Here,

d_L is the luminosity distance, c is the speed of light, z is the redshift

$$H = H_0\sqrt{\Omega_{m0}(1+z)^3 + (1-\Omega_{m0})(1+z)^{3(1+\omega)}}$$

CHAPTER – 2

METHODS

2.1 Statistical techniques:

2.1.1 Bayesian Estimation:

Bayesian estimation serves as a powerful framework for parameter inference and model comparison in cosmological models, allowing researchers to quantitatively assess the likelihood of different models given observational data and prior knowledge. In the context of cosmology, Bayesian estimation enables physicists to infer the values of cosmological parameters, such as the density of dark matter, the expansion rate of the universe, and the amplitude of primordial fluctuations, from observational data, such as galaxy surveys, cosmic microwave background measurements, and supernova observations.

The Bayesian framework comprises several key elements, each playing a crucial role in the parameter inference process:

1. Likelihood Function:

The likelihood function represents the probability of observing the data given a set of model parameters. In cosmology, the likelihood function encapsulates our theoretical understanding of how the model predicts observational data, accounting for uncertainties and statistical fluctuations. For example, in the context of galaxy surveys, the likelihood function may involve comparing the observed distribution of galaxies on the sky with predictions from cosmological models.

2. Prior Probability Distribution:

The prior probability distribution encapsulates our prior beliefs or knowledge about the model parameters before considering the observational data. Priors can be informed by theoretical considerations, previous experimental results, or other sources of information. In cosmology, priors may encode constraints from other cosmological measurements, such as the age of the universe or the abundance of light elements from Big Bang nucleosynthesis.

3. Posterior Probability Distribution:

The posterior probability distribution represents the updated probability distribution of the model parameters after incorporating the observational data. It is obtained by combining the likelihood function and the prior probability distribution using Bayes' theorem. In cosmology, the posterior distribution provides insights into the likelihood of different values of cosmological parameters given the observational data and prior knowledge.

4. Bayesian Evidence:

The Bayesian evidence, also known as the marginal likelihood, represents the probability of observing the data under the model, averaged over all possible parameter values weighted by the prior distribution. The Bayesian evidence plays a crucial role in model comparison, allowing researchers to assess the relative support for different cosmological models given the observational data.

In summary, Bayesian estimation provides a rigorous and flexible framework for parameter inference and model comparison in cosmological models. By combining observational data with prior knowledge and theoretical models, Bayesian methods enable physicists to quantitatively assess the likelihood of different scenarios, infer the values of cosmological parameters, and make robust predictions about the nature and evolution of the universe.

2.1.2 Chi^2 Analysis

The Chi-square Chi^2 analysis is a fundamental statistical technique used in cosmological model fitting, particularly in the context of parameter estimation and hypothesis testing. It serves as a powerful tool for quantifying the goodness-of-fit between theoretical models and observed data, enabling researchers to assess the compatibility of their models with observational constraints. Here's a detailed description of each element involved in Chi-square analysis in a cosmological model:

1. Observed Data (Oi):

These are the actual measurements or observations obtained from cosmological experiments or surveys. For example, in cosmology, observed data might include measurements of the cosmic microwave background radiation, galaxy redshift surveys, or supernova luminosity distances. Each data point is represented by O_i, where i indexes the individual data points.

2. Theoretical Model Prediction (Ti):

Theoretical models in cosmology predict the expected values of the observables based on a set of model parameters. For instance, in the context of the cosmic microwave background, theoretical models predict the temperature fluctuations across the sky as a function of various cosmological parameters such as the density of dark matter, dark energy, and baryonic matter, as well as the amplitude of primordial density fluctuations. Ti, represents the predicted value of the observable corresponding to the i-th data point.

3. Error or Uncertainty (σi):

Uncertainties are inherent in any observational measurement and are typically represented by the error bars associated with each data point. These uncertainties can arise from various sources, including instrumental noise, systematic errors, and statistical fluctuations. **σi** denotes the uncertainty associated with the i-th data point.

4. Residuals (Δi):

Residuals are the differences between the observed data and the corresponding theoretical model predictions, accounting for the uncertainties in the measurements. Mathematically, the residuals are calculated as $\Delta i = Oi - Ti$. Residuals quantify how well the model reproduces the observed data, with smaller residuals indicating a better fit between the model and the data.

5. Chi-square Statistic (Chi^2):

The Chi-square statistic is computed as the sum of the squared residuals normalized by the corresponding uncertainties. It is given by the formula:

$$\chi^2 = \sum_i \frac{(O_i - T_i)^2}{\sigma_i^2}$$

The Chi-square statistic quantifies the overall discrepancy between the observed data and the theoretical model predictions, taking into account both the deviations and the uncertainties associated with the measurements.

6. Degrees of Freedom (dof):

The degrees of freedom represent the number of independent data points used in the Chi-square analysis. In cosmological model fitting, the degrees of freedom are typically equal to the total number of data points minus the number of model parameters being constrained. It is denoted by dof.

7. Reduced Chi-square (Chi^2_{red}):

The reduced Chi-square statistic is obtained by dividing the Chi-square statistic by the degrees of freedom, (dof). It is given by:

$$\chi^2_{\text{red}} = \frac{\chi^2}{\text{dof}}$$

The reduced Chi-square provides a measure of the goodness-of-fit per data point, allowing for comparisons between different models and analyses. A reduced Chi-square close to unity indicates a good agreement between the model and the data, while values significantly larger than one suggest a poor fit.

By performing Chi-square analysis in cosmological model fitting, researchers can quantitatively assess the consistency of their models with observational data, identify the best-fit parameters, and evaluate the uncertainties associated with the parameter estimates. This statistical technique serves as a cornerstone in the validation and refinement of cosmological

models, enabling scientists to test hypotheses, make predictions, and advance our understanding of the universe.

When constructing confidence intervals in Chi-square analysis based on the number of degrees of freedom ((dof)) and a given p-value (probability), it's essential to consider the critical values from the Chi-square distribution. These critical values correspond to the boundaries that define the confidence interval and depend on both the significance level (1 - p-value) and the degrees of freedom.

1. Choose the Confidence Level (CL):

Determine the desired confidence level for the confidence interval. Common choices include 90%, 95%, and 99%.

2. Determine the Significance Level (α):

Calculate the significance level (α) corresponding to the chosen confidence level. For example, for a 95% confidence level, α = 1 - {CL} = 0.05.

3. Identify the Critical Value :

Look up the critical value from the Chi-square distribution table or use statistical software to find the critical value corresponding to the significance level (α/2) and degrees of freedom ((dof)). This critical value represents the upper boundary of the confidence interval.

4. Calculate the Lower Boundary:

Similarly, calculate the critical value corresponding to the significance level (1 - α/2) and degrees of freedom ((dof)). This critical value represents the lower boundary of the confidence interval.

5. Construct the Confidence Interval:

The confidence interval is then defined as

$$(\chi^2_{\alpha/2,\mathrm{dof}}, \chi^2_{1-\alpha/2,\mathrm{dof}})$$

Any Chi-square statistic falling within this interval is considered consistent with the null hypothesis at the chosen confidence level.

6. Interpretation:

When interpreting the confidence interval, note that it represents a range of values for which the null hypothesis (e.g., the model fits the data well) is not rejected with the chosen confidence level. If the calculated Chi-square statistic falls outside this interval, it suggests that the null hypothesis may be rejected in favor of an alternative hypothesis.

By following these steps and choosing an appropriate confidence level based on the desired level of certainty, researchers can construct confidence intervals in Chi-square analysis that provide valuable insights into the goodness-of-fit between theoretical models and observed data.

2.2 Computational techniques:

2.2.1 Markov chain Monte Carlo (MCMC)

Markov Chain Monte Carlo (MCMC) is a versatile computational technique widely used in Bayesian inference and probabilistic modeling, especially in cosmological model analysis and other scientific disciplines. MCMC methods allow researchers to sample from complex probability distributions, particularly those that are high-dimensional or analytically intractable, enabling estimation of posterior distributions, model fitting, uncertainty quantification, and hypothesis testing.

1. Objective of MCMC:

The primary objective of MCMC is to generate samples from a target probability distribution, typically the posterior distribution of model parameters in Bayesian inference. By simulating a Markov chain whose stationary distribution is the target distribution, MCMC methods provide a means to explore the parameter space and obtain representative samples from the posterior.

2. Markov Chains:

MCMC algorithms are based on Markov chains, stochastic processes where the probability of transitioning from one state to another depends only on the current state and not on previous states. In MCMC, the states of the Markov chain correspond to different parameter values in the parameter space.

3. Sampling Algorithm:

MCMC algorithms iteratively sample parameter values from the posterior distribution using a Markov chain. At each iteration, a proposed parameter value is generated based on a transition probability distribution, and its acceptance is determined probabilistically based on the posterior distribution and the current parameter value. Accepted samples are added to the Markov chain, and the process continues until a sufficient number of samples are obtained.

4. Gibbs Sampling:

One popular MCMC algorithm is Gibbs sampling, particularly useful for sampling from high-dimensional parameter spaces with conditional distributions that are tractable. In Gibbs sampling, parameter values are updated sequentially, with each update based on the conditional distribution of the parameter given the values of the other parameters.

5. Burn-in and Thinning:

In MCMC sampling, the initial samples may not be representative of the target distribution due to the algorithm's transient behavior. To mitigate this, a burn-in period is often employed, during which initial samples are discarded. Thinning, the process of retaining only every k-th sample to reduce autocorrelation, is also commonly used to improve the efficiency of MCMC sampling.

6. Posterior Inference:

Once a sufficient number of samples have been generated, statistical inference can be performed on the samples to estimate the posterior distribution of model parameters. This typically involves computing summary statistics such as the mean, median, and credible intervals

of the parameter values, as well as assessing convergence and mixing properties of the Markov chain.

2.2.2 Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm is a powerful Markov Chain Monte Carlo (MCMC) method widely used in cosmological model analysis to sample from complex probability distributions, particularly in scenarios where direct sampling or analytical methods are impractical. It serves as a cornerstone technique for Bayesian inference, enabling researchers to estimate the posterior distribution of model parameters given observed data.

1. Objective of Metropolis-Hastings Algorithm:

The primary objective of the Metropolis-Hastings algorithm is to generate a Markov chain that converges to the target distribution, typically the posterior distribution of model parameters in Bayesian analysis. By iteratively proposing and accepting or rejecting parameter values based on a set of transition probabilities, the algorithm explores the parameter space and produces samples from the target distribution.

2. Initialization:

The algorithm begins by initializing the Markov chain with an initial parameter value or set of parameter values. These initial values can be chosen arbitrarily or based on prior knowledge about the parameter space.

3. Proposal Distribution:

At each iteration of the algorithm, a new parameter value is proposed based on a proposal distribution. The proposal distribution defines the probability density function from which new parameter values are drawn. Common choices for the proposal distribution include Gaussian distributions centered around the current parameter value.

4. Acceptance Probability:

Once a new parameter value is proposed, its acceptance is determined based on an acceptance probability calculated using the Metropolis-Hastings acceptance criterion. This

criterion compares the posterior probabilities of the current and proposed parameter values, along with the transition probabilities from the current value to the proposed value and vice versa.

5. Acceptance or Rejection:

If the proposed parameter value has a higher posterior probability than the current value, it is always accepted. However, if the proposed value has a lower posterior probability, it may still be accepted with a certain probability determined by the acceptance probability criterion. If the proposed value is rejected, the current parameter value remains unchanged for the next iteration.

6. Iterative Sampling:

The algorithm iterates through multiple steps, generating a sequence of parameter values that form a Markov chain. As the number of iterations increases, the Markov chain converges to the stationary distribution, which approximates the target distribution (i.e., the posterior distribution of model parameters).

7. Posterior Inference:

Once a sufficient number of samples have been generated, statistical inference can be performed on the samples to estimate the posterior distribution of model parameters. This typically involves computing summary statistics such as the mean, median, and credible intervals of the parameter values, as well as assessing convergence and mixing properties of the Markov chain.

CHAPTER – 3

APPROACH

3 Approach:

3.1 Unpacking the Softwares:

3.1.1 Python

Python has emerged as a powerful and versatile tool for computational physics, offering a rich ecosystem of libraries and tools that enable researchers to tackle complex problems across a wide range of disciplines. With its intuitive syntax, extensive libraries, and active community support, Python has become the language of choice for many physicists seeking to model and simulate physical systems, analyze experimental data, and solve mathematical problems.

One of the key strengths of Python in computational physics lies in its extensive libraries for numerical computation and scientific computing. Libraries such as NumPy, SciPy, and Matplotlib provide comprehensive functionality for array manipulation, numerical integration, optimization, and data visualization, making it straightforward to implement numerical algorithms and analyze physical phenomena. These libraries offer efficient implementations of mathematical operations and numerical methods, allowing physicists to focus on the problem at hand rather than low-level details of implementation.

3.1.2 Jupyter notebook

Jupyter Notebook has revolutionized computational physics by providing a dynamic and interactive platform for conducting research, exploring theoretical concepts, and analyzing experimental data. Combining the capabilities of a traditional programming environment with the flexibility of a document-based interface, Jupyter Notebook offers physicists a powerful tool for developing, documenting, and sharing computational workflows in a seamless and interactive manner.

One of the key features of Jupyter Notebook is its support for literate programming, which allows physicists to combine code, narrative text, mathematical equations, and rich media content in a single interactive document. This enables researchers to create comprehensive and

self-contained computational narratives that not only describe the theoretical concepts and numerical methods employed but also provide detailed explanations, visualizations, and interpretations of results.

3.1.3 Google Colaboratory

Google Colab (short for Colaboratory) has become an invaluable tool for computational physics research, offering a convenient and powerful platform for developing, running, and sharing computational workflows in the cloud. Built on top of Google's infrastructure and integrated with popular data science libraries, Google Colab provides physicists with a seamless and collaborative environment for conducting numerical simulations, analyzing experimental data, and sharing research findings.

One of the key advantages of Google Colab is its accessibility and ease of use. Physicists can access Colab notebooks directly from their web browser without the need for any installation or setup, making it ideal for researchers who may not have access to high-performance computing resources or specialized software environments. Moreover, Colab provides free access to GPU and TPU resources, enabling physicists to accelerate computationally intensive tasks such as machine learning, numerical simulations, and data processing.

3.2 Loading the Libraries:

3.2.1 Matplotlib:

- Matplotlib is a versatile plotting library that allows you to create a wide range of visualizations, including line plots, scatter plots, histograms, and more.
- It offers extensive customization options for fine-tuning plot appearance and creating publication-quality figures.
- Matplotlib integrates seamlessly with other Python libraries, making it suitable for visualizing cosmological data and analysis results.

3.2.2 NumPy:

- NumPy is a fundamental package for numerical computing in Python.
- It provides support for multi-dimensional arrays, mathematical functions, random number generation, and linear algebra operations.
- NumPy arrays are efficient data structures for handling large datasets and performing computations required for cosmological analysis.

3.2.3 SciPy:

- SciPy is a library built on top of NumPy, offering additional functionality for scientific computing.
- It includes modules for optimization, interpolation, integration, signal processing, and statistical functions.
- SciPy's optimization and statistical tools are particularly useful for parameter estimation and hypothesis testing in cosmological analysis.

3.2.4 Pandas:

- Pandas is a powerful library for data manipulation and analysis, particularly suited for tabular and time-series data.
- It provides DataFrame objects for handling structured data, enabling tasks such as data cleaning, filtering, aggregation, and transformation.
- Pandas simplifies data preprocessing and organization, facilitating the preparation of datasets for analysis and visualization.

3.2.5 Corner:

- Corner is a Python package specialized in creating corner plots, also known as "triangle plots" or "pairwise posterior plots."
- These plots visualize multi-dimensional parameter spaces, showing pairwise correlations and marginal distributions of parameters.
- Corner plots are commonly used in cosmological parameter estimation to explore the posterior distributions of model parameters obtained from Bayesian inference.

By leveraging the capabilities of Matplotlib, NumPy, SciPy, Pandas, and Corner, cosmologists can effectively analyze observational data, perform statistical inference, visualize results, and gain insights into the underlying cosmological phenomena.

3.3 Python Codes:

3.3.1 Code for parameter estimation through chi-square minimization in the ΛCDM model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from numpy import *
from scipy.integrate import quad
from tqdm import tqdm

# Load data
h_data =
np.loadtxt('https://raw.githubusercontent.com/Madhavananalyst/pg_physics_project/main/panthe
on_binned.txt')

def g_t(t, a, b):
    return a * np.sqrt(b * (1 + t) ** 3 + 1 - b)

def integrand(t, a, b):
    return 1 / g_t(t, a, b)

def f_x_th(x, a, b):
    integrals = []
    c = 3 * 10**5
    for xi in x:
        result, error = quad(integrand, 0, xi, args=(a, b))
        integrals.append(result)
    return c * (1 + x) * np.array(integrals)

# Output file for parameters
f = open(r"lcdm.txt", 'w')
```

```
def range_with_floats(start, stop, step):
    while stop > start:
        yield start
        start += step

# Total iterations for progress tracking
total_iterations = int((100 - 30) / 0.01 * (0.5 / 0.001))

# Iterate through parameter combinations and track progress with tqdm
with tqdm(total=total_iterations, desc="Progress") as pbar:
    for h0 in range_with_floats(30, 100, 0.01):
        for om in range_with_floats(0.0, 0.5, 0.001):
            h_th = f_x_th(h_data[:, 0], h0, om)
            chi_sq = ((h_data[:, 1] - h_th) / h_data[:, 2]) ** 2
            t = round(h0, 3), round(om, 3), round(sum(chi_sq), 3)  # Round decimals
            t = str(t).replace(',', '')  # Remove commas
            f.write(str(t)[1:-1])  # Remove ()
            f.write("\n")
            pbar.update(1)  # Update progress bar

f.close()

# Read results into a DataFrame
df = pd.read_csv(r'lcdm.txt', sep=" ", names=["$H_0$", "$\Omega_{m0}$", "$\chi^2$"])

# Find parameters with minimum chi^2
df_min = df.loc[df['$\chi^2$'].idxmin()]

# Calculate confidence intervals
sigma = [1, 2, 3]
```

```
conf_int = [scipy.stats.chi2.cdf(s**2, 1) for s in sigma]
dof = range(1, 11)

for d in dof:
    chi_squared = [scipy.stats.chi2.ppf(ci, d) for ci in conf_int]

# Plot 3D scatter plots for different confidence levels
plt.figure(figsize=(12, 10))
plt.rcParams['font.weight'] = 'bold'
plt.rcParams['axes.labelweight'] = 'bold'
plt.xlabel('$H(z)~[km.Mpc^{-1}.sec^{-1}]$')
plt.ylabel("$\Omega_{m0}$")

df_1_sig = df[round(df["$\\chi^2$"], 1) == round((df_min[2] + 2.3), 1)]
df_2_sig = df[round(df["$\\chi^2$"], 1) == round((df_min[2] + 6.18), 1)]
df_3_sig = df[round(df["$\\chi^2$"], 1) == round((df_min[2] + 11.83), 1)]

plt.plot(df_1_sig['$H_0$'], df_1_sig['$\Omega_{m0}$'], '.', label='$68.27\%$ confidence level
contour')
plt.plot(df_2_sig['$H_0$'], df_2_sig['$\Omega_{m0}$'], '.', label='$95.47\%$ confidence level
contour')
plt.plot(df_3_sig['$H_0$'], df_3_sig['$\Omega_{m0}$'], '.', label='$99.73\%$ confidence level
contour')

plt.legend(loc="best", prop={'size': 10.70})
plt.annotate("$\chi^2_{\mathrm{min}}$", xy=(df_min[0], df_min[1]), xytext=(df_min[0],
df_min[1] + 0.04), arrowprops=dict(arrowstyle="->", mutation_scale=4))
plt.tight_layout()
plt.show()
```

3.3.2 Code for parameter estimation through chi-square minimization in the wCDM model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from numpy import *
from scipy.integrate import quad
from tqdm import tqdm

# Load data
h_data =
np.loadtxt('https://raw.githubusercontent.com/Madhavananalyst/pg_physics_project/main/panthe
on_binned.txt')

def g_t(z, H_o, omega_mo, w):
    return H_o * np.sqrt(omega_mo * (1 + z) ** 3 + (1 - omega_mo) * (1 + z) ** (3 * (1 + w)))

def integrand(z, H_o, omega_mo, w):
    return 1 / g_t(z, H_o, omega_mo, w)

def f_x_th(z, H_o, omega_mo, w):
    c = 3 * 10**5
    integrals = []
    for zi in z:
        result, error = quad(integrand, 0, zi, args=(H_o, omega_mo, w))
        integrals.append(result)
    return c * (1 + z) * np.array(integrals)

# Generate combinations of parameters
def range_with_floats(start, stop, step):
    while stop > start:
```

```
        yield start
        start += step

h0_values = list(range_with_floats(50, 100, 0.01))
om_values = list(range_with_floats(0.1, 0.5, 0.001))
w_values = list(range_with_floats(-3, 2, 0.001))

# Output file for parameters
f = open(r"omega_cdm_final_best.txt", 'w')

chi_sq1 = 0

# Total iterations for progress tracking
total_iterations = len(h0_values) * len(om_values) * len(w_values)

# Iterate through parameter combinations and track progress with tqdm
with tqdm(total=total_iterations, desc="Progress") as pbar:
    for h0 in h0_values:
        for om in om_values:
            for w in w_values:
                h_th = f_x_th(h_data[:, 0], h0, om, w)
                chi_sq = ((h_data[:, 1] - h_th) / h_data[:, 2]) ** 2
                t = round(h0, 3), round(om, 3), round(w, 3), round(sum(chi_sq), 3)
                t = str(t).replace(',', '')  # Remove commas
                f.write(str(t)[1:-1])  # Remove ()
                f.write("\n")
                pbar.update(1)  # Update progress bar

f.close()

# Read results into a DataFrame
```

```
df = pd.read_csv(r'final_omega_parameters.txt', sep=" ", names=["$H_0$", "$\Omega_{m0}$",
"w", "$\chi^2$"])

# Find parameters with minimum chi^2
df_min = df.loc[df['$\chi^2$'].idxmin()]

# Calculate confidence intervals
sigma = [1, 2, 3]
conf_int = [scipy.stats.chi2.cdf(s**2, 1) for s in sigma]
dof = range(1, 11)

for d in dof:
    chi_squared = [scipy.stats.chi2.ppf(ci, d) for ci in conf_int]

# Plot 3D scatter plots for different confidence levels
fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111, projection='3d')

df_1_sig = df[round(df['$\chi^2$'], 2) == round(df_min['$\chi^2$'] + 3.53, 2)]
df_2_sig = df[round(df['$\chi^2$'], 2) == round(df_min['$\chi^2$'] + 8.02, 2)]
df_3_sig = df[round(df['$\chi^2$'], 2) == round(df_min['$\chi^2$'] + 14.16, 2)]

ax.scatter(df_1_sig['$H_0$'], df_1_sig['$\Omega_{m0}$'], df_1_sig['w'], marker='.',
label='$68.27\%$ confidence level contour')
ax.scatter(df_2_sig['$H_0$'], df_2_sig['$\Omega_{m0}$'], df_2_sig['w'], marker='.',
label='$95.47\%$ confidence level contour')
ax.scatter(df_3_sig['$H_0$'], df_3_sig['$\Omega_{m0}$'], df_3_sig['w'], marker='.',
label='$99.73\%$ confidence level contour')

ax.set_xlabel('$H(z)~[km.Mpc^{-1}.sec^{-1}]$', fontweight='bold')
ax.set_ylabel("$\Omega_{m0}$", fontweight='bold')
```

```
ax.set_zlabel("w", fontweight='bold')

min_chi2_label = "$\chi^2_{\mathrm{min}}$"
arrow_properties = dict(facecolor='black', arrowstyle='->')
ax.quiver(df_min['$H_0$'], df_min['$\Omega_{m0}$'], df_min['w'], 0, 0.1, 0.1,
arrow_length_ratio=0.3, color='black')
ax.text(df_min['$H_0$'], df_min['$\Omega_{m0}$'] + 0.1, df_min['w'] + 0.1, min_chi2_label,
color='black', zorder=10)

ax.legend(loc="best", prop={'size': 10.70})

plt.tight_layout()
plt.show()
```

3.3.3 Code for parameter estimation through Bayesian estimation method in the ΛCDM model

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, uniform
import corner
from scipy.integrate import quad

# Load data
z, dl, dl_err =
np.loadtxt('https://raw.githubusercontent.com/Madhavananalyst/pg_physics_project/main/panthe
on_binned.txt', unpack=True)

# Define functions for calculations
def g_t(t, a, b):
    return ((a * np.sqrt(b * (1 + t) ** 3 + 1 - b)))
```

```
def integrand(t, a, b):
    return ((1 / g_t(t, a, b)))

def f_x_th(z, a, b):
    integrals = []
    c = 3 * 10**5
    for zi in z:
        result, error = quad(integrand, 0, zi, args=(a, b))
        integrals.append(result)
    return np.array(c * (1 + z) * integrals)

# Define log likelihood, prior, and posterior
def likelihood(theta, z, dl, dl_err):
    a, b = theta
    model = f_x_th(z, a, b)
    return np.sum(-0.5 * ((dl - model) / dl_err) ** 2 - 0.5 * np.log(2 * np.pi * dl_err ** 2))

def prior(theta):
    a, b = theta
    if 50 < a < 80 and 0.15 < b < 0.55:
        return np.log10(1.0 / ((80 - 50) * (0.55 - 0.15)))
    return -np.inf

def posterior(theta, z, dl, dl_err):
    lp = prior(theta)
    if not np.isfinite(lp):
        return -np.inf
    return (lp + likelihood(theta, z, dl, dl_err))

# Metropolis-Hastings algorithm
```

```
def Metropolis_Hastings(parameter_init, nsteps):
    result = []  # List to store the sampled parameter values
    result.append(parameter_init)  # Add the initial parameter values to the result list
    for t in range(nsteps):  # Iterate over the specified number of steps
        step_var = [1, 0.1]  # Variance of the proposal distribution for each parameter
        proposal = norm.rvs(loc=result[-1], scale=step_var)  # Generate a proposal parameter value from a normal distribution
        probability = np.exp(posterior(proposal, z, dl, dl_err) - posterior(result[-1], z, dl, dl_err))  # Calculate the acceptance probability
        if (uniform.rvs() < probability):  # Accept the proposal with the acceptance probability
            result.append(proposal)  # Add the proposal to the result list
        else:
            result.append(result[-1])  # Reject the proposal and add the previous parameter value to the result list
    return result  # Return the sampled parameter values

# Initial parameters and settings
a_ini, b_ini = 60, 0.25
initials = a_ini, b_ini
nsteps = 1000000

# Run Metropolis-Hastings algorithm
result = Metropolis_Hastings(initials, nsteps)
samples_MH = np.array(result)

# Plot traceplots
fig, axes = plt.subplots(2, 1, figsize=(8, 8))
samples = samples_MH.T

# Plot the traceplot of a
axes[0].plot(samples[0], "g")
```

```
axes[0].set_ylabel("$H_0$")

# Plot the traceplot of b
axes[1].plot(samples[1], "r")
axes[1].set_ylabel("$\Omega_{m0}$")

# Burn-in period
nburn_in = 5000
result_b = result[nburn_in:]
samples_MH_b = np.array(result_b)

# Plot traceplots after burn-in
fig, axes = plt.subplots(2, 1, figsize=(8, 8))
samples_b = samples_MH_b.T

# Plot the traceplot of a
axes[0].plot(samples_b[0], "g")
axes[0].set_ylabel("$H_0$")

# Plot the traceplot of b
axes[1].plot(samples_b[1], "r")
axes[1].set_ylabel("$\Omega_{m0}$")

# Extract parameter chains
a_chain = samples_MH_b[:, 0]
b_chain = samples_MH_b[:, 1]

# Estimate means and standard deviations
a_best = np.mean(a_chain)
b_best = np.mean(b_chain)
sig_a = np.std(a_chain)
```

```
sig_b = np.std(b_chain)

# Plot histograms
plt.figure(figsize=(8, 3))

# Plot the histogram of a
plt.subplot(1, 2, 1)
plt.hist(a_chain, bins=100, color='magenta')
plt.xlabel('$H_0$')
plt.ylabel('Count')

# Plot the histogram of b
plt.subplot(1, 2, 2)
plt.hist(b_chain, bins=100, color='green')
plt.xlabel('$\Omega_{m0}$')
plt.ylabel('Count')

# Plot corner plot
param_limits = [(70.5, 72.9), (0.235, 0.33)]

fig = corner.corner(samples_MH, bins=27, color="m", labels=['$H_0$', '$\Omega_{m0}$'],
truths=[a_best, b_best], fill_contours=True,
                levels=(0.68, 0.95, 0.99,), smooth=True, range=param_limits,
                quantiles=[0.16, 0.5, 0.84], title_fmt='.3f', plot_datapoints=False, show_titles=True)
```

3.3.4 Code for parameter estimation through Bayesian estimation method in the wCDM model

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, uniform
```

```
import corner
from scipy.integrate import quad

# Load data
z, dl, dlerr =
np.loadtxt('https://raw.githubusercontent.com/Madhavananalyst/pg_physics_project/main/panthe
on_binned.txt', unpack=True)

# Define functions for calculations
def g_t(x, H_o, omega_mo, w):
    return H_o * np.sqrt(omega_mo * (1 + x) ** 3 + (1 - omega_mo) * (1 + x) ** (3 * (1 + w)))

def integrand(x, H_o, omega_mo, w):
    return 1 / g_t(x, H_o, omega_mo, w)

def f_x_th(x, H_o, omega_mo, w):
    integrals = []
    for xi in x:
        result, error = quad(integrand, 0, xi, args=(H_o, omega_mo, w))
        integrals.append(result)
    return np.array(integrals)

# Define log likelihood, prior, and posterior
def ln_likelihood(theta, z, dl, dlerr):
    H_o, omega_mo, w = theta
    model = 3E5 * (1 + z) * f_x_th(z, H_o, omega_mo, w)
    return np.sum(-0.5 * ((dl - model) / dlerr) ** 2 - 0.5 * np.log(2 * np.pi * dlerr ** 2))

def ln_prior(theta):
    H_o, omega_mo, w = theta
    if 50 < H_o < 80 and 0.15 < omega_mo < 0.55 and -2 < w < 0:
```

```
        return np.log10(1.0 / ((80 - 50) * (0.55 - 0.15) * (0 + 2)))
    return -np.inf

def ln_posterior(theta, z, dl, dlerr):
    lp = ln_prior(theta)
    if not np.isfinite(lp):
        return -np.inf
    return lp + ln_likelihood(theta, z, dl, dlerr)

# Metropolis-Hastings algorithm
def Metropolis_Hastings(parameter_init, nsteps):
    result = [parameter_init]  # List to store the sampled parameter values
    for t in range(nsteps):  # Iterate over the specified number of steps
        step_var = [1, 0.05, 0.1]  # Variance of the proposal distribution for each parameter
        proposal = norm.rvs(loc=result[-1], scale=step_var)  # Generate a proposal parameter value
from a normal distribution
        probability = np.exp(ln_posterior(proposal, z, dl, dlerr) - ln_posterior(result[-1], z, dl,
dlerr))  # Calculate the acceptance probability
        if uniform.rvs() < probability:  # Accept the proposal with the acceptance probability
            result.append(proposal)  # Add the proposal to the result list
        else:
            result.append(result[-1])  # Reject the proposal and add the previous parameter value to
the result list
    return result  # Return the sampled parameter values

# Initial parameters and settings
H_o_ini, omega_mo_ini, w_ini = 60, 0.25, -0.9
initials = H_o_ini, omega_mo_ini, w_ini
nsteps = 1000000

# Run Metropolis-Hastings algorithm
```

```
result = Metropolis_Hastings(initials, nsteps)
samples_MH = np.array(result)

# Plot traceplots
fig, axes = plt.subplots(3, 1, figsize=(8, 12))
samples = samples_MH.T
axes[0].plot(samples[0], "g")
axes[0].set_ylabel("$H_o$")
axes[1].plot(samples[1], "r")
axes[1].set_ylabel("$\Omega_{mo}$")
axes[2].plot(samples[2], "r")
axes[2].set_ylabel("$\omega$")

# Burn-in period
nburn_in = 5000
result_b = result[nburn_in:]
samples_MH_b = np.array(result_b)

# Plot traceplots after burn-in
fig, axes = plt.subplots(3, 1, figsize=(8, 12))
samples_b = samples_MH_b.T
np.shape(samples_b)
axes[0].plot(samples_b[0], "g")
axes[0].set_ylabel("$H_o$")
axes[1].plot(samples_b[1], "r")
axes[1].set_ylabel("$\Omega_{mo}$")
axes[2].plot(samples_b[2], "r")
axes[2].set_ylabel("$\omega$")

# Extract parameter chains
H_o_chain = samples_MH_b[:, 0]
```

```
Omega_mo_chain = samples_MH_b[:, 1]
w_chain = samples_MH_b[:, 1]

# Estimate means and standard deviations
H_o_best = np.mean(H_o_chain)
Omega_mo_best = np.mean(Omega_mo_chain)
w_best = np.mean(w_chain)
sig_H_o = np.std(H_o_chain)
sig_Omega_mo = np.std(Omega_mo_chain)
sig_w = np.std(w_chain)

# Plot histograms
plt.figure(figsize=(12, 3))
plt.subplot(1, 3, 1)
plt.hist(H_o_chain, bins=100, color='blue')
plt.xlabel('$H_o$')
plt.ylabel('Count')
plt.subplot(1, 3, 2)
plt.hist(Omega_mo_chain, bins=100, color='blue')
plt.xlabel('$\Omega_{mo}$')
plt.ylabel('Count')
plt.subplot(1, 3, 3)
plt.hist(w_chain, bins=100, color='blue')
plt.xlabel('$\omega$')
plt.ylabel('Count')

# Plot corner plot
fig = corner.corner(samples_MH, bins=50, color="r", labels=['$H_o$', '$\Omega_{mo}$',
"$\omega$"], truths=[H_o_best, Omega_mo_best, w_best], fill_contours=True,
                    levels=(0.68, 0.95, 0.99,), smooth=True,
                    quantiles=[0.16, 0.5, 0.84], title_fmt='.3f', plot_datapoints=False, show_titles=True)
```

CHAPTER – 4

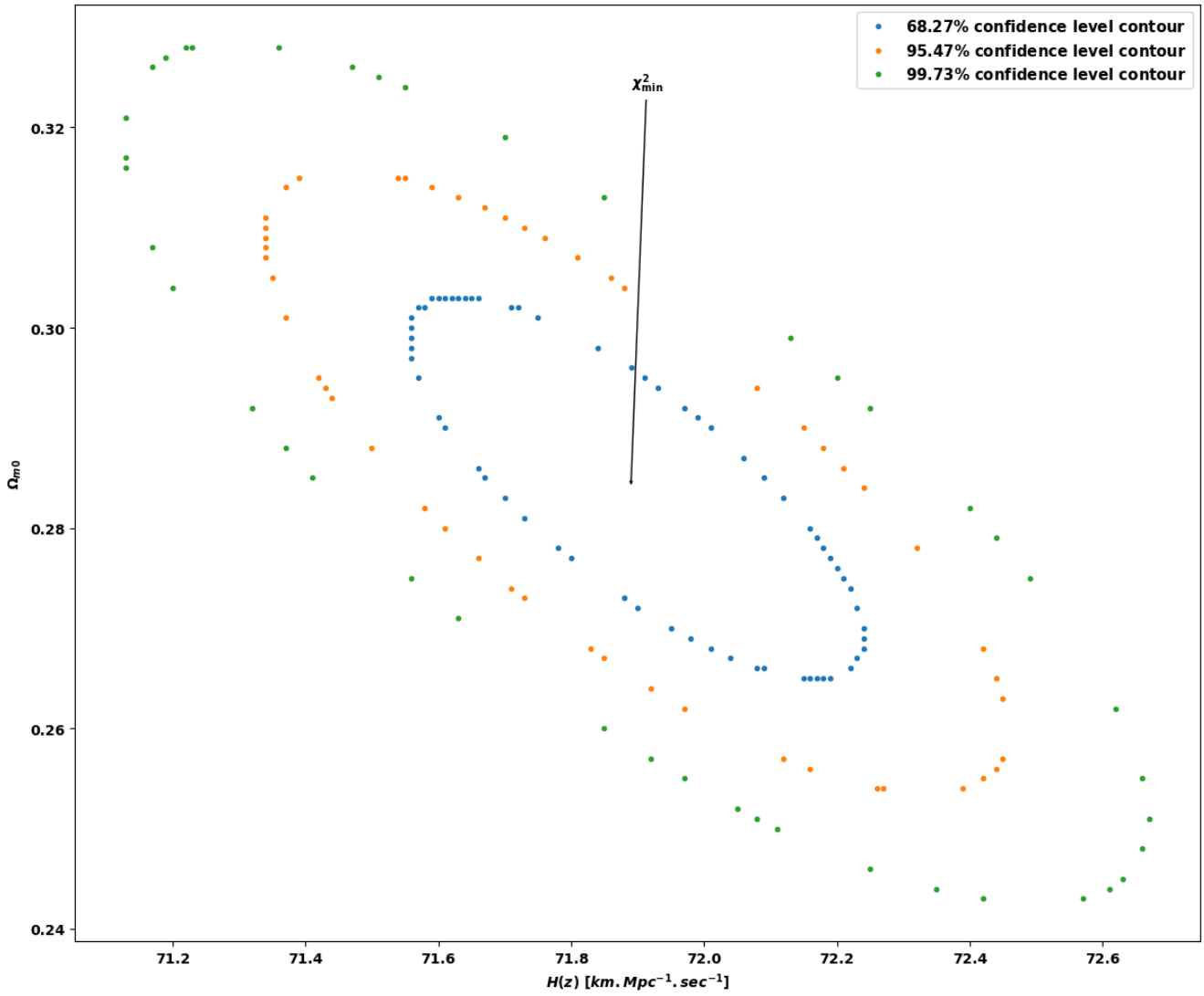
RESULTS AND DISCUSSIONS

4.1.1 Estimated Cosmological Parameters of the ΛCDM Model Using Chi-Square Minimization:

Models/ Parameters	ΛCDM model (χ2 minimization)
H_0 (km/s/Mpc)	71.89
Ω_{mo}	0.284

(Table 4.1.1 Estimated Cosmological Parameters of the ΛCDM Model Using Chi-Square Minimization)

4.1.2 Estimated Cosmological Parameters of the ΛCDM Model Using Chi-Square Minimization at Various Confidence Levels:



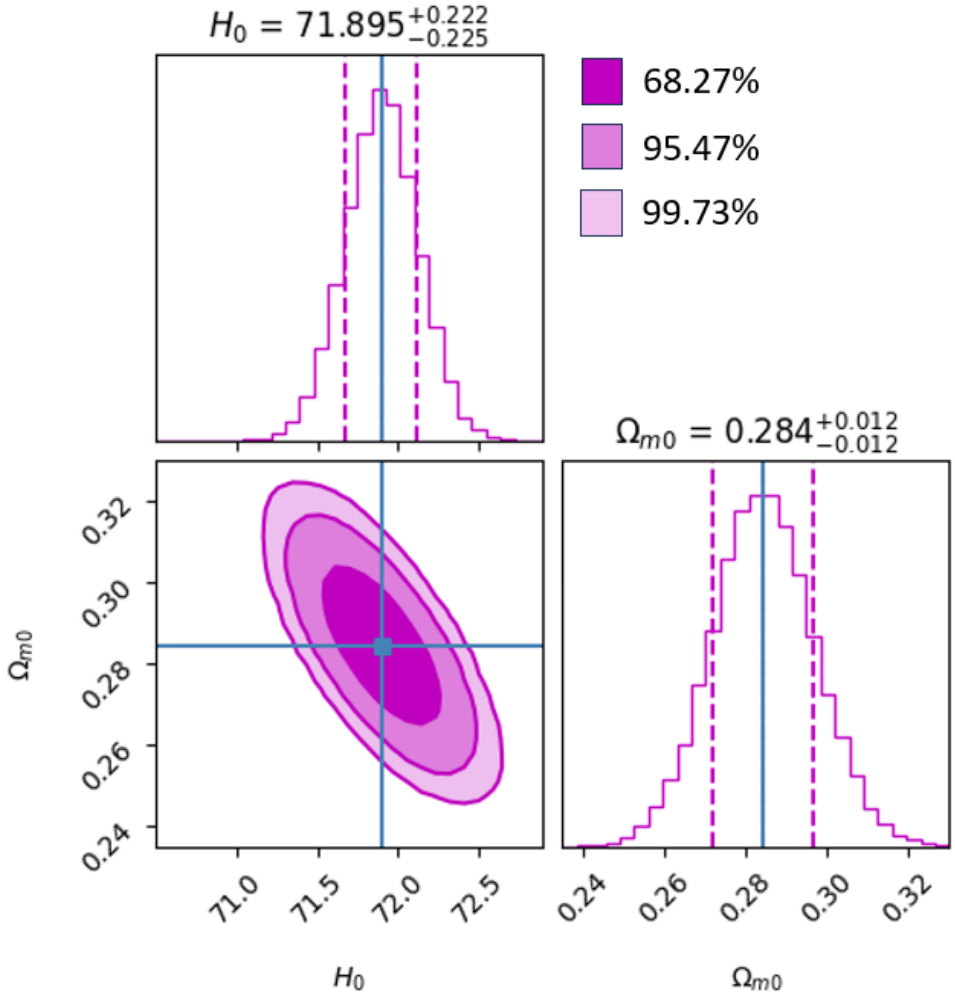
(Fig. 4.1.2 Estimated Cosmological Parameters of the ΛCDM Model Using Chi-Square Minimization at Various Confidence Levels)

4.2.1 Estimated Cosmological Parameters of the ΛCDM Model Using Bayesian Estimation method:

Models/ Parameters	ΛCDM model (Bayesian estimation)
H_0 (km/s/Mpc)	71.895
Ω_{mo}	0.284

(Table 4.2.1 Estimated Cosmological Parameters of the ΛCDM Model Using Bayesian Estimation method)

4.2.2 Estimated Cosmological Parameters of the ΛCDM Model Using Bayesian Estimation method at Various Confidence Levels:



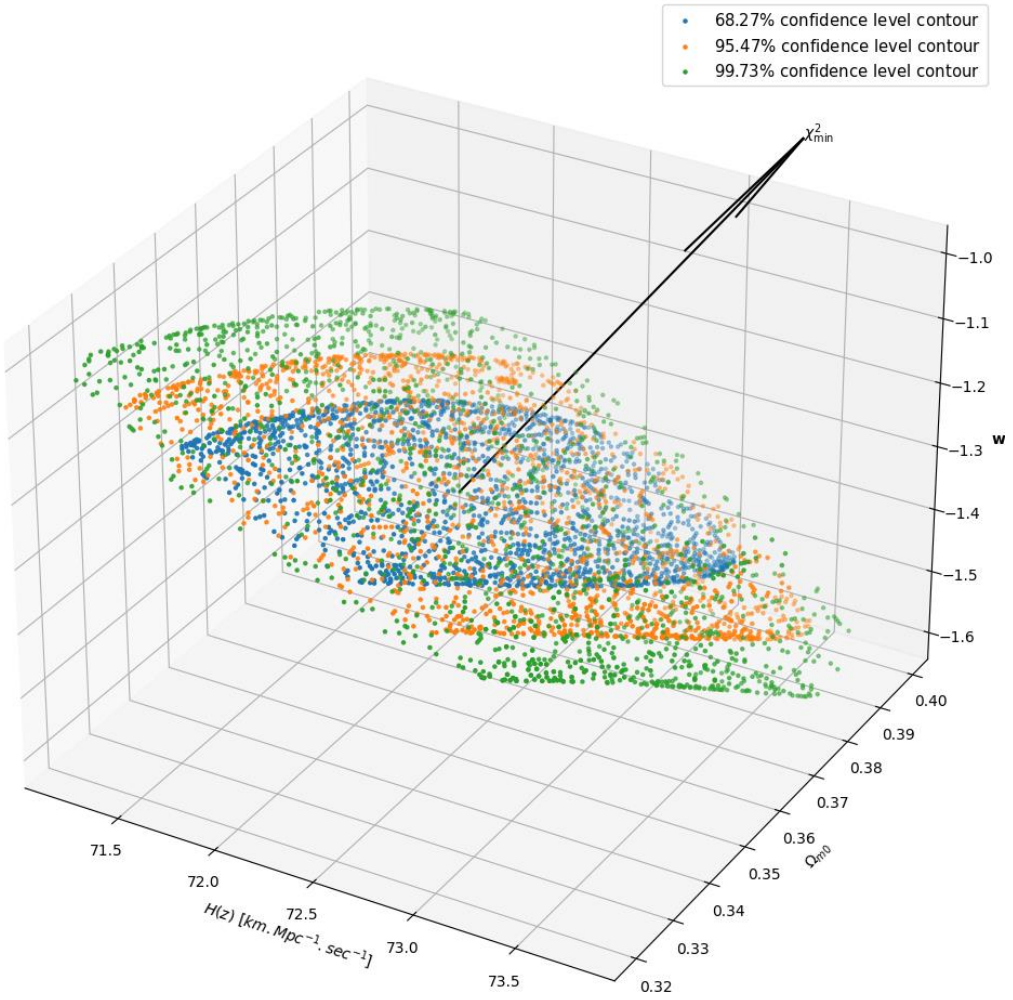
(Fig. 4.2.2 Estimated Cosmological Parameters of the ΛCDM Model Using Bayesian Estimation method at Various Confidence Levels)

4.3.1 Estimated Cosmological Parameters of the wCDM Model Using Chi-Square Minimization:

Models/ Parameters	wCDM model ($\chi 2$ minimization)
H_0 (km/s/Mpc)	72.43
Ω_{mo}	0.356
ω	-1.265

(Table 4.3.1 Estimated Cosmological Parameters of the wCDM Model Using Chi-Square Minimization)

4.3.2 Estimated Cosmological Parameters of the wCDM Model Using Chi-Square Minimization at Various Confidence Levels:



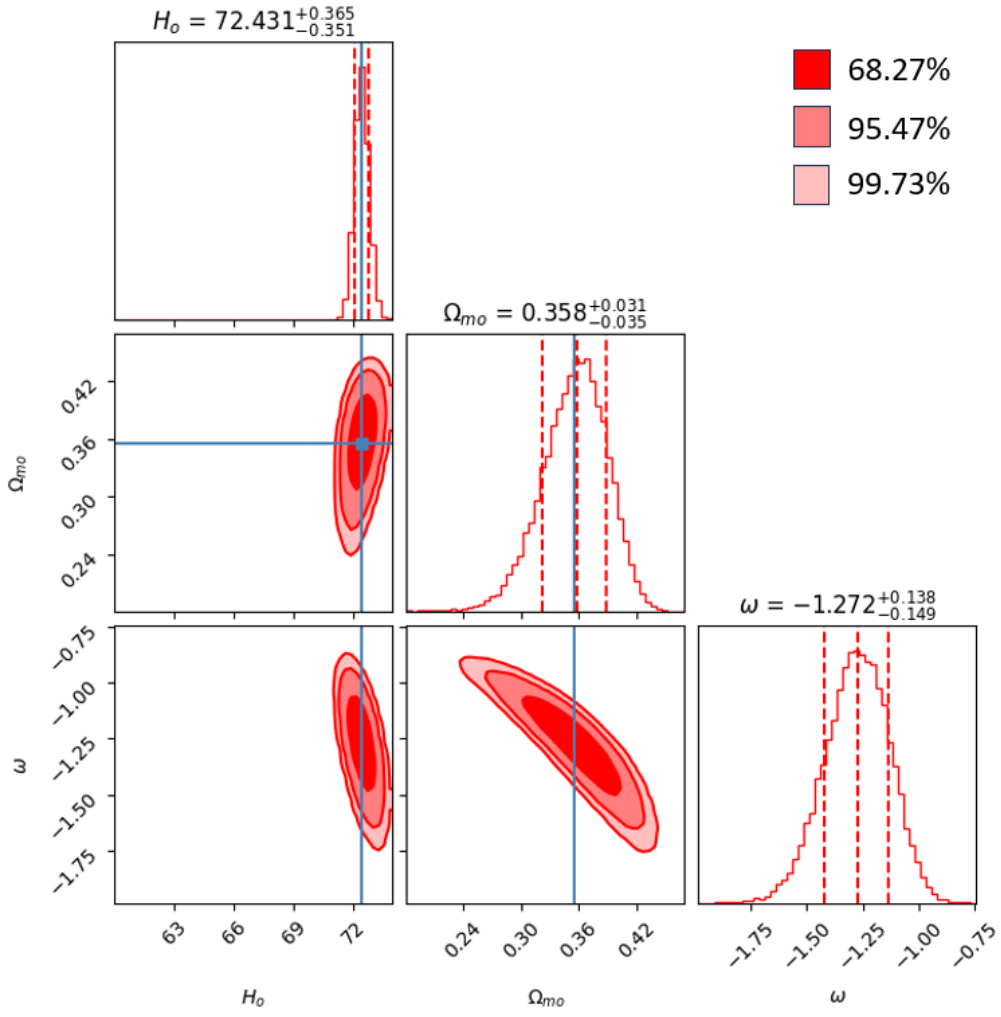
(Fig. 4.3.2 Estimated Cosmological Parameters of the wCDM Model Using Chi-Square Minimization at Various Confidence Levels)

4.4.1 Estimated Cosmological Parameters of the wCDM Model Using Bayesian Estimation method:

Models/ Parameters	wCDM model (Bayesian estimation)
H_0 (km/s/Mpc)	72.431
Ω_{mo}	0.358
ω	-1.272

(Table 4.4.1 Estimated Cosmological Parameters of the wCDM Model Using Bayesian Estimation method)

4.4.2 Estimated Cosmological Parameters of the wCDM Model Using Bayesian Estimation method at Various Confidence Levels:



(Fig. 4.4.2 Estimated Cosmological Parameters of the wCDM Model Using Bayesian Estimation method at Various Confidence Levels)

The Hubble parameter (H_0) represents the rate of expansion of the universe and is a crucial parameter for understanding its evolution. The values obtained for H_0 in both models are consistent with recent observations and provide valuable constraints on the expansion rate.

The matter density parameter (Ω_{mo}) indicates the proportion of matter in the universe relative to the critical density required for a flat universe. The values obtained suggest that the universe is predominantly composed of matter, consistent with the predictions of standard cosmological models.

The equation of state parameter (ω) characterizes the nature of dark energy in the universe. In the Flat-ΛCDM model, ω is not applicable as dark energy is assumed to be represented by a cosmological constant (Λ). However, in the Flat-wCDM model, ω provides insights into the dynamics of dark energy. The estimated values of ω suggest that dark energy exhibits a behavior consistent with a cosmological constant.

4.5 Comparison of Results:

Models/ Parameters	ΛCDM model (χ2 minimization)	ΛCDM model (Bayesian estimation)	wCDM model (χ2 minimization)	wCDM model (Bayesian estimation)
H_0 (km/s/Mpc)	71.89	71.895	72.43	72.431
Ω_{mo}	0.284	0.284	0.356	0.358
Ω	Assumed constant	Assumed constant	-1.265	-1.272

(Table 4.5 Comparison of estimated parameters)

Comparing the results obtained from the chi-square minimization method and Bayesian estimation method reveals consistency between the two approaches. Both methods yield similar values for the parameters, indicating robustness in the estimation process. Additionally, the confidence intervals provide a measure of the uncertainty associated with the parameter estimates, allowing for a more comprehensive interpretation of the results.

Values of Ω_{mo} less than 1 suggest an open universe, where matter is not the dominant driving force of expansion. In our analysis, the estimated value of Ω_{mo} for the Flat-ΛCDM model is

0.284 and for the Flat-wCDM model is 0.356, indicating a universe with a substantial amount of matter but not sufficient to dominate the expansion entirely.

Similarly, values of ω less than -1 suggest exotic phenomena like "phantom energy," accelerating the universe's expansion at an increasing rate over time. In our analysis, the estimated value of ω for the Flat-ΛCDM model is not applicable as dark energy is represented by a cosmological constant (Λ). However, for the Flat-wCDM model, the estimated value of ω is -1.265, suggesting a deviation from the behavior of a cosmological constant but not reaching the threshold of phantom energy. This indicates that the expansion of the universe may be accelerating, albeit at a slower rate than predicted by phantom energy.

CHAPTER – 5

CONCLUSION

Conclusion:

In conclusion, this study has presented a comprehensive analysis of cosmological models utilizing Type Ia supernova luminosity distance data. The primary aim was to estimate key parameters: the Hubble parameter, matter density parameter, and equation of state parameter within the framework of the Flat-ΛCDM and Flat-wCDM models using both chi-square minimization and Bayesian estimation methodologies.

The findings regarding the Hubble parameter (H_0) align with recent observational data, providing valuable insights into the universe's expansion rate. Furthermore, the estimated matter density parameter (Ω_{mo}) suggests that the universe is predominantly composed of matter, consistent with theoretical predictions.

In the context of the Flat-ΛCDM model, the equation of state parameter (ω) was found to be constant, since the curvature is assumed constant (by the model definition) and does not vary over time as dark energy is assumed to be represented by a cosmological constant (Λ). However, in the Flat-wCDM model, the curvature of universe is not assumed to be constant, however it's considered to be time evolving variable and our analysis reveals intriguing insights into the nature of dark energy, with estimated values of ω indicating behavior consistent with a cosmological constant.

Comparison between the results obtained from chi-square minimization and Bayesian estimation methods demonstrates consistency, underscoring the robustness of the estimation techniques employed. Additionally, the consideration of confidence intervals provides a nuanced understanding of parameter uncertainty, enhancing the interpretation of the results.

CHAPTER – 6

REFERENCES

References:

1. https://github.com/darshanbeniwal/Astrophy_Py_STACUP_BDU_CUTN_IUCAA_2023/blob/main/Text_files_Datasets/SNIa_40_dL_Pantheon_Binned.txt
2. Scolnic, S.M., Jones, D.O., Rest, A., et al.: The complete light-curve sample of spectroscopically confirmed sne ia from pan-starrs1 and cosmological constraints from the combined pantheon sample', Astrophys. J. 859, id. 101, 28 pp.
3. Choudhuri, A. R. (2010). Astrophysics for physicists. Cambridge University Press.
4. Colin, J., Mohayaee, R., Rameez, M., & Sarkar, S. (2019). Evidence for anisotropy of cosmic acceleration. Astronomy & Astrophysics, 631 , L13.
5. Guy, J., Astier, P., Baumont, S., Hardin, D., Pain, R., Regnault, N., . . . others (2007). Salt2: using distant supernovae to improve the use of type ia supernovae as distance indicators. Astronomy & Astrophysics, 466 (1), 11–21.
6. Kessler, R., Becker, A. C., Cinabro, D., Vanderplas, J., Frieman, J. A., Marriner, J., . . . others (2009). First-year sloan digital sky survey-ii supernova results: Hubble diagram and cosmological parameters. The Astrophysical Journal Supplement Series, 185 (1), 32.
7. Kessler, R., Bernstein, J. P., Cinabro, D., Dilday, B., Frieman, J. A., Jha, S., . . . Vanderplas, J. (2009, September). SNANA: A Public Software Package for Supernova Analysis. , 121 (883), 1028. doi: 10.1086/605984
8. Mohayaee, R., Rameez, M., & Sarkar, S. (2021). Do supernovae indicate an accelerating universe? The European Physical Journal Special Topics, 230 (9), 2067–2076.
9. Nielsen, J. T., Guffanti, A., & Sarkar, S. (2016). Marginal evidence for cosmic acceleration from type ia supernovae. Scientific reports, 6 (1), 1–8.
10. Tripp, R. (1998, March). A two-parameter luminosity correction for Type IA supernovae. , 331 , 815-820.