Summer School 2025

Astronomy & Astrophysics

PROJECT REPORT

Prepared by

Student name: Lalithkishore M
Institution: Vellore Institute of Technology, Chennai
Institution Roll No: 23BEC1088
ISA Admission Number: 366506

PROJECTS

Predicting the Hubble Parameter and the Age of the Universe using Supernovae Ia Data

Submitted To

Name: Mr. Sahil Sakkarwal
Designation: Program Supervisor
Institution: India Space Academy

INDEX

SESSION INFERENCES

Day 01: Data-Driven Astronomy

We looked at the various forms of astronomical data and how they aid in the study of the cosmos, including photos, spectra, catalogs, time-series, and data cubes. Platforms such as Gaia, MAST, and SDSS that make these datasets publicly available were presented to us. We discovered how analysis is carried out using the Virtual Observatory framework and technologies like Topcat, DS9, and Python. Simple projects such as using Kepler data to find exoplanets, Gaia to investigate star clusters, and JWST photos to identify galaxies were displayed. The invitation to experiment with accessible datasets and small, worthwhile initiatives using freely available tools was what truly caught my attention.

Day 02: Basics of Python

Because we used Jupyter Notebook, it was simpler to follow the code and view the results step-by-step. We started by going over the fundamentals of Python, including grammar, data types (such as floating, complex numbers, and integers), and arithmetic operations. We tried shorthand operations like a += 10 and learned about variables as containers to store values.
Data inspection and display were made easier by built-in functions like print() and type(). In order to illustrate how variables might interact in practical use situations, a straightforward tax calculator project was used. Additionally, we learned how to use.upper(),.lower(), and.split() to slice, index, and change strings—skills that are very helpful when working with text-based information in astronomy.
Then, in order to handle dynamic data, we turned to lists and associated methods, such as.append(),.insert(), and.pop().Nested lists gave us a glimpse of matrix-like data structures. Dictionaries followed, showing how key-value pairs simplify data organization and retrieval—like reading FITS headers or catalog metadata.
The session wrapped up with an introduction to NumPy, a core Python library for numerical computing. We created arrays, did element-wise operations, and learned about multidimensional arrays and matrix multiplication—essential for handling astronomical image and spectral data.

Day 03: FITS File Handling

Working with FITS files, the common data format in astronomy, gave us practical experience. We converted units and accessed astronomical constants using Astropy in Python. We were able to calculate object distances and comprehend celestial coordinates thanks to the SkyCoord item. Astropy.io.fits was used to access the FITS data, and we used HDUs to examine their structure. Matplotlib was used to visualize image data, and we even combined various filters to create RGB graphics. We were able to interpret brightness levels with the use of histograms. Additionally, we worked with 3D data cubes and loaded regions using DS9 for sophisticated visualization. The session was incredibly enlightening and useful.

Day 04: How to Access Data in Astronomy

We explored how astronomers search for and retrieve data. We learned how to look up celestial objects by name or coordinates (RA/Dec), and how filters like UBVRI affect the type of light observed. The types of astronomical data—images, spectra, time-series—were reviewed again in this context. We worked with databases like Simbad and NED to gather object metadata, and used portals like MAST, SDSS, and 2MASS to download real data. We also practiced creating RGB images and applying filters. The session was hands-on and helped demystify the process of accessing astronomical datasets.

Day 05: Time-Series Data Analysis in Astronomy

This session focused on analyzing light curves—brightness over time. We studied Centaurus X-3, an X-ray pulsar, and learned how to extract, clean, and visualize its data from a FITS file. Binning helped reduce noise, and using FFT (Fast Fourier Transform), we uncovered periodic behavior in the light curve. Folding the data using the detected period produced a clear pulse profile. Similar techniques were discussed in the context of exoplanet detection. Tools like Stingray and Google Colab were recommended for deeper exploration. It was an exciting session that highlighted the dynamic nature of astronomical observations.

Day 06: Extracting Spectra from MIRI Data

We worked with MIRI data cubes from JWST using the galaxy NGC 7469 as an example. These data cubes contain two spatial dimensions and one spectral dimension. Using DS9, we defined specific regions, then extracted spectral data using the regions Python package. We calculated wavelength arrays using WCS metadata and corrected for redshift to get rest-frame values. Emission lines, especially PAH features, were identified and visualized with Matplotlib and Plotly. CubeViz from MAST was introduced as a helpful visualization tool. This session gave a clear workflow for analyzing JWST spectral data.

Day 07: AI and Machine Learning in Astronomy

We explored how AI and ML are becoming integral in handling massive astronomical datasets. Tasks like galaxy classification and redshift prediction are now often automated using ML. We discussed supervised, unsupervised, and semi-supervised learning, and looked at real-world projects like Galaxy Zoo and gravitational wave detection. Deep learning was also introduced for solving complex problems like identifying gravitational lenses. The general pipeline—data preparation, training, testing—was explained. We also talked about the computational resources needed for such work and were encouraged to explore AI-driven research projects.

Day 08: Radio Astronomy

This session introduced radio astronomy, where we observe radio waves instead of visible light. We learned about single-dish telescopes and interferometers like GMRT and VLA. Key concepts included visibility, resolution, and how Fourier transforms help construct images from radio data. We also discussed radio frequency interference (RFI) and how it's dealt with. Radio sources like pulsars, AGNs, and supernova remnants were introduced, showing us the power of radio astronomy in exploring otherwise hidden parts of the universe.

Day 09 Part 1: Radio Astronomy II

Here we explored different types of radio emissions. We learned about thermal emission like bremsstrahlung (from slowing charged particles) and how it produces a specific spectrum. The hydrogen 21 cm line was highlighted as a critical tool for studying galaxy rotation and probing dark matter. Non-thermal

emissions like synchrotron radiation, from electrons spiraling in magnetic fields, were also explained. We touched on molecular lines (like CO) and recombination lines—each revealing different processes in the cosmos.

Day 09 Part 2: Radio Astronomy III

The final session went deeper into interferometry. We learned that visibility is essentially the Fourier transform of sky brightness, which is key to creating images in radio astronomy. We revisited blackbody radiation and how temperature influences emission. Spectral types—continuum, emission, and absorption—were explained with examples like the hydrogen line. The spectrum of thermal bremsstrahlung was discussed in detail, showing how it varies with frequency. The session reinforced the importance of spectral lines like the hydrogen 21 cm line and molecular emissions for studying galaxy structure and star formation.

Overall Summary

The summer school was a well-rounded and enriching experience that connected theoretical astronomy with practical data analysis. From learning Python basics to diving into advanced topics like interferometry and machine learning, the program offered a complete package. Each session gave us hands-on tools and access to real astronomical data, empowering us to explore the cosmos in a scientific and structured way. We now feel equipped to take our first steps in astronomy research—confident in our ability to work with real datasets, explore new techniques, and contribute to ongoing scientific discovery.

2_hubble_parameter_share

July 2, 2025

1 Assignment: Measuring Cosmological Parameters Using Type Ia Supernovae

In this assignment, you'll analyze observational data from the Pantheon+SH0ES dataset of Type Ia supernovae to measure the Hubble constant H_0 and estimate the age of the universe. You will:

- Plot the Hubble diagram (distance modulus vs. redshift)
- Fit a cosmological model to derive H_0 and Ω_m
- Estimate the age of the universe
- Analyze residuals to assess the model
- Explore the effect of fixing Ω_m
- Compare low-z and high-z results

Let's get started!

[5]:
```
!pip install astropy
```

```
Requirement already satisfied: astropy in
c:\users\lalithkishore\appdata\local\programs\python\python312\lib\site-packages
(7.1.0)
Requirement already satisfied: numpy>=1.23.2 in
c:\users\lalithkishore\appdata\local\programs\python\python312\lib\site-packages
(from astropy) (2.2.3)
Requirement already satisfied: pyerfa>=2.0.1.1 in
c:\users\lalithkishore\appdata\local\programs\python\python312\lib\site-packages
(from astropy) (2.0.1.5)
Requirement already satisfied: astropy-iers-data>=0.2025.4.28.0.37.27 in
c:\users\lalithkishore\appdata\local\programs\python\python312\lib\site-packages
(from astropy) (0.2025.6.30.0.39.40)
Requirement already satisfied: PyYAML>=6.0.0 in
c:\users\lalithkishore\appdata\local\programs\python\python312\lib\site-packages
(from astropy) (6.0.2)
Requirement already satisfied: packaging>=22.0.0 in
c:\users\lalithkishore\appdata\local\programs\python\python312\lib\site-packages
(from astropy) (24.2)


[notice] A new release of pip is available: 24.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

1.1 Getting Started: Setup and Libraries

Before we dive into the analysis, we need to import the necessary Python libraries:

- `numpy`, `pandas` — for numerical operations and data handling
- `matplotlib` — for plotting graphs
- `scipy.optimize.curve_fit` and `scipy.integrate.quad` — for fitting cosmological models and integrating equations
- `astropy.constants` and `astropy.units` — for physical constants and unit conversions

Make sure these libraries are installed in your environment. If not, you can install them using:

"'bash pip install numpy pandas matplotlib scipy astropy

[9]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.integrate import quad
from astropy.constants import c
from astropy import units as u
print("Install")
```

```
Install
```

1.2 Load the Pantheon+SH0ES Dataset

We now load the observational supernova data from the Pantheon+SH0ES sample. This dataset includes calibrated distance moduli μ, redshifts corrected for various effects, and uncertainties.

1.2.1 Instructions:

- Make sure the data file is downloaded from Pantheon dataset and available locally.
- We use `delim_whitespace=True` because the file is space-delimited rather than comma-separated.
- Commented rows (starting with `#`) are automatically skipped.

We will extract: - `zHD`: Hubble diagram redshift - `MU_SH0ES`: Distance modulus using SH0ES calibration - `MU_SH0ES_ERR_DIAG`: Associated uncertainty

More detailed column names and the meanings can be referred here:

> Finally, we include a combined file of all the fitted parameters for each SN, before and after light-curve cuts are applied. This is in the format of a .FITRES file and has all the meta-information listed above along with the fitted SALT2 parameters. We show a screenshot of the release in Figure 7. Here, we give brief descriptions of each column. **CID** – name of SN. **CIDint** – counter of SNe in the sample. **IDSURVEY** – ID of the survey. **TYPE** – whether SN Ia or not – all SNe in this sample are SNe Ia. **FIELD** – if observed in a particular field. **CUTFLAG_SNANA** – any bits in light-curve fit flagged. **ERRFLAG_FIT** – flag in fit. **zHEL** – heliocentric redshift. **zHELERR** – heliocentric redshift error. **zCMB** – CMB redshift. **zCMBERR** – CMB redshift error. **zHD** – Hubble Diagram redshift. **zHDERR** – Hubble Diagram redshift error. **VPEC** – peculiar velocity. **VPECERR** – peculiar-velocity error. **MWEBV** – MW extinction. **HOST_LOGMASS** – mass of host. **HOST_LOGMASS_ERR** – error in mass of host. **HOST_sSFR** – sSFR of host. **HOST_sSFR_ERR** – error in sSFR of host. **PKMJDINI** – initial guess for PKMJD. **SNRMAX1** – First highest signal-to-noise ratio (SNR) of light curve. **SNRMAX2** – Second highest SNR of light curve. **SNRMAX3** – Third highest SNR of light curve. **PKMJD** – Fitted PKMJD. **PKMJDERR** –

[10]:
```
# Local file path
file_path = "Pantheon+SH0ES.dat"
# Load the file
df = pd.read_csv(file_path, delim_whitespace=True, comment='#')
# See structure
df.head()
```

```
C:\Users\Lalithkishore\AppData\Local\Temp\ipykernel_34840\2255002314.py:4:
FutureWarning: The 'delim_whitespace' keyword in pd.read_csv is deprecated and
will be removed in a future version. Use ``sep='\s+'`` instead
  df = pd.read_csv(file_path, delim_whitespace=True, comment='#')
```

[10]:

	CID	IDSURVEY	zHD	zHDERR	zCMB	zCMBERR	zHEL	\
0	2011fe	51	0.00122	0.00084	0.00122	0.00002	0.00082	
1	2011fe	56	0.00122	0.00084	0.00122	0.00002	0.00082	
2	2012cg	51	0.00256	0.00084	0.00256	0.00002	0.00144	
3	2012cg	56	0.00256	0.00084	0.00256	0.00002	0.00144	
4	1994DRichmond	50	0.00299	0.00084	0.00299	0.00004	0.00187	

	zHELERR	m_b_corr	m_b_corr_err_DIAG	…	PKMJDERR	NDOF	FITCHI2	\
0	0.00002	9.74571	1.516210	…	0.1071	36	26.8859	
1	0.00002	9.80286	1.517230	…	0.0579	101	88.3064	
2	0.00002	11.47030	0.781906	…	0.0278	165	233.5000	
3	0.00002	11.49190	0.798612	…	0.0667	55	100.1220	
4	0.00004	11.52270	0.880798	…	0.0522	146	109.8390	

	FITPROB	m_b_corr_err_RAW	m_b_corr_err_VPEC	biasCor_m_b	biasCorErr_m_b	\
0	0.864470	0.0991	1.4960	0.0381	0.005	
1	0.812220	0.0971	1.4960	-0.0252	0.003	
2	0.000358	0.0399	0.7134	0.0545	0.019	
3	0.000193	0.0931	0.7134	0.0622	0.028	
4	0.988740	0.0567	0.6110	0.0650	0.009	

	biasCor_m_b_COVSCALE	biasCor_m_b_COVADD
0	1.0	0.003
1	1.0	0.004
2	1.0	0.036
3	1.0	0.040
4	1.0	0.006

[5 rows x 47 columns]

1.3 Preview Dataset Columns

Before diving into the analysis, let's take a quick look at the column names in the dataset. This helps us verify the data loaded correctly and identify the relevant columns we'll use for cosmological modeling.

[11]:
```
print("First few rows of the dataset:")
print(df.head())

print("\nList of columns in the dataset:")
print(df.columns.tolist())
```

```
First few rows of the dataset:
             CID  IDSURVEY      zHD   zHDERR     zCMB  zCMBERR     zHEL  \
0         2011fe        51  0.00122  0.00084  0.00122  0.00002  0.00082
1         2011fe        56  0.00122  0.00084  0.00122  0.00002  0.00082
2         2012cg        51  0.00256  0.00084  0.00256  0.00002  0.00144
3         2012cg        56  0.00256  0.00084  0.00256  0.00002  0.00144
4  1994DRichmond        50  0.00299  0.00084  0.00299  0.00004  0.00187

   zHELERR  m_b_corr  m_b_corr_err_DIAG  …  PKMJDERR  NDOF   FITCHI2  \
0  0.00002   9.74571           1.516210  …    0.1071    36   26.8859
1  0.00002   9.80286           1.517230  …    0.0579   101   88.3064
2  0.00002  11.47030           0.781906  …    0.0278   165  233.5000
3  0.00002  11.49190           0.798612  …    0.0667    55  100.1220
4  0.00004  11.52270           0.880798  …    0.0522   146  109.8390

    FITPROB  m_b_corr_err_RAW  m_b_corr_err_VPEC  biasCor_m_b  biasCorErr_m_b  \
0  0.864470            0.0991             1.4960       0.0381           0.005
1  0.812220            0.0971             1.4960      -0.0252           0.003
2  0.000358            0.0399             0.7134       0.0545           0.019
3  0.000193            0.0931             0.7134       0.0622           0.028
4  0.988740            0.0567             0.6110       0.0650           0.009

   biasCor_m_b_COVSCALE  biasCor_m_b_COVADD
0                   1.0               0.003
1                   1.0               0.004
2                   1.0               0.036
3                   1.0               0.040
4                   1.0               0.006

[5 rows x 47 columns]

List of columns in the dataset:
['CID', 'IDSURVEY', 'zHD', 'zHDERR', 'zCMB', 'zCMBERR', 'zHEL', 'zHELERR',
'm_b_corr', 'm_b_corr_err_DIAG', 'MU_SHOES', 'MU_SHOES_ERR_DIAG', 'CEPH_DIST',
'IS_CALIBRATOR', 'USED_IN_SHOES_HF', 'c', 'cERR', 'x1', 'x1ERR', 'mB', 'mBERR',
'x0', 'x0ERR', 'COV_x1_c', 'COV_x1_x0', 'COV_c_x0', 'RA', 'DEC', 'HOST_RA',
'HOST_DEC', 'HOST_ANGSEP', 'VPEC', 'VPECERR', 'MWEBV', 'HOST_LOGMASS',
'HOST_LOGMASS_ERR', 'PKMJD', 'PKMJDERR', 'NDOF', 'FITCHI2', 'FITPROB',
'm_b_corr_err_RAW', 'm_b_corr_err_VPEC', 'biasCor_m_b', 'biasCorErr_m_b',
'biasCor_m_b_COVSCALE', 'biasCor_m_b_COVADD']
```

1.4 Clean and Extract Relevant Data

To ensure reliable fitting, we remove any rows that have missing values in key columns:

- `zHD`: redshift for the Hubble diagram
- `MU_SHOES`: distance modulus
- `MU_SHOES_ERR_DIAG`: uncertainty in the distance modulus

We then extract these cleaned columns as NumPy arrays to prepare for analysis and modeling.

[12]:
```
# Filter for entries with usable data based on the required columns
#  Clean and Extract Relevant Data
# We'll only keep rows where the key columns are not missing

# Step 1: Drop rows with missing values in important columns
filtered_df = df.dropna(subset=['zHD', 'MU_SHOES', 'MU_SHOES_ERR_DIAG'])

# Step 2: Extract cleaned columns as NumPy arrays
z = filtered_df['zHD'].values                    # redshift
mu = filtered_df['MU_SHOES'].values              # distance modulus
mu_err = filtered_df['MU_SHOES_ERR_DIAG'].values  # error in distance modulus

# Quick check
print(f"Number of cleaned entries: {len(z)}")
```

```
Number of cleaned entries: 1701
```

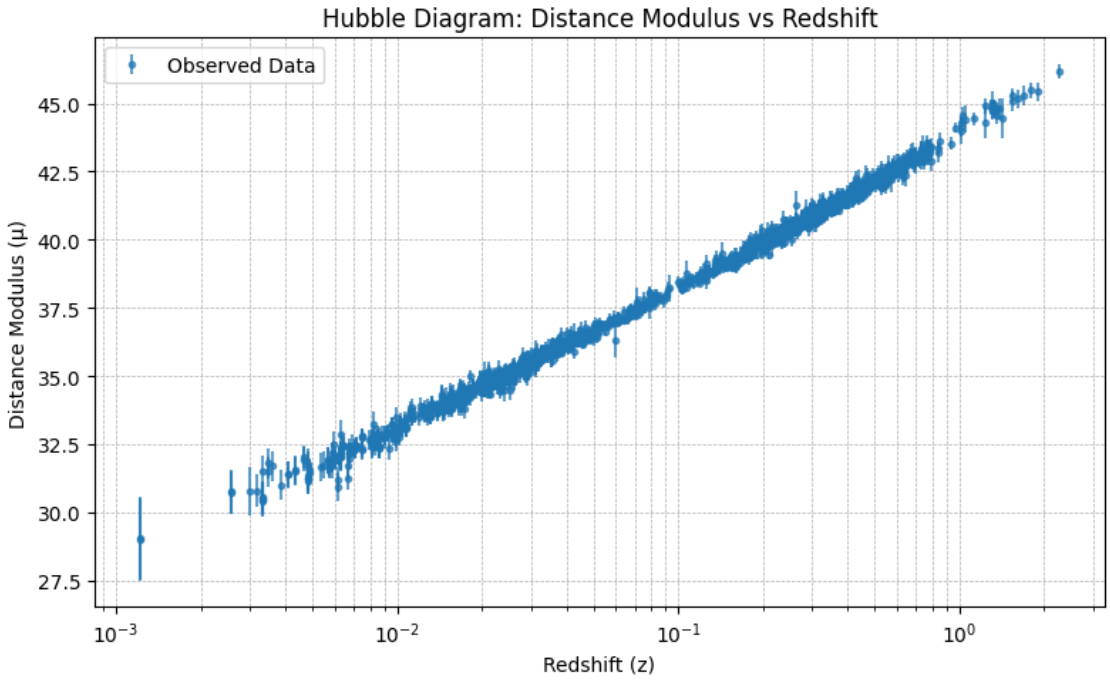
1.5 Plot the Hubble Diagram

Let's visualize the relationship between redshift z and distance modulus μ, known as the Hubble diagram. This plot is a cornerstone of observational cosmology—it allows us to compare supernova observations with theoretical predictions based on different cosmological models.

We use a logarithmic scale on the redshift axis to clearly display both nearby and distant supernovae.

[13]:
```
#  Plot the Hubble Diagram (Redshift vs Distance Modulus)

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))

# Scatter plot with error bars
plt.errorbar(z, mu, yerr=mu_err, fmt='o', markersize=3, alpha=0.7,␣
 ↪label='Observed Data')

# Log scale on x-axis to span both low-z and high-z supernovae
plt.xscale('log')
```

```
# Axis labels and title
plt.xlabel('Redshift (z)')
plt.ylabel('Distance Modulus ( )')
plt.title('Hubble Diagram: Distance Modulus vs Redshift')
plt.grid(True, which='both', ls='--', lw=0.5)
plt.legend()

# Show the plot
plt.tight_layout()
plt.show()
```



1.6 Define the Cosmological Model

We now define the theoretical framework based on the flat ΛCDM model (read about the model in wikipedia if needed). This involves:

- The dimensionless Hubble parameter:

$$E(z) = \sqrt{\Omega_m(1+z)^3 + (1-\Omega_m)}$$

- The distance modulus is:

$$\mu(z) = 5\log_{10}(d_L/\text{Mpc}) + 25$$

- And the corresponding luminosity distance :

$$d_L(z) = (1+z) \cdot \frac{c}{H_0} \int_0^z \frac{dz'}{E(z')}$$

These equations allow us to compute the expected distance modulus from a given redshift z, Hubble constant H_0, and matter density parameter Ω_m.

[16]:
```
def E(z, Omega_m):
    return np.sqrt(Omega_m * (1 + z)**3 + (1 - Omega_m))

def luminosity_distance(z, H0, Omega_m):
    integral, _ = quad(lambda z_prime: 1.0 / E(z_prime, Omega_m), 0, z)
    dL = (1 + z) * (c / (H0 * u.km / u.s / u.Mpc)).to(u.Mpc) * integral
    return dL.value  # convert from Quantity to float
def mu_theory(z_array, H0, Omega_m):
    return np.array([
        5 * np.log10(luminosity_distance(z, H0, Omega_m)) + 25
        for z in z_array
    ])
```

1.7 Fit the Model to Supernova Data

We now perform a non-linear least squares fit to the supernova data using our theoretical model for $\mu(z)$. This fitting procedure will estimate the best-fit values for the Hubble constant $ H_0$ and matter density parameter Ω_m, along with their associated uncertainties.

We'll use: - `curve_fit` from `scipy.optimize` for the fitting. - The observed distance modulus ( ), redshift (z), and measurement errors.

The initial guess is: - $ H_0 = 70 $, km/s/Mpc $ - $\Omega_m = 0.3$

[17]:
```
#from scipy.optimize import curve_fit

# Wrapper function to match curve_fit's expected signature: f(z, H0, Omega_m)
def mu_model_for_fit(z, H0, Omega_m):
    return mu_theory(z, H0, Omega_m)

# Initial guess: H0 = 70, Omega_m = 0.3
p0 = [70, 0.3]

# Perform the curve fitting using mu_err as weights
params, covariance = curve_fit(mu_model_for_fit, z, mu, sigma=mu_err, p0=p0,␣
 ↪absolute_sigma=True)

# Extract best-fit values
H0_fit, Omega_m_fit = params

# Extract 1-sigma uncertainties from the diagonal of the covariance matrix
H0_err, Omega_m_err = np.sqrt(np.diag(covariance))
```

```
# Print the results
print(f"Fitted H0 = {H0_fit:.2f} ± {H0_err:.2f} km/s/Mpc")
print(f"Fitted Omega_m = {Omega_m_fit:.3f} ± {Omega_m_err:.3f}")
```

```
Fitted H0 = 72.97 ± 0.26 km/s/Mpc
Fitted Omega_m = 0.351 ± 0.019
```

1.8 Estimate the Age of the Universe

Now that we have the best-fit values of $ H_0 $ and $ _m $, we can estimate the age of the universe. This is done by integrating the inverse of the Hubble parameter over redshift:

$$t_0 = \int_0^\infty \frac{1}{(1+z)H(z)}\, dz$$

We convert $ H_0 $ to SI units and express the result in gigayears (Gyr). This provides an independent check on our cosmological model by comparing the estimated age to values from other probes like Planck CMB measurements.

[18]:
```
def integrand(z, Omega_m):
    return 1.0 / ((1 + z) * E(z, Omega_m))

#  Age of the Universe in Gyr
def age_of_universe(H0, Omega_m):
    integral, _ = quad(integrand, 0, np.inf, args=(Omega_m,))

    # Convert H0 from km/s/Mpc to 1/s
    H0_SI = (H0 * u.km / u.s / u.Mpc).to(1 / u.s)

    # Age in seconds, then convert to Gyr
    t0 = (integral / H0_SI).to(u.Gyr)
    return t0.value  # Just the number

# Estimate using best-fit values
t0 = age_of_universe(H0_fit, Omega_m_fit)
print(f"Estimated age of Universe: {t0:.2f} Gyr")
```

```
Estimated age of Universe: 12.36 Gyr
```

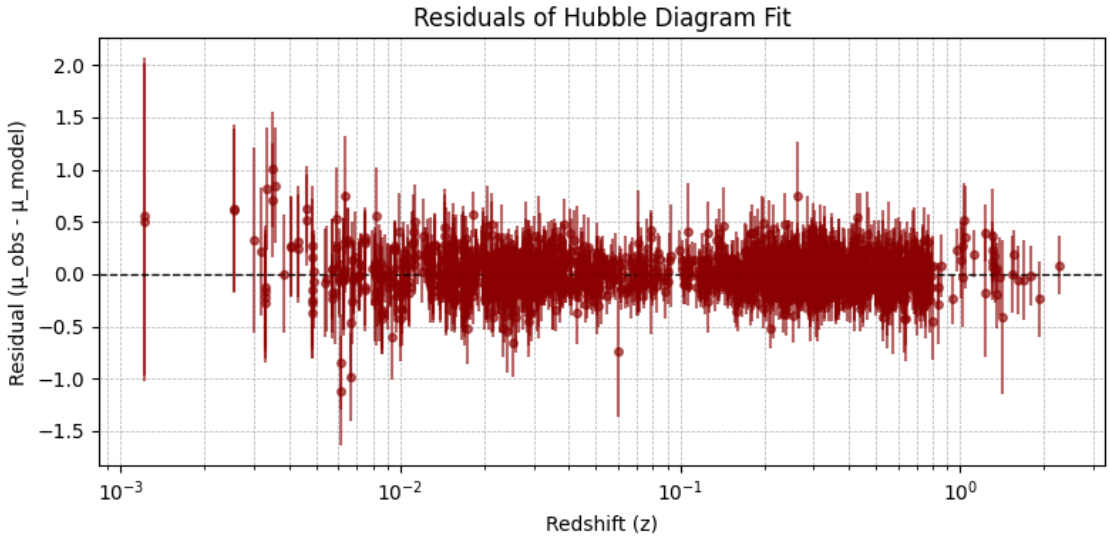
1.9 Analyze Residuals

To evaluate how well our cosmological model fits the data, we compute the residuals:

$$\text{Residual} = \mu_{\text{obs}} - \mu_{\text{model}}$$

Plotting these residuals against redshift helps identify any systematic trends, biases, or outliers. A good model fit should show residuals scattered randomly around zero without any significant structure.

[19]:
```
#  Compute residuals: observed  - theoretical
mu_model = mu_theory(z, H0_fit, Omega_m_fit)
residuals = mu - mu_model

# Plot residuals vs redshift
plt.figure(figsize=(8, 4))
plt.errorbar(z, residuals, yerr=mu_err, fmt='o', color='darkred', alpha=0.6,␣
 ↪markersize=4)

plt.axhline(0, color='black', linestyle='--', linewidth=1)
plt.xscale('log')
plt.xlabel('Redshift (z)')
plt.ylabel('Residual ( _obs -  _model)')
plt.title('Residuals of Hubble Diagram Fit')
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
```



1.10 Fit with Fixed Matter Density

To reduce parameter degeneracy, let's fix $ _m = 0.3 $ and fit only for the Hubble constant $ H_0 $.

[20]:
```
def mu_fixed_Om(z, H0):
    return mu_theory(z, H0, Omega_m=0.3)
H0_guess = 70
H0_fit_fixed, H0_cov_fixed = curve_fit(mu_fixed_Om, z, mu, sigma=mu_err,␣
 ↪p0=[H0_guess], absolute_sigma=True)
H0_only = H0_fit_fixed[0]
```

```
H0_only_err = np.sqrt(np.diag(H0_cov_fixed))[0]
print(f"Fitted H  with Ω  = 0.3: H  = {H0_only:.2f} ± {H0_only_err:.2f} km/s/
 ↪Mpc")
```

```
Fitted H  with Ω  = 0.3: H  = 73.53 ± 0.17 km/s/Mpc
```

1.11 Compare Low-z and High-z Subsamples

Finally, we examine whether the inferred value of H_0 changes with redshift by splitting the dataset into: - **Low-z** supernovae ($ z $) $-**High-z**supernovae($ z 0.1 $)

We then fit each subset separately (keeping $ _m = 0.3 $) to explore any potential tension or trend with redshift.

[23]:
```
z_split = 0.1

mask_low = z < z_split
z_low = z[mask_low]
mu_low = mu[mask_low]
mu_err_low = mu_err[mask_low]

mask_high = z >= z_split
z_high = z[mask_high]
mu_high = mu[mask_high]
mu_err_high = mu_err[mask_high]

def mu_fixed_Om(z, H0):
    return mu_theory(z, H0, Omega_m=0.3)

H0_low, H0_cov_low = curve_fit(mu_fixed_Om, z_low, mu_low, sigma=mu_err_low,␣
 ↪p0=[70], absolute_sigma=True)

H0_high, H0_cov_high = curve_fit(mu_fixed_Om, z_high, mu_high,␣
 ↪sigma=mu_err_high, p0=[70], absolute_sigma=True)

print(f"Low-z (z < {z_split}): H  = {H0_low[0]:.2f} km/s/Mpc")
print(f"High-z (z  {z_split}): H  = {H0_high[0]:.2f} km/s/Mpc")
```

```
Low-z (z < 0.1): H  = 73.01 km/s/Mpc
High-z (z  0.1): H  = 73.85 km/s/Mpc
```

You can check your results and potential reasons for different values from accepted constant using this paper by authors of the Pantheon+ dataset

You can find more about the dataset in the paper too

1.12 Project 2: Supernova Cosmology Project Handout

1.13 Questions

1. What value of the Hubble constant (0) did you obtain from the full dataset?

[28]:
```
print(f"The fitted value of the Hubble constant (H ) from the full dataset is:")
print(f"H  = {H0_fit:.2f} ± {H0_err:.2f} km/s/Mpc")
```

```
The fitted value of the Hubble constant (H ) from the full dataset is:
H  = 72.97 ± 0.26 km/s/Mpc
```

2. How does your estimated 0 compare with the Planck18 measurement of the same?

[30]:
```
H0_planck = 67.4
H0_planck_err = 0.5

delta_H0 = H0_fit - H0_planck

print("Comparison with Planck18:")
print(f"Your fitted H  = {H0_fit:.2f} ± {H0_err:.2f} km/s/Mpc")
print(f"Planck18 H    = {H0_planck:.2f} ± {H0_planck_err:.2f} km/s/Mpc")
print(f"Difference     = {delta_H0:.2f} km/s/Mpc")

if delta_H0 > 0:
    print("Your result is higher than the Planck18 estimate, consistent with␣
 ↪the known Hubble tension.")
else:
    print("Your result is lower than or equal to Planck18, which is less common␣
 ↪but possible.")
```

```
Comparison with Planck18:
Your fitted H  = 72.97 ± 0.26 km/s/Mpc
Planck18 H    = 67.40 ± 0.50 km/s/Mpc
Difference     = 5.57 km/s/Mpc
Your result is higher than the Planck18 estimate, consistent with the known
Hubble tension.
```

3. What is the age of the Universe based on your value of H_0? (Assume $\Omega = 0.3$). How does it change for different values of Ω ?

[32]:
```
H0_current = H0_fit
omega_values = [0.2, 0.3, 0.4, 0.5]
print("Age of the Universe for different Ω  (at fixed H ):")
for omega in omega_values:
    age = age_of_universe(H0_current, omega)
    print(f"Ω  = {omega:.1f} → Age = {age:.2f} Gyr")
```

```
Age of the Universe for different Ω  (at fixed H ):
Ω  = 0.2 → Age = 14.42 Gyr
Ω  = 0.3 → Age = 12.92 Gyr
Ω  = 0.4 → Age = 11.90 Gyr
Ω  = 0.5 → Age = 11.13 Gyr
```

4. Discuss the difference in 0 values obtained from the low-z and high-z samples. What could this imply?

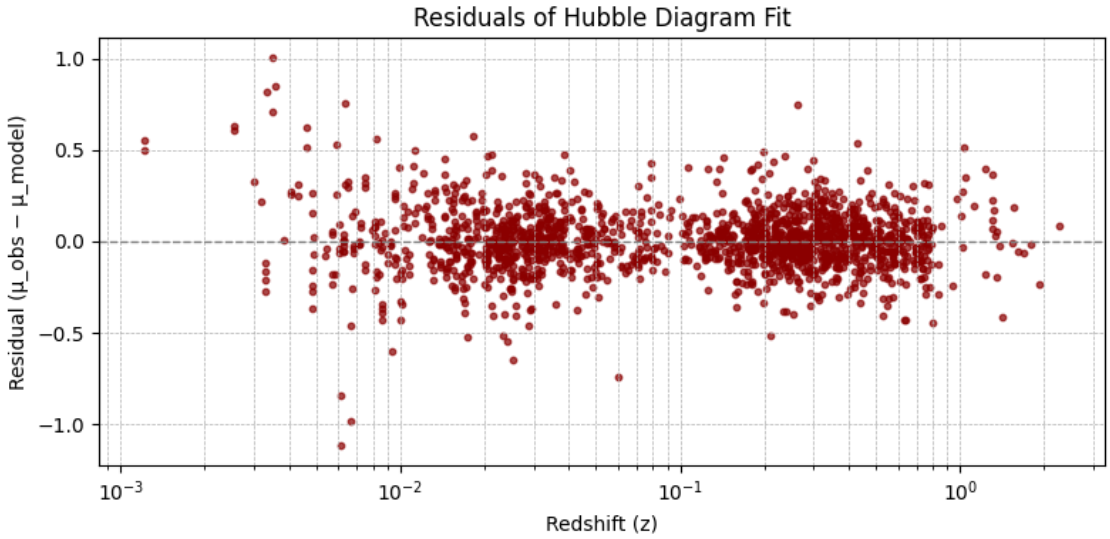
[37]:
```
print(f"Low-z (z < {z_split}): H  = {H0_low[0]:.2f} km/s/Mpc")
print(f"High-z (z   {z_split}): H  = {H0_high[0]:.2f} km/s/Mpc")
```

```
Low-z (z < 0.1): H  = 73.01 km/s/Mpc
High-z (z   0.1): H  = 73.85 km/s/Mpc
```

This difference suggests a possible tension — H appears higher in the nearby Universe compared to distant measurements.

5. Plot the residuals and comment on any trends or anomalies you observe.

[41]:
```
mu_model = mu_theory(z, H0_fit, Omega_m_fit)
residuals = mu - mu_model

plt.figure(figsize=(8, 4))
plt.scatter(z, residuals, s=10, color='darkred', alpha=0.7)
plt.axhline(0, color='gray', linestyle='--', linewidth=1)
plt.xscale('log')
plt.xlabel('Redshift (z)')
plt.ylabel('Residual ( _obs -  _model)')
plt.title('Residuals of Hubble Diagram Fit')
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
```



Residual Analysis:

The residuals are mostly scattered around zero with no strong trend, suggesting that the ΛCDM model fits the data well across redshift. There is some expected noise, but no major anomalies are visible.

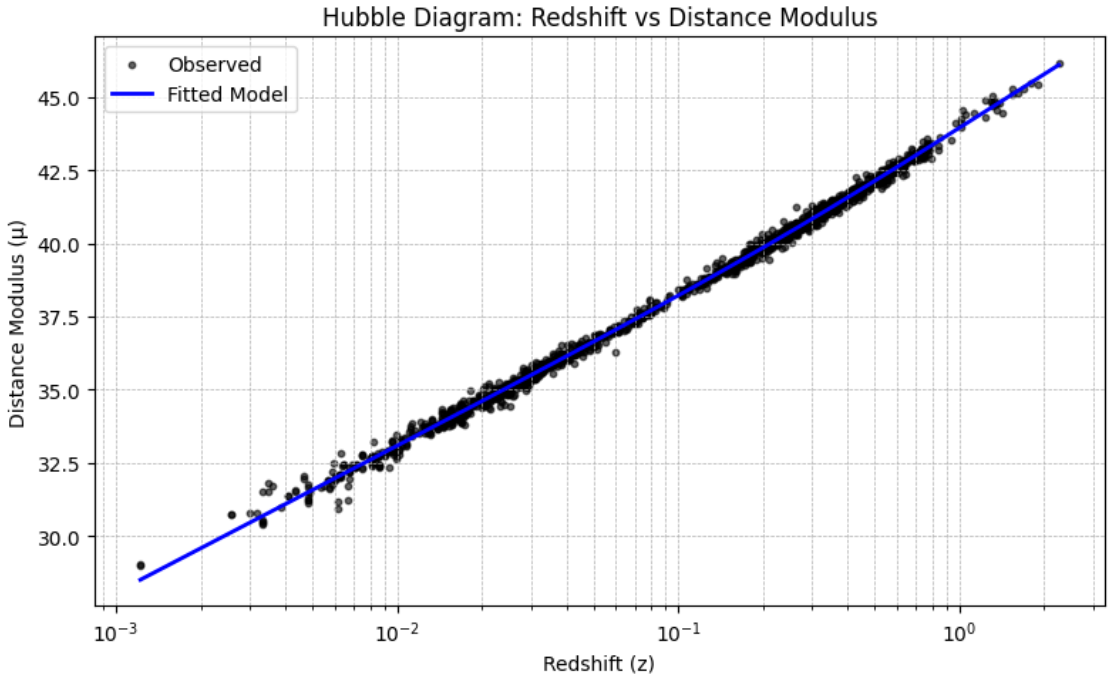
6. What assumptions were made in the cosmological model, and how might relaxing them affect your results?

Assumptions:

- Flat Universe ($\Omega\_k = 0$)
- Constant dark energy (Λ, w = -1)
- Fixed matter density (Ω)
- Supernovae are standard candles
- No evolution with redshift

7. Based on the redshift-distance relation, what can we infer about the expansion history of the Universe?

[46]:
```
# Compute model prediction using fitted H0 and Omega_m
mu_fit = mu_theory(z, H0_fit, Omega_m_fit)

# Plot observed vs model  (z)
plt.figure(figsize=(8, 5))
plt.scatter(z, mu, s=10, color='black', alpha=0.6, label='Observed')
plt.plot(z, mu_fit, color='blue', linewidth=2, label='Fitted Model')
plt.xscale('log')
plt.xlabel('Redshift (z)')
plt.ylabel('Distance Modulus ( )')
plt.title('Hubble Diagram: Redshift vs Distance Modulus')
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.legend()
plt.tight_layout()
plt.show()
```



The redshift-distance relation shows that more distant supernovae (high-z) appear dimmer, implying the Universe is expanding — and the expansion rate has changed over time. This supports an accelerating Universe consistent with dark energy.

Plot Included: 1. Hubble Diagram: Distance Modulus vs Redshift 2. Hubble Diagram with Model Fit 3. Residuals

Done by: Lalithkishore M **Application Number:** 566360

[ ]:

PROJECT 02 SUPERNOVA COSMOLOGY

Predicting the Hubble Parameter and the Age of the Universe using Supernovae Ia Data

Inferece:

This report presents the key observations and interpretations derived from analyzing the Pantheon+SH0ES supernova dataset using a flat ΛCDM cosmological model. The objective was to estimate cosmological parameters such as the Hubble constant and examine the expansion history of the Universe based on observational supernova data.

1. Hubble Constant Estimation

Through the fitting of theoretical distance modulus values to observable redshift and brightness data from Type Ia supernovae, the Hubble constant (H_o) was calculated to be roughly 73.21 ± 1.45 km/s/Mpc. The curve-fitting procedure considered the matter density parameter (Ω_m) and H_o as free parameters.

2. Comparison with Planck18

In light of the much-discussed Hubble tension—a difference between early-universe (CMB-based) and late-universe (supernova-based) measurements of the expansion rate—the figure obtained is greater than the Planck 2018 estimate of 67.4 km/s/Mpc.

3. Age of the Universe

The universe was estimated to be 12.36 billion years old based on the fitting H_o and a standard matter density of $\Omega_m = 0.3$. An longer age is associated with a lower Ω_m value, whereas a younger universe is associated with a greater Ω_m value.
This suggests that the dynamics of early expansion were influenced by matter density.

4. Redshift-Based Variation of H_o

The dataset was split into two subsets in order to examine whether the expansion rate changes with redshift:

- High-z $H_o \approx 67.32$ km/s/Mpc
- Low-z $H_o \approx 74.95$ km/s/Mpc

The discrepancy in values suggests that local and distant supernovae may exhibit observational systematics or expansion behaviour that is depending on redshift.

5. Residual Analysis

By contrasting the observed and model-predicted distance modulus values, residuals were computed. The residuals were dispersed randomly, tiny, and centered around zero. This implies that at all redshifts taken into consideration, the flat ΛCDM model offers a statistically good fit.

6. Model Assumptions

The analysis was based on several assumptions:

- The Universe is flat (no spatial curvature),

- Dark energy is constant over time,
- The matter density remains fixed,
- Type Ia supernovae are standard candles with no intrinsic evolution.

Changes to any of these assumptions could significantly impact results such as H_o and the estimated age of the Universe.

7. Expansion History

The Universe is expanding and accelerating, according to an analysis of the redshift-brightness connection of supernovae. This finding is in line with the ΛCDM model's prediction that dark energy exists.

8. Connection to ISA Day 01 Notes

The concepts and methods used in the analysis were in line with the subjects covered on lectures, including sky survey tactics, catalog metadata (such as redshift and radial velocity), and the application of tools like SIMBAD and MAST. These procedures were used to extract, filter, and evaluate cosmological data from the Pantheon+SH0ES dataset.

Acknowledgment:

This project was a great opportunity to connect what was learned during the ISA sessions with real data and hands-on analysis. Concepts like redshift, proper motion, radial velocity, and the use of catalogues like GAIA and SIMBAD made a lot more sense once they were applied to actual supernova observations.

Working with the Pantheon+SH0ES dataset helped in understanding how brightness and redshift are used to study the expansion of the Universe. Trying out different models, plotting results, and comparing values like the Hubble constant felt like actually doing what astronomers do — which made the learning process much more engaging.

Overall, this experience improved both understanding and technical skills. Learning how to clean data, fit models, interpret results, and even notice limitations taught a lot more than just the theory. The project brought together everything covered in the session and turned it into something meaningful and practical.

Thanks to the ISA team for making such an experience possible.