🔭 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!

📦 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:

```
pip install numpy pandas matplotlib scipy astropy
```

In [3]:
```python
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
```

📁 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.

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** –

In [16]:
```python
# Local file path

file_path = "Pantheon+SH0ES (1).dat"




# Load the file
df = pd.read_csv(file_path, delim_whitespace=True, comment='#')

# See structure
df.head(10)
```

```
C:\Users\chaka\AppData\Local\Temp\ipykernel_27920\1665632831.py:9: FutureWarning: 
The 'delim_whitespace' keyword in pd.read_csv is deprecated and will be removed i
n a future version. Use ``sep='\s+'`` instead
  df = pd.read_csv(file_path, delim_whitespace=True, comment='#')
```

Out[16]:

	CID	IDSURVEY	zHD	zHDERR	zCMB	zCMBERR	zHEL	zHELERR
0	2011fe	51	0.00122	0.00084	0.00122	0.00002	0.00082	0.00002
1	2011fe	56	0.00122	0.00084	0.00122	0.00002	0.00082	0.00002
2	2012cg	51	0.00256	0.00084	0.00256	0.00002	0.00144	0.00002
3	2012cg	56	0.00256	0.00084	0.00256	0.00002	0.00144	0.00002
4	1994DRichmond	50	0.00299	0.00084	0.00299	0.00004	0.00187	0.00004
5	1981B	50	0.00317	0.00084	0.00350	0.00001	0.00236	0.00001
6	2013aa	56	0.00331	0.00085	0.00478	0.00015	0.00411	0.00015
7	2013aa	5	0.00331	0.00085	0.00478	0.00015	0.00411	0.00015
8	2017cbv	5	0.00331	0.00085	0.00478	0.00015	0.00411	0.00015
9	2017cbv	18	0.00331	0.00085	0.00478	0.00015	0.00411	0.00015

10 rows × 47 columns

🧾 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.

In [ ]:

```

```

🧹 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_SH0ES` : distance modulus
- `MU_SH0ES_ERR_DIAG` : uncertainty in the distance modulus

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

In [26]:

```
# Filter for entries with usable data based on the required columns
df = df.dropna(subset=["zHD", "MU_SH0ES", "MU_SH0ES_ERR_DIAG"])
```

📈 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.

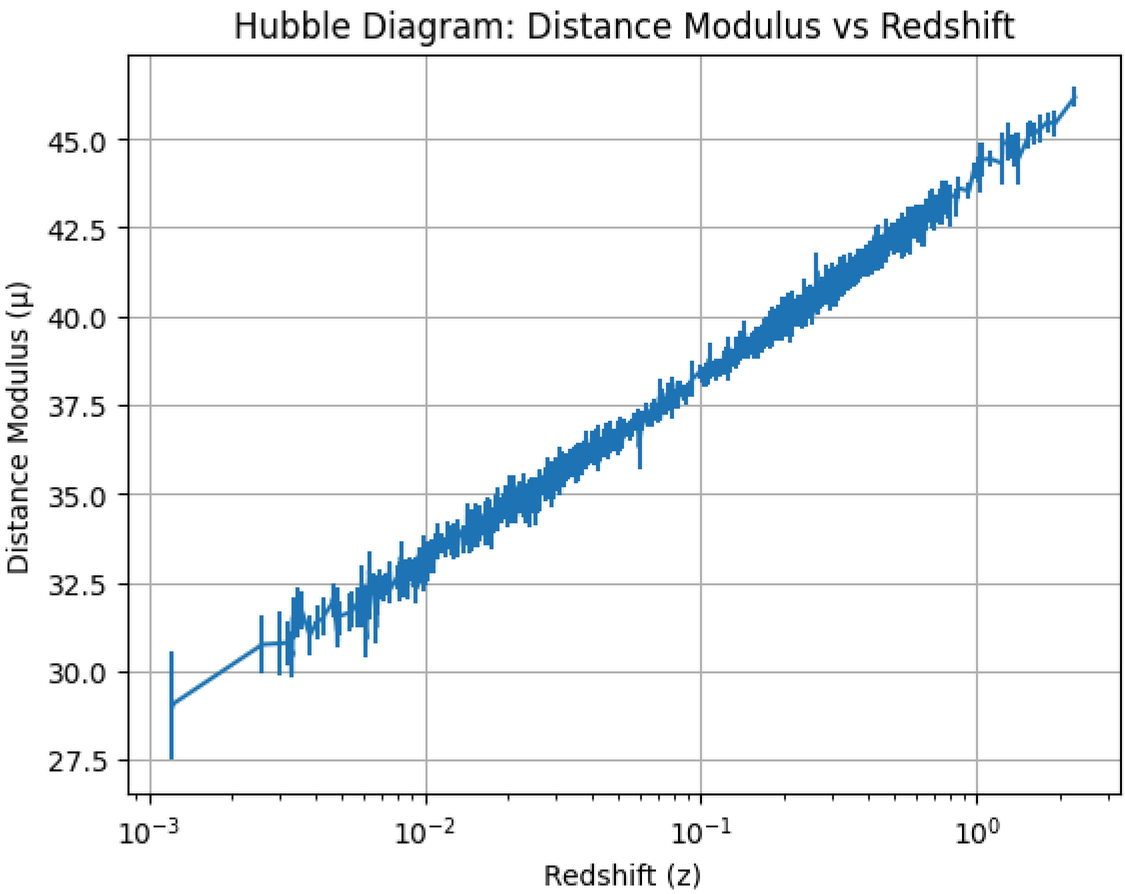
We use a logarithmic scale on the redshift axis to clearly display both nearby and distant supernovae.

In [36]:

```
# Write a code to plot the distance modulus and the redshift (x-axis), label the
import matplotlib.pyplot as plt


plt.errorbar(df["zHD"], df["MU_SH0ES"], yerr=df["MU_SH0ES_ERR_DIAG"])
plt.xscale("log")
plt.xlabel("Redshift (z)")
plt.ylabel("Distance Modulus (μ)")
plt.title("Hubble Diagram: Distance Modulus vs Redshift")
plt.grid(True)
plt.show()

#Try using log scale in x-axis
```



🧮 Define the Cosmological Model

We now define the theoretical framework based on the flat $\Lambda\mathrm{CDM}$ 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:

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

- 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.

In [45]:
```
# Define the E(z) for flat LCDM
def E(z, Omega_m):
    return np.sqrt(Omega_m * (1 + z)**3 + (1 - Omega_m))


# Luminosity distance in Mpc, try using scipy quad to integrate.
def luminosity_distance(z, H0, Omega_m):
    integral, _ = quad(lambda z_prime: 1 / E(z_prime, Omega_m), 0, z)
    H0 = H0 * (u.km / u.s / u.Mpc)  # Give H0 proper units
    H_D = (c / H0).to(u.Mpc).value  # Now this works!  # Hubble distance
    return (1 + z) * H_D * integral


# Theoretical distance modulus, use above function inside mu_theory to compute L
def mu_theory(z, H0, Omega_m):
    d_L = np.array([luminosity_distance(zi, H0, Omega_m) for zi in z])
    return 5 * np.log10(d_L) + 25
```

🔧 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 (\mu), redshift (z), and measurement errors.

The initial guess is:

- $H_0 = 70\,\mathrm{km/s/Mpc}$
- $\Omega_m = 0.3$

In [46]:
```
def mu_model(z, H0, Omega_m):
    return mu_theory(z, H0, Omega_m)

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

# Write a code for fitting and taking error out of the parameters
```

```python
# Fit the model to the data
popt, pcov = curve_fit(mu_model, df["zHD"], df["MU_SH0ES"], sigma=df["MU_SH0ES_E

# Extract fitted values and uncertainties
H0_fit, Omega_m_fit = popt
H0_err, Omega_m_err = np.sqrt(np.diag(pcov))

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 ± 0.17 km/s/Mpc
Fitted Omega_m = 0.351 ± 0.012
```

⌛ Estimate the Age of the Universe

Now that we have the best-fit values of H_0 and Ω_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.

In [48]:
```python
# Write the function for age of the universe as above

def age_of_universe(H0, Omega_m):
    integrand = lambda z: 1 / ((1 + z) * E(z, Omega_m))
    integral, _ = quad(integrand, 0, np.inf)

    H0_with_units = H0 * (u.km / u.s / u.Mpc)
    H_T= (1 / H0_with_units).to(u.Gyr).value  # Hubble time in Gyr

    return H_T * integral  # Age in Gyr
# Estimate age using your fitted parameters
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
```

📊 Analyze Residuals

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

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

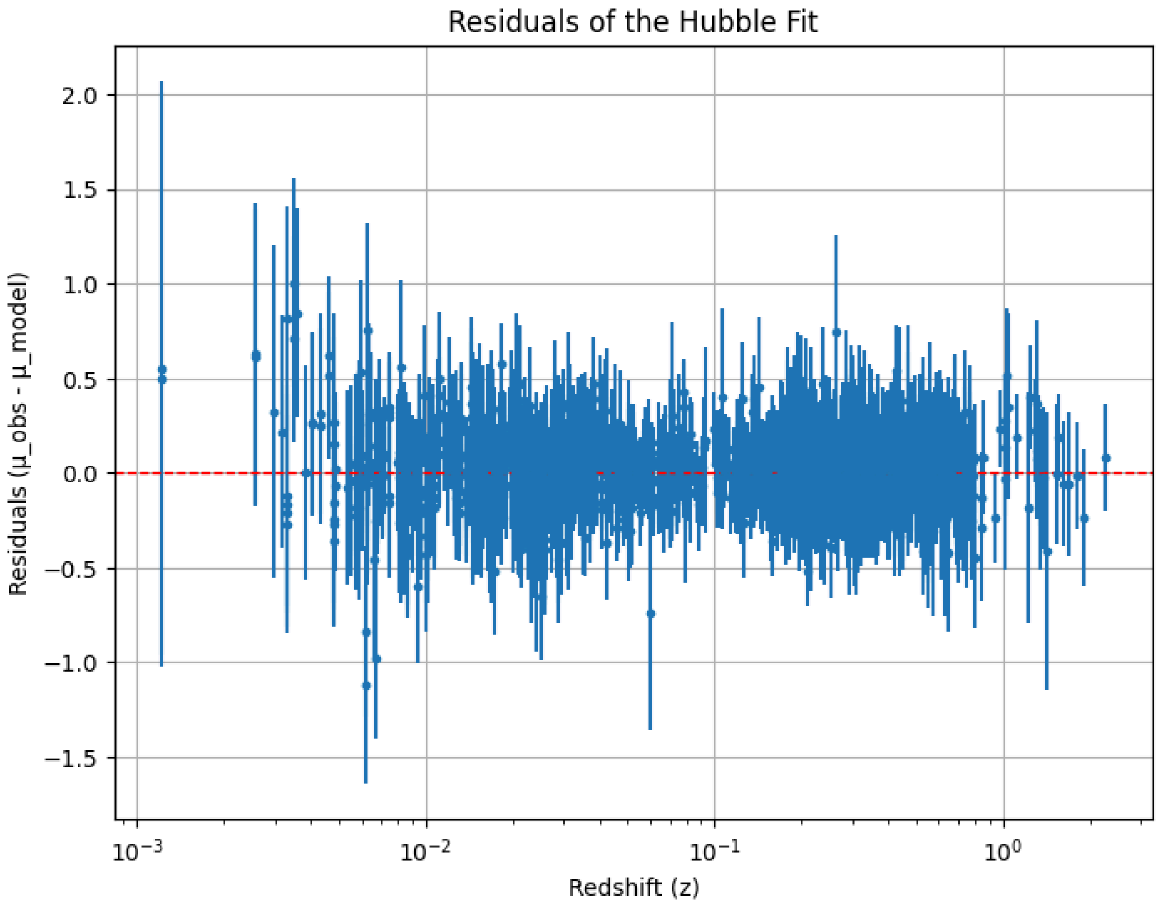
Plotting these residuals against redshift helps identify any systematic trends, biases, or outliers. A good model fit should show residuals scattered randomly around zero without any significant structure.

In [52]:
```python
# Calculate the predicted distance modulus using the best-fit H0 and Omega_m
mu_model = mu_theory(df["zHD"], H0_fit, Omega_m_fit)

# Calculate residuals (observed - model)
```

```
residuals = df["MU_SH0ES"] - mu_model

# Plot the residuals
plt.figure(figsize=(8,6))
plt.errorbar(df["zHD"], residuals, yerr=df["MU_SH0ES_ERR_DIAG"], fmt='o', marker
plt.axhline(0, color='red', linestyle='--', linewidth=1)
plt.xscale("log")
plt.xlabel("Redshift (z)")
plt.ylabel("Residuals (μ_obs - μ_model)")
plt.title("Residuals of the Hubble Fit")
plt.grid(True)
plt.show()
```



🔧 Fit with Fixed Matter Density

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

In [53]:
```
def mu_fixed_Om(z, H0):
    return mu_theory(z, H0, Omega_m=0.3)

# Try fitting with this fixed value
```

🔍 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 ($z < 0.1$)
- **High-z** supernovae ($z \geq 0.1$)

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

In [54]:
```
# Split the data for the three columns and do the fitting again and see
# Define the split value
z_split = 0.5

# Create two subsets
df_low = df[df["zHD"] < z_split]
df_high = df[df["zHD"] >= z_split]

# Fit H0 separately for both
H0_low, _ = curve_fit(mu_fixed_Om, df_low["zHD"], df_low["MU_SH0ES"], sigma=df_l
H0_high, _ = curve_fit(mu_fixed_Om, df_high["zHD"], df_high["MU_SH0ES"], sigma=d

print(f"Low-z (z < {z_split}): H₀ = {H0_low[0]:.2f} km/s/Mpc")
print(f"High-z (z ≥ {z_split}): H₀ = {H0_high[0]:.2f} km/s/Mpc")
```

```
Low-z (z < 0.5): H₀ = 73.39 km/s/Mpc
High-z (z ≥ 0.5): H₀ = 74.96 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