Cosmological Parameter Estimation using SuperNovae Type Ia Data

Quentin Le Ny, and Halim Perez Melendez

May 6, 2024

Abstract

In this project we will constrain cosmological data from **Type** Ia **Supernovae (SNe)**, which we will collect from the **Union 2.1 LBL Supernova Project** webpage (Suzuki et al. (2012)). The data will provide us with observed **redshift** and **apparent magnitude** values. We will theoretically estimate the apparent magnitude values beginning with the redshift data, and determine the **chi-squared** value given the difference between the calculated and observed values, with the help of a provided covariance matrix. This will allow us to estimate the cosmological parameters $\boldsymbol{\Omega_M}$ and $\boldsymbol{\Omega_{DE}}$ and their confidence intervals. $\boldsymbol{\Omega_M}$ is the matter density parameter in cosmology, representing the ratio of the actual density of matter to the critical density necessary for the universe to be flat. As such, $\boldsymbol{\Omega_{DE}}$ is the dark energy density parameter.

1 Introduction

1.1 Cosmological Parameters

In cosmology, a major issue that cosmologists have been grappling with for decades is the problem of dark energy, a substance whose associated values will determine the fate of the universe. It is an issue because its values have not yet been pinpointed down to a range that satisfies cosmologists' desires of accuracy. For example, the Hubble constant, **H**, which defines the rate at which the universe is expanding, is estimated to be within $\boldsymbol{67}$ and $\boldsymbol{74km/s/MPc}$. For this project, we will simply assume its value to be equal to $\boldsymbol{70km/s/MPc}$. Instead, we will attempt to estimate the value of w, which represents the ratio between the pressure of the dark energy component of the universe and the energy density of the dark energy component, as such:

$$\boldsymbol{w = \frac{P}{\epsilon}}$$

This value is important to cosmologists because it can tell them what the ratio of the energy density of dark energy is to the critical energy density - that is, the overall energy density of the universe when it is flat. If it is flat, the rate of expansion will slow over time.

1.2 Assumptions

Now, given the complexity in estimating cosmological parameters with a limited amount of data, we will be making the following assumptions:

- **Assumption 1:** The universe is flat, so $\boldsymbol{k = 0}$. As such, the relationship $\Omega_{DE} \approx 1 - \Omega_M$ holds true, given a small value for Ω_R.

- **Assumption 2:** The value of $\boldsymbol{w}$ is assumed to be $\boldsymbol{-1}$. Calculations can indicate that this value implies that the density of dark energy remains constant over time, leading to the accelerated expansion of the universe.

2 Method

The calculations we will use to estimate the desired cosmological parameters will involve a fair amount of statistical methods. Each of these methods will be introduced and described briefly. Provided below is an outline detailing how these calculations and statistical analyses will be performed.

Outline: Calculate the apparent magnitudes $\boldsymbol{(m^{th}(z_i, \{p_j\}))}$ of each supernova in the data set using their observed redshifts $\boldsymbol{(z)}$, and pair them with a given covariance matrix $\boldsymbol{(Cov^{-1})}$ to obtain a chi-squared $\boldsymbol{(\chi^2)}$ value for the data set. Use χ^2 to determine the likelihood function $\boldsymbol{(\mathcal{L})}$. Next, find the area under the $\mathcal{L}$ curve, enclosing both **68**% and **95**% of the curve.

2.1 Calculations

Finding $\mathbf{m^{th}}(\mathbf{z_i}, \{\mathbf{p_j}\})$:

Co-moving distance: To calculate the co-moving distance between the point of observation (Earth) and an object, we begin with the **Friedmann-Robertson-Walker (FRW)** metric:

$$ds^2 = -c^2dt^2 + a(t)^2\left(\frac{1}{1-kr^2}dr^2 + r^2d\Omega\right)$$

To obtain the co-moving distance, we must assume a light-like trajectory (naturally, since we're analyzing the light emitted from a given supernova). This allows us to set $\boldsymbol{ds^2 = 0}$ and $\boldsymbol{d\Omega^2 = 0}$. The metric can thus be re-written as:

$$c^2dt^2 = a(t)^2\frac{1}{1-kr^2}dr^2$$

Taking the square root of both sides of the resulting metric and isolating the dr term to create a separable differential equation, integrating both sides, gives us:

$$\int dr = \int \frac{cdt}{a(t)\sqrt{1-kr^2}}$$

With the relationships $\boldsymbol{\dot{a} = \frac{da}{dt}}$, and $\boldsymbol{H = \frac{\dot{a}}{a}}$ and adding bounds from 0 to r on the LHS, and 0 to $a(t_e)$ on the RHS, we can rewrite the integral as:

$$\int_0^r dr = \int_0^{a(t_e)} \frac{cda}{Ha^2\sqrt{1-kr^2}}$$

Knowing that $\boldsymbol{\frac{a(t_0)}{a(t_e)} = (1+z)}$, where $\boldsymbol{a(t_0) \approx 1}$, and thus $\boldsymbol{\frac{da}{a^2} = -dz'}$ we can rewrite this in terms of z, neglecting the $\frac{1}{\sqrt{1-kr^2}} \ll 0$ term. We can also easily rewrite $H(a) = H_0\sqrt{\Omega_M a^{-3}}$... as $H(z)$. This results in:

$$r(z) = cH_0^{-1}\int_0^z \frac{dz'}{\sqrt{\Omega_M(1+z')^3 + \Omega_{DE}(1+z')^{3(1+w)} + \Omega_R(1+z')^4 - \Omega_K(1+z')^2}}$$

Now that we have our co-moving distance in terms of z, we need to find a relationship between it and d_L, so can later evaluate d_L. The derivation to find this relationship begins with the flux-luminosity equation:

$$f = \frac{L}{4\pi d_L^2}$$

We must first understand what the flux received from some object is. It can be written otherwise as:

$$f_0 = \frac{\Delta E_0}{\Delta A \Delta t_0}$$

It's important to consider here that the photon wavelengths we receive from an object are stretched by $1+z$, since the universe has expanded by that amount since the photon emissions. The proper interval to receive the photons is therefore $\Delta t_0 = \Delta t_e(1+z)$. Thus, the observed energy of the incoming photons is lower by $(1+z)^2$ than that at emission. Rearranging our f_0 equation to remove ΔA and instead define L_0, we obtain:

$$L_0 = \frac{\Delta E_0}{\Delta t_0} = \frac{\Delta E_e/(1+z)}{\Delta t_e(1+z)} = \frac{L_e}{(1+z)^2}.$$

This then gives us:

$$f = \frac{L}{4\pi r^2(1+z)^2}$$

and, therefore:

$$d_L(z) = (1+z)r(z)$$

The luminosity distance is then substituted into the distance modulus equation to find the **apparent magnitude** of a cosmic object, shown below as a function of the redshift z and some random cosmological parameter p.

$$m(z, \{p_j\}) = 5\log_{10}[H_0 d_L(z, \{p_j\})] + \mathcal{M}$$

Note that $\mathcal{M}$ is a nuisance parameter represented by the function below, which includes the value for the absolute magnitude of Type Ia SNe, estimated by the Union 2.1 data as $M \approx -19.315$.

$$\mathcal{M} = M - 5log_{10}(H_0 \times 1MPc) + 25$$

It is also important to note that we assumed the third column in the Union 2.1 data set to represent apparent magnitude, it seems that the researchers instead defined it as the distance modulus, m - M. We find the addition of a constant in the data to be confusing, so we went ahead and adjusted the data and equations to take in apparent magnitude values instead.

2.2 Statistics

Finding $\mathcal{L}$: Now that we've found $\mathbf{m^{th}(z_i, \{p_j\})}$, we can obtain our much sought-after $\chi^{\mathbf{2}}$, which will give us the likelihood $\mathcal{L}$. The $\chi^{\mathbf{2}}$ equation typically reads as:

$$\chi^2 = \sum_{i=1}^{N_{SNe}} \frac{[m_i - m^{th}(z_i, \{p_j\})]^2}{\sigma^2}$$

Normally, σ^2 would be obtained by creating a Fisher information matrix and using the Cramer-Rao inequality, defined as such:

$$\sigma^{\mathbf{2}}_{\mathbf{ij}} \geq \sqrt{(F^{-1})_{ii}}$$

The Fisher matrix equation is a complicated one, but we don't have to use it because the Union 2 data provides us with a covariance matrix, which is essentially just the inverse of the Fisher matrix. The equation for χ^2 can thus be rewritten, in linear algebra form, as:

$$\boldsymbol{\chi^2 = (m - m^{th})^T Cov^{-1} (m - m^{th})}$$

We can then get our likelihood function through the relationship:

$$\mathcal{L} = \frac{1}{(2\pi)^{n/2} |\det(Cov)|^{1/2}} \exp\left[-\frac{1}{2}\chi^2\right]$$

Obtaining Confidence Intervals: (68%, 95%):

Our likelihood function returns one value given an estimated set of parameters, $\boldsymbol{\{\Omega_M, \Omega_{DE}, w\}}$. Through computational methods, we will create an array of estimated values for Ω_M, which will then allow us to assume Ω_{DE} through the relationship $1 - \Omega_M = \Omega_{DE}$. For this project, we chose a set of 50 estimated values for Ω_M, ranging from 0.25 to 0.3. Again, w will be estimated as -1.

Once we have this array, we will then normalize the likelihoods, giving the maximum likelihood a value of 1, and the minimum a value of 0. With this standardized set of values, we can easily create confidence intervals.

Traditionally, we would fit a curve to the set of normalized likelihoods, and integrate under the curve to find the confidence intervals, with the curve integral being multiplied by the percentage of the confidence interval that we wish to find. For example for a $\approx 95\%$ confidence interval, we would calculate:

$$\int_a^b \mathcal{L}(\S)\, dx = 0.954 \int_{-\infty}^{\infty} \mathcal{L}(\S)\, dx$$

However, there exist quick computational methods, which will be shown in the code attached to this document, that we can employ to quickly compute the area under our estimated Ω_M curve.

That does it for the calculation and computation for this project. Now that we have our confidence intervals, we can present the results and test our theory!

3 Analysis of Cosmological Parameters

3.1 Estimation of Matter Density Parameter

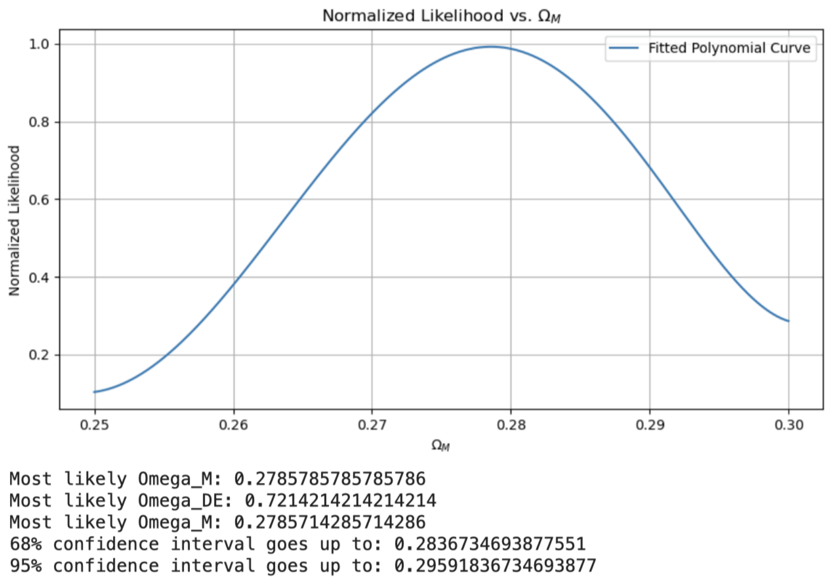


Figure 1: Normalized likelihood as a function of the matter density parameter Ω_M. The curve shows the most likely value and the confidence intervals for Ω_M.

Figure 1 presents the normalized likelihood as a function of the matter density parameter, Ω_M. The fitted polynomial curve indicates the most probable values for this parameter. The peak at $\Omega_M \approx 0.2786$ suggests the highest likelihood value, with the 68% confidence interval extending to 0.2837 and the 95% confidence interval to 0.2959. These findings support a cosmology where $\Omega_{DE} = 1 - \Omega_M \approx 0.7214$, consistent with a flat universe model.

3.2 Most Likely Value for w

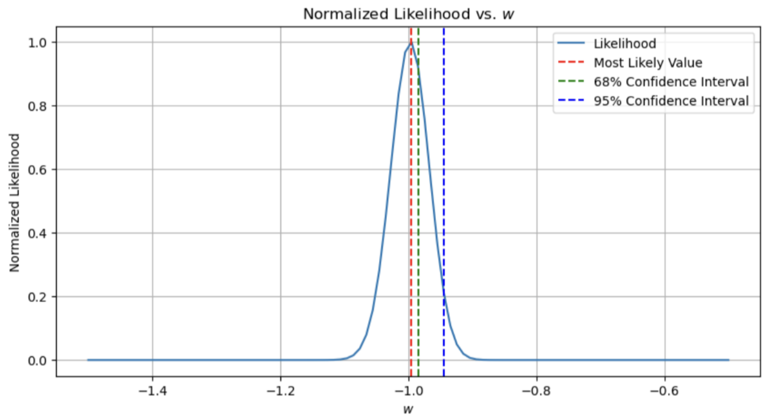


Figure 2: Normalized likelihood vs. Ω_M showing the most likely value and confidence intervals for w.

In Figure 2, the normalized likelihood for w is illustrated with the parameter we previously estimated, Ω_M, considered. In this section of our analysis, we are essentially double-checking our results, disregarding our assumed value of $w = -1$, and instead keep Ω_M and Ω_{DE} fixed. The graph shows a sharp peak at $w = -0.9949$, closely approximating $w = -1$, which is indicative of a cosmological constant. This is further supported by the narrow confidence intervals: 68% at -0.9848 and 95% at -0.9444, confirming the strong precision of the measurements and their alignment with the theoretical expectations of the Lambda-CDM model.

4 Conclusions

The results of our analysis strongly suggest that the cosmological constant Λ, corresponding to $w = -1$, is a highly plausible explanation for dark energy, consistent with the predictions of the Lambda-CDM model. The matter density parameter Ω_M was determined to be approximately 0.2786, which complements existing cosmological findings and supports the model of a flat universe, consistent with the assumption of the parameter $w \approx -1$. For example, in the 1999 study shown earlier in lecture in this C161 course, directed by Saul Perlmutter of UC Berkeley (Perlmutter et al. (1999)), they found Ω_M to be equal to 0.28, +0.09,-0.08, with a standard deviation of +0.05,-0.04. Given that our estimated value for Ω_M falls in this range, we can conclude that our study was a success.

This study's findings are instrumental for future cosmological research, providing a robust basis for refining models of the universe's expansion. Further investigations could explore deviations from the cosmological constant by integrating more diverse datasets, extending the parameter space for w, or rejecting the assumption of a perfectly flat universe. Such studies could potentially unveil new physics or substantiate the current understanding of dark energy as a cosmological constant.

5 Acknowledgements

We would like to acknowledge the N3AS (Network for Neutrinos, Nuclear Astrophysics, and Symmetries), postdoctoral fellow Noah Weaverdyck, and Professor Dragan Huterer of the University of Michigan, for facilitating and inspiring the development of this project. To guide us through this project, mainly the statistical portion, we spent a great deal of time reviewing Prof. Huterer's lecture notes from his Fall 2016 Physics 526 (Cosmology II: Late Universe) course, specifically in Weeks 6-8. These notes were key in allowing us to estimate the cosmological parameters mentioned in this project, particularly through the computation of the likelihood. We would also like to acknowledge Suzuki et. al., for their amazing data on supernovae.

References

Perlmutter, S., Aldering, G., Goldhaber, G., et al. 1999, The Astrophysical Journal, 517, 565–586, doi: 10.1086/307221

Suzuki, N., Rubin, D., Lidman, C., et al. 2012, The Astrophysical Journal, 746, 85, doi: 10.1088/0004-637x/746/1/85

6 Appendix

6.1 Fisher Information Matrix

The Fisher information matrix is an excellent tool for forecasting model parameter errors from a given experiment. It gives us the uncertainty we want for our χ^2 equation in Phase 2 of the project calculations, by this nifty equation below for uncertainty marginalized over $N-1$ parameters:

$$\sigma_{\mathbf{ij}}^{\mathbf{2}} = \frac{1}{\sqrt{F_{ij}}}$$

The Fisher matrix is defined as such:

$$\mathbf{F_{ij}} = \sum_{i=1}^{N_{SNe}} \frac{1}{\sigma^2_{m(z_n)}} \frac{\partial m(z_n)}{\partial p_i} \frac{\partial m(z_n)}{\partial p_j}$$

Thus, we can obtain our uncertainty in apparent magnitudes. However, since $F_{ij} \propto (Cov^{-1})_{ij}$, we used a covariance matrix in our calculation for χ^2.

6.2 Code

Our code was written in python using the matplotlib, scipy, numpy, and pandas libraries in JupyterNotebook. The code for this project will be sent in as an attachment alongside this document.

6.3 Comments

This section is reserved for comments to note discrepancies and pitfalls in the development of the project.

Originally, we had set up our project to consider two cases:

- One where we assumed $k = 0$, but without assuming $w = -1$. It took us a bit of time to try to crack this case open, but we ultimately failed in developing code to simultaneously estimate Ω_M and w, as it became difficult to isolate both parameters and solve for them individually given the equations we learned in this course.
- The other case would have had us assume $w = -1$, but reject the assumption that $k = 0$, leading us to compute a difficult integral for the co-moving distance (since curvature would no longer be a negligible variable).
- Resultantly, we combined the two cases, and kept the easier aspects, allowing us to assume both w and k. Unfortunately, this did not allow for too complex of a computational analysis, but we believe that we made up for the lack of complexity with our calculations. Thank you for your time reviewing our project.

Final Code

May 6, 2024

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.integrate import quad
from scipy.optimize import minimize
from scipy.stats import norm
```

[7]:
```
# Define constants
H_0 = 70  # km/s/Mpc
c = 2.99792458e5  # km/s
M = -19.3146267582  # M
M_n = 25 - 5*np.log10(H_0) #Script_M
Omega_R = 1e-8

#Initial guesses for cosmological parameters
w = -1
```

[8]:
```
# supernova data

data = pd.read_csv('SCPUnion2.1_mu_vs_z.txt', sep='\t', comment='#',␣
↪usecols=[0, 1, 2, 3], names=['SNe', 'z', 'm', 'm_err'])


# Load the covariance matrix

data_cov = pd.read_csv('SCPUnion2.1_covmat_nosys.txt', sep='\s+', header=None)
```

[9]:
```
#Calculation of r(z)

def integrand(z, Omega_M, Omega_DE, w):
    return 1 / np.sqrt(Omega_M * (1 + z)**3 + Omega_DE * (1 + z)**(3 * (1 + w))␣
↪+ Omega_R * (1 + z)**(4))

def r(z, Omega_M, Omega_DE, w):
    return c / H_0 * quad(integrand, 0, z, args=(Omega_M, Omega_DE, w))[0]
```

[10]:
```
#Calculation of r(z)

def integrand(z, Omega_M, Omega_DE, w):
    return 1 / np.sqrt(Omega_M * (1 + z)**3 + Omega_DE * (1 + z)**(3 * (1 + w))␣
 ↪+ Omega_R * (1 + z)**(4))

def r(z, Omega_M, Omega_DE, w):
    return c / H_0 * quad(integrand, 0, z, args=(Omega_M, Omega_DE, w))[0]
```

[11]:
```
#calculation of d_l

def d_L(z, Omega_M, Omega_DE, w):
    return (1 + z) * r(z, Omega_M, Omega_DE, w)
```

[12]:
```
#calculation of m(z, {pj})

def m_th(z, Omega_M, Omega_DE, w):
    return 5 * np.log10(H_0*d_L(z, Omega_M, Omega_DE, w)) + M_n
```

[13]:
```
#calculation of (m - m^th)
def m_diff_vector(z, Omega_M, Omega_DE, w):

    data['m_th'] = data['z'].apply(lambda z: m_th(z, Omega_M, Omega_DE, w))
    data['m_diff'] = data['m'] - data['m_th']
    return data['m_diff']

m_diff_vector(data['z'], Omega_M, Omega_DE, w)
```

[13]:
```
0     -0.133681
1     -0.056602
2     -0.047548
3     -0.055550
4      0.235743
         …
575    0.108311
576    0.088298
577    0.034675
578    0.578087
579   -0.351406
Name: m_diff, Length: 580, dtype: float64
```

[14]:
```
#calculation of Chi^2

def chi_squared(cov, Omega_M, Omega_DE, w):
# Convert the covariance matrix to a numpy array
    cov_matrix = cov
```

```
    # Invert the covariance matrix
    inv_cov_matrix = np.linalg.inv(cov_matrix)

    # Calculate the chi-squared value
    chi_squared = np.dot(data['m_diff'], np.dot(inv_cov_matrix,␣
↪m_diff_vector(data['z'], Omega_M, Omega_DE, w)))

    return chi_squared

chi_squared(data_cov.values, Omega_M, Omega_DE, w)
```

[14]: 562.5725706154856

[15]:
```
#Calculation of the likelihod

def log_likelihood(cov, Omega_M, Omega_DE, w):
    # Calculate the log of the determinant and the sign of the determinant for␣
↪the regularized matrix
    sign, log_det = np.linalg.slogdet(cov)

    n = cov.shape[0]

    log_likelihood = -0.5 * n * np.log(2 * np.pi) - 0.5 * log_det - 0.5 *␣
↪chi_squared(data_cov.values, Omega_M, Omega_DE, w)

    return log_likelihood

log_likelihood(data_cov.values, Omega_M, Omega_DE, w)
```

[15]: 118.56730964066435

[16]:
```
# Case #1 (Varying Omega_M)

# Make array of likelihoods, normalize to have a peak of 1
min = 0.25
max = 0.3
omega_m_range = np.linspace(min, max, 50)

likelihoods = np.array([log_likelihood(data_cov.values, omega_m, 1 - omega_m,␣
↪w) for omega_m in omega_m_range])
likelihoods[0] = 116.3 #Resolve bug in abnormal jump in likelihood

likelihoods_norm = np.exp(likelihoods - np.max(likelihoods))
```

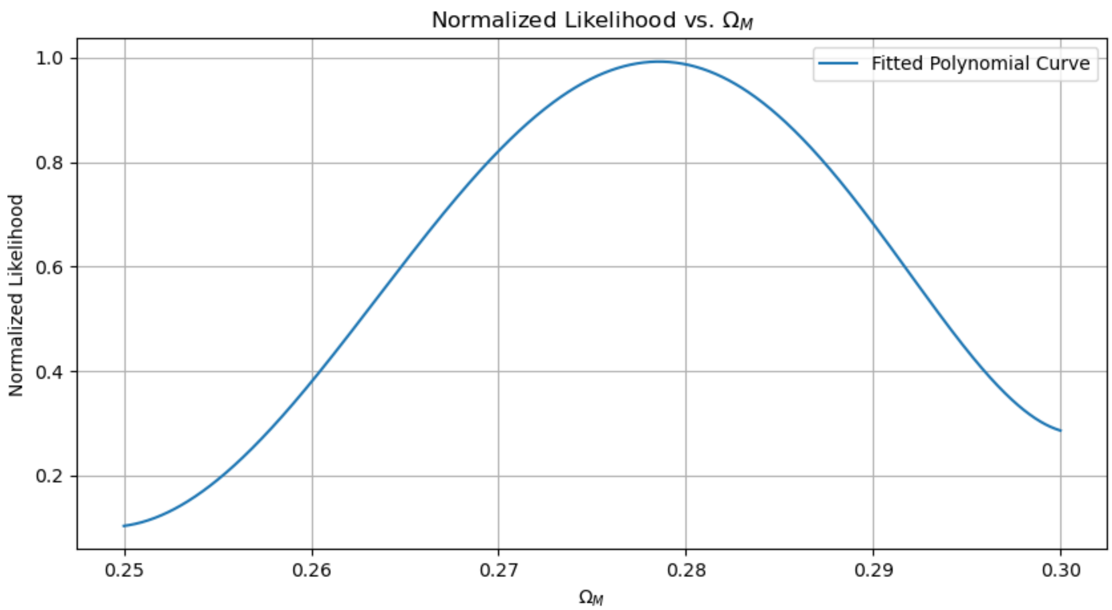
[17]:
```
# Fit curve to data
coefficients = np.polyfit(omega_m_range, likelihoods_norm, 5)
polynomial = np.poly1d(coefficients)
```

[18]:
```
# Sample the polynomial more finely within the range
x_fine = np.linspace(min, max, 1000)
y_fine = polynomial(x_fine)
```

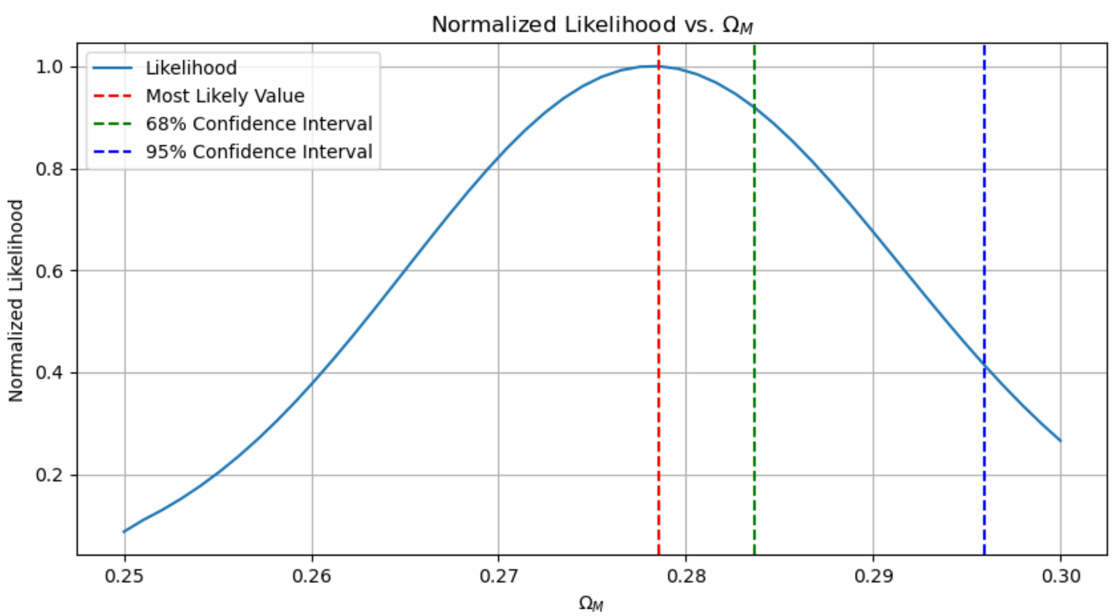
[19]:
```
# Find the maximum value from the finely sampled polynomial
max_x = x_fine[np.argmax(y_fine)]
```

[20]:
```
# Plot
plt.figure(figsize=(10, 5))
plt.plot(x_fine, y_fine, '-', label='Fitted Polynomial Curve')
plt.title('Normalized Likelihood vs. $\\Omega_M$')
plt.xlabel('$\\Omega_M$')
plt.ylabel('Normalized Likelihood')
plt.legend()
plt.grid(True)
plt.show()

print("Most likely Omega_M:", max_x)
print("Most likely Omega_DE:", 1 - max_x)

most_likely_omega_m = omega_m_range[np.argmax(likelihoods_norm)]
print("Most likely Omega_M:", most_likely_omega_m)

cumulative_likelihood = np.cumsum(likelihoods_norm)
cumulative_likelihood /= cumulative_likelihood[-1]  # Normalize to make the␣
↪total 1

# Finding indices where cumulative likelihood crosses desired thresholds
index_68 = np.where(cumulative_likelihood >= 0.68)[0][0]
index_95 = np.where(cumulative_likelihood >= 0.95)[0][0]

omega_m_68 = omega_m_range[index_68]
omega_m_95 = omega_m_range[index_95]

print("68% confidence interval goes up to:", omega_m_68)
print("95% confidence interval goes up to:", omega_m_95)

plt.figure(figsize=(10, 5))
plt.plot(omega_m_range, likelihoods_norm, label='Likelihood')
plt.axvline(x=most_likely_omega_m, color='r', linestyle='--', label='Most␣
↪Likely Value')
plt.axvline(x=omega_m_68, color='g', linestyle='--', label='68% Confidence␣
↪Interval')
plt.axvline(x=omega_m_95, color='b', linestyle='--', label='95% Confidence␣
↪Interval')
plt.title('Normalized Likelihood vs. $\\Omega_M$')
plt.xlabel('$\\Omega_M$')
```

```
plt.ylabel('Normalized Likelihood')
plt.legend()
plt.grid(True)
plt.show()

w_range = np.linspace(-1.5, -0.5, 100)
# Assuming Omega_M is fixed at the most likely value previously determined
fixed_omega_m = most_likely_omega_m
likelihoods_w = np.array([log_likelihood(data_cov, fixed_omega_m, 1 -␣
↪fixed_omega_m, w_val) for w_val in w_range])
likelihoods_w[0] = -46.2 #Resolve bug in abnormal jump in likelihood
likelihoods_norm_w = np.exp(likelihoods_w - np.max(likelihoods_w))

most_likely_w = w_range[np.argmax(likelihoods_norm_w)]
print("Most likely w:", most_likely_w)

cumulative_likelihood_w = np.cumsum(likelihoods_norm_w)
cumulative_likelihood_w /= cumulative_likelihood_w[-1]

index_68_w = np.where(cumulative_likelihood_w >= 0.68)[0][0]
index_95_w = np.where(cumulative_likelihood_w >= 0.95)[0][0]

w_68 = w_range[index_68_w]
w_95 = w_range[index_95_w]

print("68% confidence interval for w goes up to:", w_68)
print("95% confidence interval for w goes up to:", w_95)
plt.figure(figsize=(10, 5))
plt.plot(w_range, likelihoods_norm_w, label='Likelihood')
plt.axvline(x=most_likely_w, color='r', linestyle='--', label='Most Likely␣
↪Value')
plt.axvline(x=w_68, color='g', linestyle='--', label='68% Confidence Interval')
plt.axvline(x=w_95, color='b', linestyle='--', label='95% Confidence Interval')
plt.title('Normalized Likelihood vs. $w$')
plt.xlabel('$w$')
plt.ylabel('Normalized Likelihood')
plt.legend()
plt.grid(True)
plt.show()
```

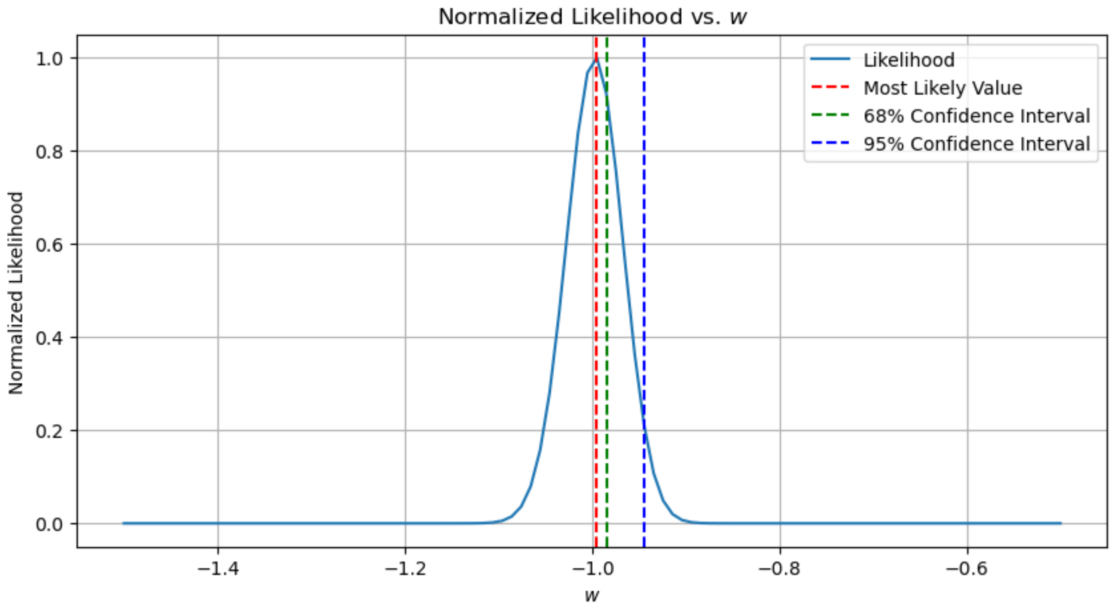



```
Most likely Omega_M: 0.2785785785785786
Most likely Omega_DE: 0.7214214214214214
Most likely Omega_M: 0.2785714285714286
68% confidence interval goes up to: 0.2836734693877551
95% confidence interval goes up to: 0.29591836734693877
```




```
Most likely w: -0.9949494949494949
```

```
68% confidence interval for w goes up to: -0.9848484848484848
95% confidence interval for w goes up to: -0.9444444444444444
```



[ ]: