實驗物理學（二）
實驗結報

Fundamental Python
Fitting Data of Simple Pendulum

Group 2

洪　瑜 B125090009
黃巧涵 B122030003
洪懌平 B102030019

2025/04/01

1 摘要

本實驗旨在探討單擺週期與擺長之關係，並透過Chi-square擬合方法推算重力加速度g。

儘管此主題爲基礎物理定律的驗證，缺乏原創性，但在實驗物理學課程中極具訓練價值，尤其有助於理解誤差處理與數據擬合的實用性與局限性。我們將週期與擺長取對數後進行線性化處理，並使用Chi-square擬合最小化加權殘差平方和，考慮每筆資料的標準差作爲權重，以提升擬合的可靠性。

實驗結果顯示，最佳擬合參數爲斜率$m = 0.4802$、截距$b = 0.7927$，對應估算之重力加速度爲$g = 9.969\,m/s^2$，相對誤差約 1.66%。卡方統計量$\chi^2 = 0.1525$，顯著小於自由度，推測可能爲誤差估計偏大或模型過擬合所致。與其他方法（如least squares與MCMC）相比，Chi-square擬合在誤差加權上的應用雖理論合理，但若未妥善估算誤差，仍可能導致擬合結果過於樂觀。根據本次擬合結果與殘差圖分析，模型整體表現良好，無明顯系統性偏差。

2 前言

2.1 Simple Pendulum

A simple pendulum is conventionally defined as a set of a massless and tough stick or wire whose length is L, one side rooted to a fixed point and the other attached to a point mass source m. (See Fig.1)

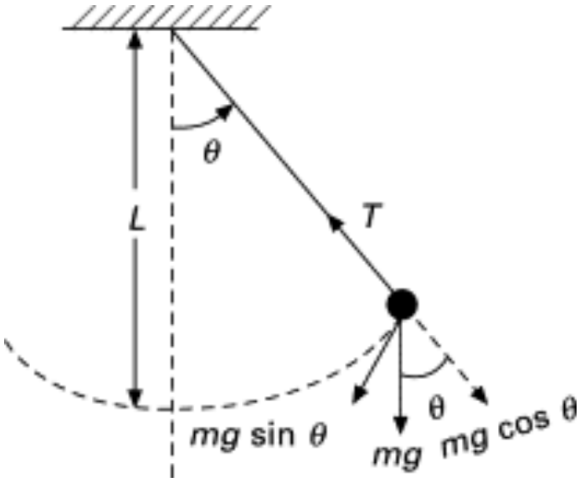

Figure 1: Demonstration of a simple pendulum. (Credit: This site)

Using Newton's Second Law, we can get

$$mL\frac{d^2\theta}{dt^2} = -mgsin\theta \tag{1}$$

As $\theta \sim 0$, i.e., under the small-angle approximation, $sin\theta \sim \theta$ (if we take the first Taylor expansion), Eq. 1 can then be simplified to:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\theta = 0 \tag{2}$$

The solution to this ODE can actually be regarded as the simple harmonic oscillation/motion (SHM), i.e., the superposition of sinusoidal functions.

$$\theta(t) = Asin(\omega t) + Bcos(\omega t) \tag{3}$$

$$\begin{cases} \omega = \sqrt{\frac{g}{L}} \\ T = \frac{2\pi}{\omega} = 2\pi\sqrt{\frac{L}{g}} \end{cases} \tag{4}$$

where T is the period of the simple pendulum, and ω is the angular frequency.

To be noted, the period only depends on the pendulum's length. In this case, when we analyze the correlation between the length and the period of the pendulum (Eq.4), the linear relation can be found through logarithmic transformation.

$$\ln T = \ln(2\pi) + \frac{1}{2}\ln g \tag{5}$$

We can then fit the experimental data, measuring period and pendulum's length, of a simple pendulum with Eq.5 to determine the gravitational acceleration g by

$$g = exp\left(2\ln(2\pi) - 2A\right) \tag{6}$$

where A is the intercept of the linear relationship (Eq.5).

2.2 Fitting Methods

2.2.1 Least Squares Method (LS)

The least squares method finds the best-fitting curve or line through a set of data points by minimizing the sum of the squares of the differences S (called residuals) between the observed values y_i and the values predicted by the model $f(x_i)$.

$$S = \sum_i (y_i - f(x_i))^2 \tag{7}$$

The reasons why the residuals are taken squared:

- Emphasizes larger errors more
- Avoids negative errors canceling out positive ones.

In `scipy.optimize.curve_fit` function, it uses optimized non-linear least squares method to find the best-fitting parameters.

2.2.2 χ^2 Fitting

χ^2 fitting is similar to least squares fitting, but it weights each data point based on its uncertainty.

$$\chi^2 = \sum_i \left(\frac{y_i - f(x_i)}{\sigma_i}\right)^2 \tag{8}$$

This makes it especially useful when your measurements have different error bars in different situations. It is different from the least square method, which adopts uniform uncertainties at varying data points.

Moreover, χ^2 fitting allows to estimate the wellness of the fitting by adopting reduced χ^2

$$\chi^2_\nu = \frac{\chi^2}{\nu} \tag{9}$$

in which ν is the degree of freedom (DOF), which equals the number of data points minus the number of the fitting parameters.

- Smaller χ^2, better fitting result
- $\chi^2_\nu \sim 1$, the model explains the data within expected errors.
- $\chi^2_\nu \gg 1$, very poor fit, serious mismatch between model and data.
- $\chi^2_\nu < 1$, possibly overfitting, or the errors are overestimated.

2.2.3 MCMC Fitting

MCMC fitting stands for Markov Chain Monte Carlo fitting, and it's a statistical method for estimating the probability distribution of model parameters, especially when models are nonlinear or have many parameters. MCMC samples from a probability distribution using a Markov chain, where each sample depends only on the previous one. It's especially useful when

- The function you're fitting is nonlinear.
- You want error bars or confidence intervals on your parameters.
- The parameter space is multi-dimensional.

Commonly, the Bayesian theorem is adopted to compute the probability distribution of the posterior.

$$P(\theta|data) \propto \mathcal{L}(\theta)P(\theta) \tag{10}$$

where $\mathcal{L}(\theta)$ is the liklihood function.

3 實驗步驟

1. 將提供數據導入程式碼。
2. 利用不同的擬合方法比較結果：
   (a) `curve_fit`
   (b) Least-squared fitting
   (c) χ^2 fitting
   (d) MCMC fitting
3. 將各個方法透過線性和對數模型擬合並觀察結果，並將結果與理論值繪製成圖。

4 實驗數據與數據分析

4.1 單擺數據

此爲課堂提供之數據：（T爲周期）

擺長L(cm)	1st擺盪10T所需時間(s)	2nd擺盪10T所需時間(s)	3rd擺盪10T所需時間(s)
56.25	15.63	15.55	15.28
48.15	13.75	13.61	13.91
42.25	13.21	13.21	13.15
36.00	12.83	12.55	15.61
30.25	11.23	11.21	11.30
25.00	10.33	10.41	10.35
20.25	9.45	9.76	9.50
16.00	8.41	8.36	8.43
12.25	7.36	7.36	7.31
9.00	6.31	6.50	6.56

4th擺盪10T所需時間(s)	5th擺盪10T所需時間(s)	平均擺盪10T所需時間(s)	$\sqrt{L}(cm^{\frac{1}{2}})$
15.46	15.66	15.52	7.50
13.41	14.06	13.75	6.94
13.23	13.06	13.17	6.50
12.81	12.63	12.69	6.00
11.19	11.28	11.24	5.50
10.36	10.35	10.36	5.00
9.66	9.68	9.61	4.50
8.36	8.51	8.41	4.00
7.33	7.31	7.33	3.50
6.45	6.45	6.45	3.00

4.2 前置作業

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
g0 = 9.80665
from scipy.optimize import curve_fit, minimize
import emcee
import corner
```

Figure 2: 前置作業程式碼(1)

在Fig.2中，我們引入函式庫：

- numpy (np): 用於數值運算
- pandas (pd): 用於讀取CSV檔案並處理表格數據
- astropy.constants.g0: 取得標準重力加速度g_0=9.80665 m/s^2
- scipy.optimize的curve_fit 和 minimize: 用來執行數據擬合
- emcee 和 corner: 用於貝葉斯MCMC參數估計

```python
def linear_function(x, a, b):
    return a * x + b

def chi_sq(y_data, y_model, sigma):
    return np.sum((y_data - y_model)**2 / sigma**2)


fname = 'data_7_group2.csv'
data = pd.read_csv(fname)


print(data.columns)
```

Figure 3: 前置作業程式碼(2)

定義線性函數$y = ax + b$、卡方檢定函數；並讀取課程提供之CSV檔案，最後輸出欄位名稱。

```python
pendulum_length = np.array(data['L(cm)']) * 1e-2
n_length = len(pendulum_length)
pendulum_length_matrix = np.tile(pendulum_length, 5)
print(pendulum_length)
```

Figure 4: 前置作業程式碼(3)

Fig.4轉換數據長度單位並把擺長複製五次，於繪圖對應使用，並輸出擺長。

```python
avg_10period      = np.array(data['AVG-10T(s)'])
avg_1period       = avg_10period / 10
period_10swing  = []
for idx, rep in enumerate(range(1,6)):
    period_10swing.append(data[f'{rep}-10T'])
period_10swing = np.array(period_10swing)

std_10period = []
std_1period = []
std_1period_log = []
for i in range(n_length):
    std_10period.append(np.std(period_10swing[:, i]))
    std_1period.append(np.std(period_10swing[:, i]/10))
    std_1period_log.append(np.std(np.log(period_10swing[:, i]/10)))

std_10period = np.array(std_10period)
std_1period = np.array(std_1period)

period_10swing = period_10swing.flatten()
period_1swing = period_10swing / 10
```

Figure 5: 前置作業程式碼(4)

```python
print(pendulum_length)
print(pendulum_length_matrix)
print(avg_1period)
print(period_10swing)
print(std_1period)
```

Figure 6: 前置作業程式碼(5)

我們先計算擺動週期：取十次週期求平均，並將1-10T到5-10T取出來並存成5×10陣列，爲5次測量的10次週期數據；接著計算標準差，並攤平成一維陣列、轉換單擺週期，最後輸出數值。

```python
l_axis = np.linspace(0.01, 1.0, 100)
t_ideal = 2 * np.pi * np.sqrt(l_axis / g0)
```

Figure 7: 前置作業程式碼(6)

產生100個擺長數據並根據公式Eq.4計算週期。

```python
plt.scatter(pendulum_length_matrix, period_1swing, s=50, marker='x', color='silver',
            label=f'Experiment')

plt.errorbar(pendulum_length, avg_1period, yerr=std_1period, fmt='o', label='Avg T', color='k', capsize=5)
plt.plot(l_axis, t_ideal, label='Theoretical T', color='teal')
plt.xlabel('Length (m)')
plt.ylabel('Average period of a swing (s)')
plt.xlim(left=0.05, right=0.6)
plt.ylim(bottom=0.6, top=1.6)
plt.title('Average period vs Length')
plt.legend()
#plt.savefig('./figures/raw_data.pdf', transparent=True)
plt.show()
```

Figure 8: 前置作業程式碼(7)

繪製單擺實驗數據與理論預測的比較圖表，包含實驗測量數據（銀色x標記）、平均週期與誤差棒（黑色圓點）(顯示每個擺長的平均測量值，並附帶標準差誤差棒，反映測量的不確定性)以及理論預測曲線（藍綠色曲線）；設定圖表標籤與範圍，設定x軸爲擺長(m)，y軸爲擺動週期(s)，`plt.legend()` 顯示圖例，區分測量數據、平均值與理論曲線。儲存並顯示圖表。

4.3 方法一：curve_fit

```
propt, _ = curve_fit(linear_function, np.log(pendulum_length_matrix), np.log(period_1swing.flatten()))
print(propt)
```

Figure 9: curve_fit程式碼(1)

使用curve_fit來擬合一條線性函數$\log(T) = a\log(L) + b$；propt[0]爲斜率，propt[1]爲截距b。

```
plt.scatter(np.log(pendulum_length_matrix), np.log(period_1swing.flatten()), s=50, marker='x', color='silver', label='Experiment')
plt.errorbar(np.log(pendulum_length), np.log(avg_1period), yerr=std_1period_log, fmt='o', label='Avg T', color='k', capsize=5)
plt.plot(np.log(l_axis), linear_function(np.log(l_axis), *propt), color='orangered', label=f'curve_fit')
plt.plot(np.log(l_axis), np.log(t_ideal), label='Theoretical T', color='teal', linestyle='--')
plt.xlabel('log(Length) [m]')
plt.ylabel('log(Duration of a swing) [s]')
plt.title('Duration of a swing vs Length (Log-Log)')
plt.xlim(left=np.log(0.05), right=np.log(0.65))
plt.ylim(bottom=np.log(0.5), top=np.log(1.7))
plt.legend()
plt.savefig('./figures/curve_fit_log.pdf', transparent=True)
plt.show()
```

Figure 10: curve_fit程式碼(2)

繪製Log-Log圖，並比較擬合結果與理論值。實驗數據的對數值用銀色交叉符號顯示，error bar使用黑色圓點（如Fig.13右圖）。

```
plt.scatter(pendulum_length_matrix, period_1swing.flatten(), s=50, marker='x', color='silver', label='Experiment')
plt.errorbar(pendulum_length, avg_1period, yerr=std_1period, fmt='o', label='Avg T', color='k', capsize=5)
plt.plot(l_axis, np.e**linear_function(np.log(l_axis), *propt), color='orangered', label=f'curve_fit')
plt.plot(l_axis, t_ideal, label='Theoretical T', color='teal', linestyle='--')
plt.xlabel('Length [m]')
plt.ylabel('Duration of a swing [s]')
plt.title('Duration of a swing vs Length')
plt.xlim(left=0.05, right=0.6)
plt.ylim(bottom=0.6, top=1.6)
plt.legend()
plt.savefig('./figures/curve_fit.pdf', transparent=True)
plt.show()
```

Figure 11: curve_fit程式碼(3)

繪製原始尺度的時間vs.擺長圖，已轉爲線性尺度（Fig.13左圖）。

```
g_fitted = np.e**(2*(np.log(2*np.pi) - propt[1]))
print(f'Ideal g = {g0.value:.3f} m/s^2')
print(f'Fitted g = {g_fitted:.3f} m/s^2')
print(f'Error = {np.abs(g_fitted - g0.value) / g0.value * 100:.3f}%')
```

Figure 12: `curve_fit`程式碼(4)

透過擬合結果估算重力加速度g。最後我們計算之結果$g = 9.902\,m/s^2$，與理論的$g_0 = 9.807\,m/s^2$誤差0.975%。

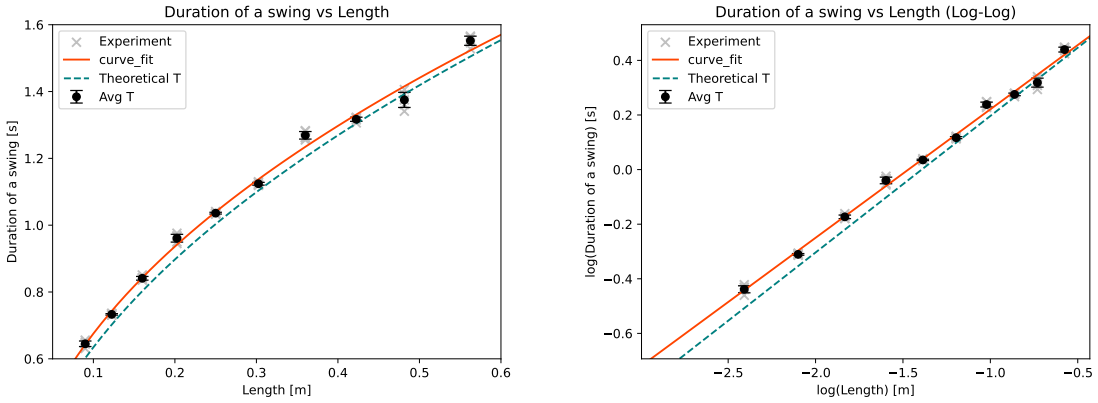

Figure 13: Enter Caption

4.3.1 `curve_fit` 數據分析

擺長（Length）與單擺週期（Period）之間的關係，理論上應符合eq.(6)和eq.(7)之公式。因此，將實驗數據取對數（log-log）後，理論上會呈現線性關係，理論之斜率值爲 0.5。

擬合結果的斜率 ≈ 0.47，與理論值 0.50接近。藉由截距項，可以得知對應的g值：$g \approx 9.902\,m/s^2$，與理論值的$g_0 = 9.807\,m/s^2$的相對誤差約爲 0.975%，表示此擬合結果具有高準確度。此外，透過圖表也可以看出擬合曲線與實驗平均值相符，顯示模型與實測結果大致具有一致性。雖然整體擬合結果看起來大致上符合預期，但是根據圖表可以看出，在較短擺長處部分與理論曲線之間有相對之下較爲顯著的差距。這可能是由於較短襬長處週期測量誤差比例較大，抑或者是運動過程非理想擺運動所造成的偏差有關，也可能是由於擺幅較大、擺錘起始角度有落差或者是其他人爲誤差所造成的。

4.3.2 `curve_fit` 誤差討論

擬合斜率偏差：擬合斜率爲0.471，而斜率之理論值爲0.5，相對誤差約爲 5.8%。
進一步討論誤差來源：

1. 缺乏誤差加權：
   預設情況下，`curve_fit` 將所有資料點視爲同樣可靠，不會根據不同資料點的誤差大小進行加權。因此如果有某些點的測量誤差較大，就會對擬合產生過度的影響。

2. 初始值選擇：
   雖然 `curve_fit` 本身會自動選擇初始值，但是在複雜模型中，若初始猜測與眞實值差太多，可能導致擬合陷入局部極小值而非全域最佳解、擬合失敗或收斂很慢等情形發生。

3. 異常值影響：
   最小平方法對異常值敏感，單一點偏差過大就有可能對整體擬合結果造成顯著影響。透過平均與標準差處理，以及觀察殘差圖與誤差棒，可以初步確認資料穩定性良好。

4.4 方法二：Least square fitting

```
A = np.vander(np.log(pendulum_length_matrix), 2)
C = np.diag(np.log(np.tile(std_1period, 5)) * np.log(np.tile(std_1period, 5)))
ATA = np.dot(A.T, A / (np.log(np.tile(std_1period, 5))**2)[:, None])
cov = np.linalg.inv(ATA)
w = np.linalg.solve(ATA, np.dot(A.T, np.log(period_1swing) / np.log(np.tile(std_1period, 5))**2))
print("Least-squares estimates:")
print("m = {0:.3f} ± {1:.3f}".format(w[0], np.sqrt(cov[0, 0])))
print("b = {0:.3f} ± {1:.3f}".format(w[1], np.sqrt(cov[1, 1])))
m_ls = w[0]
b_ls = w[1]
```

Figure 14: LS程式碼(1)

最小二乘法擬合np.vander$(x, 2)$產生 Vandermonde矩陣，對應線性模型$\log T = m \log L + b$。參數如下：

- C是對角矩陣，考慮log週期的誤差。
- ATA $= A^T\ C^{-1} A$是加權最小二乘法的矩陣。
- cov = np.linalg.inv(ATA) 計算參數的不確定性（共變異矩陣）。
- $w = [m, b]$ 是最小二乘法解出來的參數。

```
plt.scatter(np.log(pendulum_length_matrix), np.log(period_1swing.flatten()), s=50, marker='x', color='silver', label='Experiment')
plt.errorbar(np.log(pendulum_length), np.log(avg_1period), yerr=std_1period_log, fmt='o', label='Avg T', color='k', capsize=5)
# plt.plot(np.log(l_axis), linear_function(np.log(l_axis), *propt), color='orangered', label=f'curve_fit')
plt.plot(np.log(l_axis), np.log(t_ideal), label='Theoretical T', color='teal', linestyle='--')
plt.plot(np.log(l_axis), np.dot(np.vander(np.log(l_axis), 2), w), linestyle="-", color='slateblue', label="LS")
plt.xlabel('log(Length) [m]')
plt.ylabel('log(Duration of a swing) [s]')
plt.title('Duration of a swing vs Length (Log-Log)')
plt.xlim(left=np.log(0.05), right=np.log(0.65))
plt.ylim(bottom=np.log(0.5), top=np.log(1.7))
plt.legend()
plt.savefig('./figures/ls_log.pdf', transparent=True)
plt.show()
```

Figure 15: LS程式碼(2)

Log-Log圖，檢查LS擬合曲線是否與理論值相符；銀色交叉符號爲實際對數值，黑色圓點表error bar（如Fig.18 右圖）。

```
plt.scatter(pendulum_length_matrix, period_1swing.flatten(), s=50, marker='x', color='silver', label='Experiment')
plt.errorbar(pendulum_length, avg_1period, yerr=std_1period, fmt='o', label='Avg T', color='k', capsize=5)
# plt.plot(l_axis, np.e**linear_function(np.log(l_axis), *propt), color='orangered', label=f'curve_fit')
plt.plot(l_axis, t_ideal, label='Theoretical T', color='teal', linestyle='--')
plt.plot(l_axis, np.e**np.dot(np.vander(np.log(l_axis), 2), w), linestyle="-", color='slateblue', label="LS")
plt.xlabel('Length [m]')
plt.ylabel('Duration of a swing [s]')
plt.title('Duration of a swing vs Length')
plt.xlim(left=0.05, right=0.6)
plt.ylim(bottom=0.6, top=1.6)
plt.legend()
plt.savefig('./figures/ls.pdf', transparent=True)
plt.show()
```

Figure 16: LS程式碼(3)

原始尺度圖，藍色實線（LS擬合）與理論曲線（虛線）做比較，檢查擬合效果（Fig.18左圖）。

```
g_fitted = np.e**(2*(np.log(2*np.pi) - w[1]))
print(f'Ideal g = {g0.value:.3f} m/s^2')
print(f'Fitted g = {g_fitted:.3f} m/s^2')
print(f'Error = {np.abs(g_fitted - g0.value) / g0.value * 100:.3f}%')
```

Figure 17: LS程式碼(4)

最後我們計算之結果$g = 9.969\,m/s^2$，與理論的$g_0 = 9.807\,m/s^2$誤差1.660%

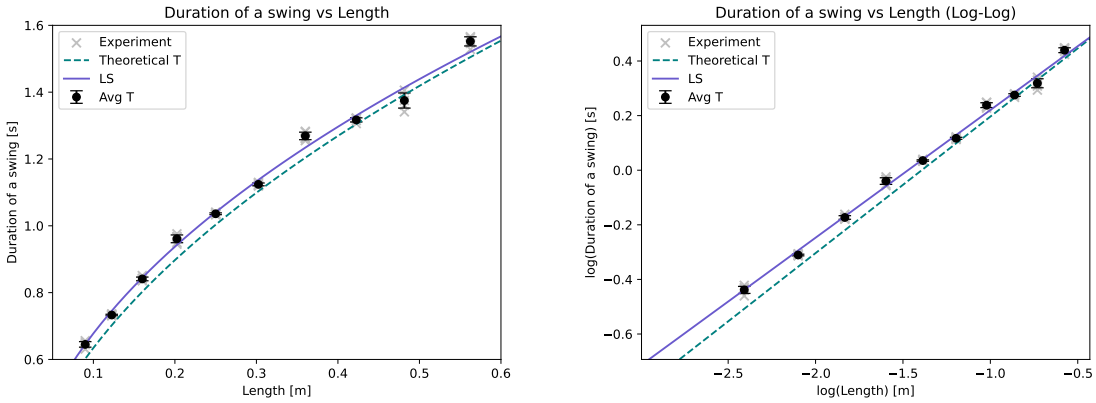

Figure 18: Enter Caption

4.4.1 LS 數據分析

此部分使用了加權最小平方法（Weighted Least Squares），考慮每筆資料的誤差（標準差）作爲權重，使得誤差較小的數據點在擬合中影響更大，進而嘗試提升實驗結果的準確度而確保不過度受到誤差大的數距點影響。

計算後的擬合參數爲：

- 斜率：$m = 0.485 \pm 0.012$
- 截距：$b = 0.668 \pm 0.008$

由上方數據可以看出，資料整體仍趨近於理論模型，落差較小。尤其式斜率數值接近斜率理論值0.5，即能代表擺長與週期的平方關係符合、成立。擬合後推算出的g值爲$g = 9.969\,m/s^2$，與理論的$g_0 = 9.807\,m/s^2$誤差1.660%。

與先前`curve_fit`得到誤差（0.975%）相比，雖然Least square fitting算出的g值誤差相對較大，但仍屬於合理範圍內，故顯示模型仍然準確合理。從圖表也可得知實驗數據（黑色點與誤差條）整體呈現隨擺長增加則週期增加的趨勢，符合理論公式eq.(6)的非線性曲線（虛線）。

4.4.2 LS 誤差討論

Least square fitting將每筆資料的測量不確定度納入考量，給予誤差小的點較高的權重，降低誤差較大的點對擬合結果的影響。但本次擬合中所得所得誤差相較之下較高，可能原因如下：

1. 誤差估計不穩定：
   資料誤差來源主要來自每筆資料標準差，如果資料的數量有限，使標準差本身不夠穩定，就有可能造成權重設置不準確進而導致誤差。

2. 誤差分佈未完全服從高斯分佈：
   最小平方法假設誤差爲常態分佈，但如果實際誤差分佈有偏態的情形，則加權擬合效果可能因此受限。

3. 異常值影響未修正：
   雖然LS fitting對誤差小的點給予較高權重，但依舊會受到異常值影響，因此可以搭配異常值偵測或穩健迴歸。

此外，理論上從截距推算重力加速度時，任何的微小誤差都會被指數放大。例如截距b之誤差±0.008，經指數與平方運算後，算出 g值的誤差爲 1.6%，反映出了非線性取線之誤差放大效應。因此若是要讓回推得出的g值更加準確，則需減少截距的不確定度，抑或是嘗試改用其他能更穩定的方法。

4.5 方法三：χ^2 fitting

```python
# Define the chi-squared function
def chi_sq_optimization(params, x_data, y_data, sigma):
    model = linear_function(x_data, *params)  # Use your model function
    return chi_sq(y_data, model, sigma)

# Initial guess for the parameters
initial_guess = [0.5, 0.5]  # Adjust based on your model

# Perform the optimization
result = minimize(chi_sq_optimization, initial_guess,
                  args=(np.log(pendulum_length_matrix), np.log(period_1swing.flatten()), np.log(np.tile(std_1period, 5))))

# Extract the optimized parameters
optimized_params = result.x
m_chisq = optimized_params[0]
b_chisq = optimized_params[1]
print("Optimized parameters:", optimized_params)
```

Figure 19: χ^2 fitting程式碼(1)

```python
chi_sq_val = chi_sq_optimization(optimized_params, L, np.log(periods), sigma)
print("Chi-square value:", chi_sq_val)
```

Figure 20: χ^2 fitting程式碼(2)

Fig.19是先定義chi-squared function，並定義：
params爲我們要最佳化的參數（直線之斜率或截距）；
x_data和y_data爲輸入數據；
sigma爲測量不確定性；
linear_function(x_data, params)爲一條直線；（$y = ax + b$）
chi_sq(y_data, model, sigma)爲計算χ^2。

接著設定初始參數，先猜測m=0.5、b=0.5；並進行最小化χ^2。

而因爲實驗要求擬合擺動週期與擺長的對數關係，所以需取對數，使其變線性關係（詳情見預習問題1、2）即可求得a、b。

最後把最佳化後的m、b，和χ^2值顯示出來。

最終輸出結果：

Optimized parameters: [0.48023142 0.79271392]
Chi-square value: 0.15251356436613928

```python
plt.scatter(np.log(pendulum_length_matrix), np.log(period_1swing.flatten()), s=50, marker='x', color='silver', label='Experiment')
plt.errorbar(np.log(pendulum_length), np.log(avg_1period), yerr=std_1period_log, fmt='o', label='Avg T', color='k', capsize=5)
# plt.plot(np.log(l_axis), linear_function(np.log(l_axis), *propt), color='orangered',linestyle='dashed',
#          label=f'curve_fit')
plt.plot(np.log(l_axis), np.log(t_ideal), label='Theoretical T', color='teal', linestyle='--')
# plt.plot(np.log(l_axis), np.dot(np.vander(np.log(l_axis), 2), w), "--k", label="LS")
plt.plot(np.log(l_axis), linear_function(np.log(l_axis), *optimized_params), linestyle="-", color='goldenrod', label="Chi-squared")
plt.xlabel('log(Length) [m]')
plt.ylabel('log(Duration of a swing) [s]')
plt.title('Duration of a swing vs Length (Log-Log)')
plt.xlim(left=np.log(0.05), right=np.log(0.65))
plt.ylim(bottom=np.log(0.5), top=np.log(1.7))
plt.legend()
plt.savefig('./figures/chisq_log.pdf', transparent=True)
plt.show()
```

Figure 21: χ^2 fitting程式碼(3)

Fig.21使用`plt.scatter()`繪製實驗數據點；x軸爲擺長取對數；y軸爲單次擺動週期取對數。利用交叉符號點出個別數據點。

接著繪出理論曲線和χ^2擬合的曲線，並設定座標軸與標題及範圍；結果爲Fig.24右圖。

```
plt.scatter(pendulum_length_matrix, period_1swing.flatten(), s=50, marker='x', color='silver', label='Experiment')
plt.errorbar(pendulum_length, avg_1period, yerr=std_1period, fmt='o', label='Avg T', color='k', capsize=5)
# plt.plot(l_axis, np.e**linear_function(np.log(l_axis), *propt), color='orangered',linestyle='dashed',
#          label=f'curve_fit')
plt.plot(l_axis, t_ideal, label='Theoretical T', color='teal', linestyle='--')
# plt.plot(l_axis, np.e**np.dot(np.vander(np.log(l_axis), 2), w), "--k", label="LS")
plt.plot(l_axis, np.e**linear_function(np.log(l_axis), *optimized_params), linestyle="-", color='goldenrod', label="Chi-squared")
plt.xlabel('Length [m]')
plt.ylabel('Duration of a swing [s]')
plt.title('Duration of a swing vs Length')
plt.xlim(left=0.07, right=0.6)
plt.ylim(bottom=0.5, top=1.6)
plt.legend()
plt.savefig('./figures/chisq.pdf', transparent=True)
plt.show()
```

Figure 22: χ^2 fitting程式碼(4)

Fig.22繪製擺長與擺動週期的關係圖，但和上部分的對數-對數圖不同，這次是線性尺度。

令x軸爲擺長、y軸爲擺動週期，用銀色交叉符號標出實驗數據，並繪出error bar（使用黑色圓點，表測量之不確定性）；最後劃出理論曲線和χ^2理論曲線與我們擬合出的結果（Fig.24左圖）。

```
g_fitted = np.e**(2*(np.log(2*np.pi) - b_chisq))
print(f'Ideal g = {g0.value:.3f} m/s^2')
print(f'Fitted g = {g_fitted:.3f} m/s^2')
print(f'Error = {np.abs(g_fitted - g0.value) / g0.value * 100:.3f}%')
```

Figure 23: χ^2 fitting程式碼(5)

Fig.23透過χ^2擬合的結果估算重力加速度g，並和理論值（g_0）比較。

計算方法參考預習問題1、2推導之公式。 最後我們計算之結果$g = 9.969\,m/s^2$，與理論的$g_0 = 9.807\,m/s^2$誤差1.660%。

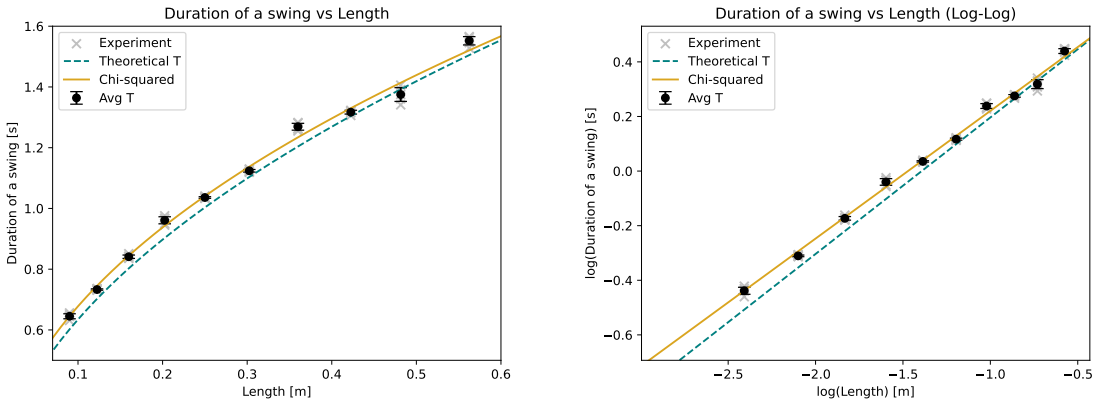

Figure 24: χ^2 擬合結果（左圖爲線性擬合；右圖爲對數擬合）

```python
# 計算模型預測值
model_logT = linear_function(L, m_opt, b_opt)

# 計算殘差
residuals = np.log(periods) - model_logT

# 印出殘差資訊
print("\nResiduals:")
print(f"{'L':>8} {'log(T)':>10} {'Model':>10} {'Residual':>10}")
for i in range(len(L)):
    log_T = np.log(periods[i])
    model_val = model_logT[i]
    residual = residuals[i]
    print(f"{L[i]:8.2f} {log_T:10.4f} {model_val:10.4f} {residual:10.4f}")


# 畫殘差圖
plt.figure(figsize=(8, 5))
plt.errorbar(np.log(L), residuals, yerr=sigma / periods, fmt='o', capsize=5, label="Residuals")
plt.axhline(0, color='red', linestyle='--', label="Zero Line")
plt.xlabel("log(擺長 L)")
plt.ylabel("殘差 Residuals (log(T) - model)")
plt.title("殘差圖 Residual Plot")
plt.legend()
plt.grid(False)
plt.show()
```

Figure 25: χ^2 fitting程式碼(6)

Fig.25為計算殘差的標準差，詳情見圖上程式碼註解，我們計算出之各數據殘差為：

擺長L(cm)	log(T)	Model	Residual
56.25	2.7492	2.7280	0.0212
48.15	2.6210	2.6533	-0.0322
42.25	2.5810	2.5905	-0.0095
36.00	2.5518	2.5136	0.0382
30.25	2.4186	2.4301	-0.0115
25.00	2.3351	2.3385	-0.0035
20.25	2.2460	2.2373	0.0087
16.00	2.1294	2.1242	0.0052
12.25	1.9961	1.9959	0.0001
9.00	1.8421	1.8479	-0.0058

殘差圖如下：

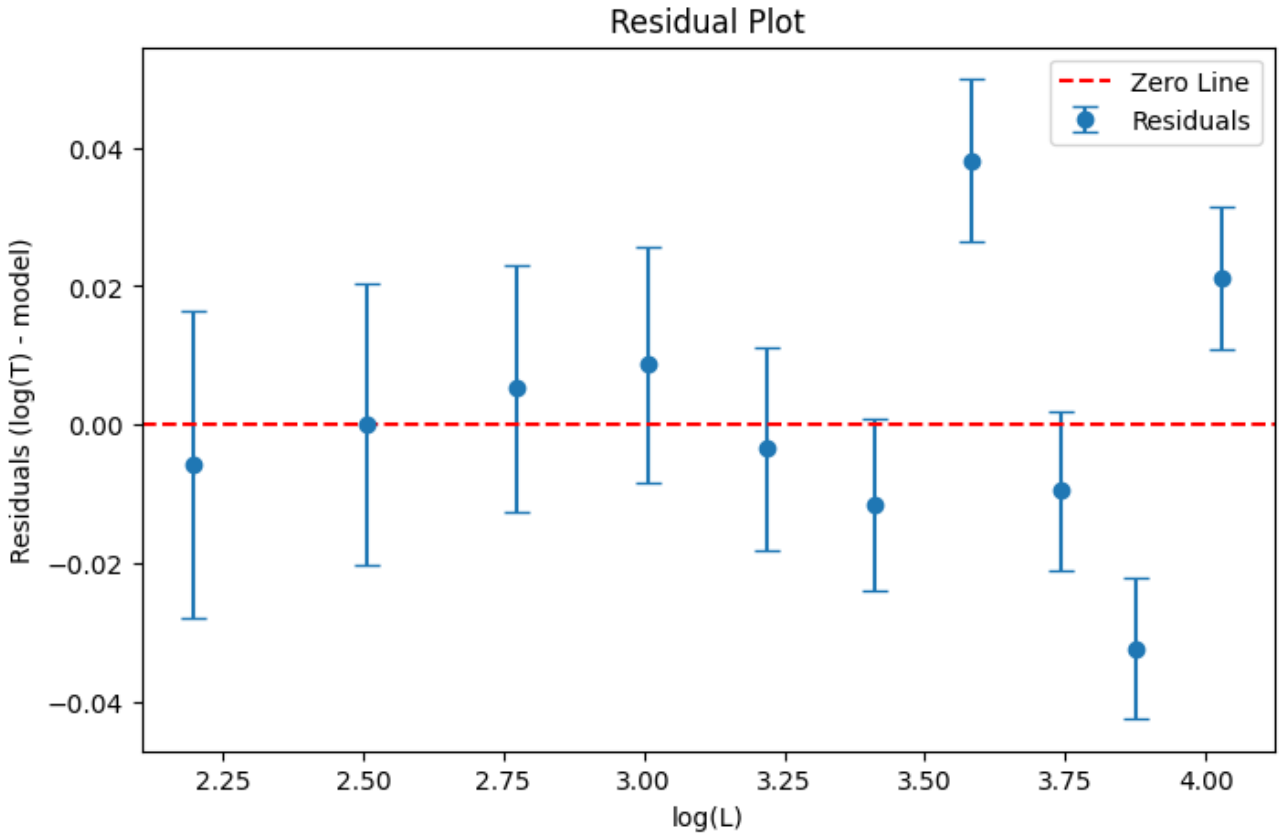

Figure 26: 殘差圖

4.5.1 χ^2 fitting 數據分析

從Fig.24左圖「擺動週期與擺長的關係」可見，實驗數據（黑色點與誤差條）整體呈現隨擺長增加而週期增加的趨勢，符合理論公式eq.(6)的非線性曲線（虛線）。

而理論曲線與部分實驗數據點有些許偏差，進一步使用Chi-square擬合（橘線）進行資料擬合，而我們得到最佳化的參數爲： $m = 0.4802$；$b = 0.7927$

其中b略大於理論值之0.5，推測來自實驗時的空氣阻力、測量誤差等非理想條件。

計算之χ^2值爲0.1525。

此值遠小於自由度，於誤差討論和問題討論第一題詳細討論；自由度爲$N - 2 = 8$：

$\chi^2 \approx 0.1525$

$\chi^2_{reduced} = \frac{0.1525}{8} \approx 0.0019$

Fig.24右圖爲相同資料轉換爲對數座標下的結果。理論預測爲一條斜率爲0.5之值線，而Chi-square擬合同樣顯示與實驗資料的對數變化趨勢更一致，進一步證實擬合品質。

見Fig.26，殘差圖呈現資料點約略隨機分佈於零線上下，而無特定趨勢。在$\log(L)$=3.5左右時較大，推測原因爲在測量數據時有較大的誤差（人工因素等等），而形成較大的殘差。

4.5.2 χ^2 fitting 誤差討論

由於我們計算出的χ^2值相對非常小，可能有以下情況：

1. 誤差估計可能偏大：
   若error bar過大，導致χ^2值過低，即使模型不正確也會看似完美。

2. 資料波動小：
   若每次實驗測得的擺動週期變異不大（數據高度集中），也可能產生偏低的χ^2值。

3. 過度擬合潛在可能：
   擬合模型使用自由參數調整得非常貼合資料點，但未必具有物理意義。此情況下，雖然卡方值低，但模型的泛化能力下降。

可透過以下方法改進：

1. 重新估算error bar：
   採每個擺長點多次重複測量週期，計算平均與標準差作爲error bar。

2. 進行殘差分析：
   檢視實驗值與擬合值之間的差異（殘差），若呈現隨擺長有系統偏移，可能代表模型還有未考慮的物理因素（如空氣阻力、繩長非理想、非簡諧運動等），可據此進行修正。

4.6 方法四：MCMC fitting

```python
# Define the log-likelihood function
def log_likelihood(params, x_data, y_data, sigma):
    model = linear_function(x_data, *params)
    return -chi_sq(y_data, model, sigma)

# Define the log-prior function (flat priors in this case)
def log_prior(params):
    a, b = params
    if -10 < a < 10 and -10 < b < 10:  # Adjust bounds as needed
        return 0.0  # Flat prior
    return -np.inf  # Log(0) for invalid parameters

# Define the log-posterior function
def log_posterior(params, x_data, y_data, sigma):
    lp = log_prior(params)
    if not np.isfinite(lp):
        return -np.inf
    return lp + log_likelihood(params, x_data, y_data, sigma)
```

Figure 27: MCMC程式碼(1)

這部分是利用MCMC（Markov Chain Monte Carlo）取樣，建立對數後驗分佈函數，並定義三個MCMC取樣所需之函數：

1. log-likelihood
2. log-prior
3. log-posterior

首先先定義log-likelihood，參數如下：

- params：模型的參數(a, b)（線性模型之斜率與截距）
- x_data：自變數
- y_data：因變數
- sigma：測量誤差

先使用linear_function(x_data, params)計算模型預測值；接著計算出χ^2，再來由於likelihood function $\mathcal{L}$通常與$e^{\frac{-\chi^2}{2}}$成正比，可取對數得其值。

最後回傳log-likelihood作爲評估參數的可能性。

再來利用Flat Prior設計log-prior，表示我們對參數(a, b)的先驗知識。

最後定義log-posterior，即爲貝葉斯定理。

```
# Set up the MCMC sampler
ndim = 2  # Number of parameters (a and b)
nwalkers = 20  # Number of walkers
nsteps = 10000  # Number of steps
initial_guess = [m_chisq, b_chisq]  # Initial guess for parameters
pos = initial_guess + 1e-4 * np.random.randn(nwalkers, ndim)  # Initialize walkers
```

Figure 28: MCMC程式碼(2)

```
progress_file = "progress.h5"
backend = emcee.backends.HDFBackend(progress_file)

# Initialize the sampler
sampler = emcee.EnsembleSampler(nwalkers, ndim, log_posterior, backend=backend,
                                args=(np.log(pendulum_length_matrix),
                                      np.log(period_1swing.flatten()),
                                      np.log(np.tile(std_1period, 5))))

# Run the MCMC
sampler.run_mcmc(pos, nsteps, progress=True)
```

Figure 29: MCMC程式碼(3)

Fig.28和Fig.29是開始使用MCMC進行參數的擬合。

首先先設定MCMC參數與初始點（兩個參數、20個行走者、每個行走者執行1000步），並在初始化MCMC採樣器後執行。

```python
sampler = emcee.backends.HDFBackend("progress.h5")

# Extract the samples
samples = sampler.get_chain(discard=100, thin=15, flat=True)

# Print the results
m_mcmc, b_mcmc = np.mean(samples, axis=0)
print(f"MCMC results: m = {m_mcmc:.3f}, b = {b_mcmc:.3f}")
```

Figure 30: MCMC程式碼(4)

```python
# Corner plot
fig_corner = corner.corner(samples, labels=['m', 'b'], show_titles=True, plot_datapoints=True, quantiles=[0.16, 0.5, 0.84])
# fig_corner.suptitle("Corner Plot of MCMC Results")
plt.savefig('./figures/corner.pdf', transparent=True)
plt.show()
```

Figure 31: MCMC程式碼(5)

Fig.30和Fig.31目的是讀取MCMC的取樣結果並分析參數的後驗分佈，最後繪製成圖。
提取MCMC採樣點部分的參數解釋：

- $discard = 100$爲丟棄前100步，確保MCMC收斂後樣本才被使用。
- $thin = 15$每15步取一個樣本。
- flat=True將所有行走者的樣本合併成一個長列表。

最後計算MCMC擬合出的最佳參數（斜率m與截距b）。

```python
# Chain-step plot
fig_chain, axes = plt.subplots(ndim, figsize=(10, 7), sharex=True)
for i in range(ndim):
    ax = axes[i]
    ax.plot(sampler.get_chain()[:, :, i], "k", alpha=0.3)
    if i == 0:
        ax.set_ylabel("m")
    else:
        ax.set_ylabel(f"b")
    ax.set_xlim(left=0, right=nsteps)
axes[-1].set_xlabel("Step number")
fig_chain.suptitle("Chain-Step Plot")
plt.savefig('./figures/chain.pdf', transparent=True)
plt.show()
```

Figure 32: MCMC程式碼(6)

此爲診斷MCMC計算後的工具之一（Fig.37）。

```python
plt.scatter(np.log(pendulum_length_matrix), np.log(period_1swing.flatten()), s=50, marker='x', color='silver', label='Experiment')
plt.errorbar(np.log(pendulum_length), np.log(avg_1period), yerr=std_1period_log, fmt='o', label='Avg T', color='k', capsize=5)
# plt.plot(np.log(l_axis), linear_function(np.log(l_axis), *propt), color='orangered',linestyle='dashed',
#          label=f'curve_fit')
plt.plot(np.log(l_axis), np.log(t_ideal), label='Theoretical T', color='teal', linestyle='--')
# plt.plot(np.log(l_axis), np.dot(np.vander(np.log(l_axis), 2), w), "--k", label="LS")
# plt.plot(np.log(l_axis), linear_function(np.log(l_axis), *optimized_params), "--b", label="Chi-squared")
plt.plot(np.log(l_axis), (m_mcmc*np.log(l_axis)+b_mcmc), color='lightcoral',linestyle='-', label=f'MCMC')
plt.xlabel('log(Length) [m]')
plt.ylabel('log(Duration of a swing) [s]')
plt.title('Duration of a swing vs Length')
plt.xlim(left=np.log(0.05), right=np.log(0.65))
plt.ylim(bottom=np.log(0.5), top=np.log(1.7))
plt.legend()
plt.savefig('./figures/mcmc_log.pdf', transparent=True)
plt.show()
```

Figure 33: MCMC程式碼(7)

此段爲MCMC的擬合結果（對數尺度），繪製出對數尺度的實驗數據與MCMC擬合曲線（爲Fig.36右圖）。

```python
plt.scatter(pendulum_length_matrix, period_1swing.flatten(), s=50, marker='x', color='silver', label='Experiment')
plt.errorbar(pendulum_length, avg_1period, yerr=std_1period, fmt='o', label='Avg T', color='k', capsize=5)
# plt.plot(l_axis, np.e**linear_function(np.log(l_axis), *propt), color='orangered',linestyle='dashed',
#          label=f'curve_fit')
plt.plot(l_axis, t_ideal, label='Theoretical T', color='teal', linestyle='--')
# plt.plot(l_axis, np.e**np.dot(np.vander(np.log(l_axis), 2), w), "--k", label="LS")
# plt.plot(l_axis, np.e**linear_function(np.log(l_axis), *optimized_params), "--b", label="Chi-squared")
plt.plot(l_axis, np.e**(m_mcmc*np.log(l_axis)+b_mcmc), color='lightcoral',linestyle='-', label=f'MCMC')
plt.xlabel('Length [m]')
plt.ylabel('Duration of a swing [s]')
plt.title('Duration of a swing vs Length')
plt.xlim(left=0.07, right=0.6)
plt.ylim(bottom=0.5, top=1.6)
plt.legend()
plt.savefig('./figures/mcmc.pdf', transparent=True)
plt.show()
```

Figure 34: MCMC程式碼(8)

此段爲MCMC的擬合結果（線性尺度），繪製出線性尺度的實驗數據與MCMC擬合曲線（爲Fig.36左圖）。

```python
g_fitted = np.e**(2*(np.log(2*np.pi) - b_mcmc))
print(f'Ideal g = {g0.value:.3f} m/s^2')
print(f'Fitted g = {g_fitted:.3f} m/s^2')
print(f'Error = {np.abs(g_fitted - g0.value) / g0.value * 100:.3f}%')
```

Figure 35: MCMC程式碼(9)

最後利用MCMC方法擬合得到$g = 9.887\,m/s^2$，與理論的$g_0 = 9.807\,m/s^2$誤差0.823%。

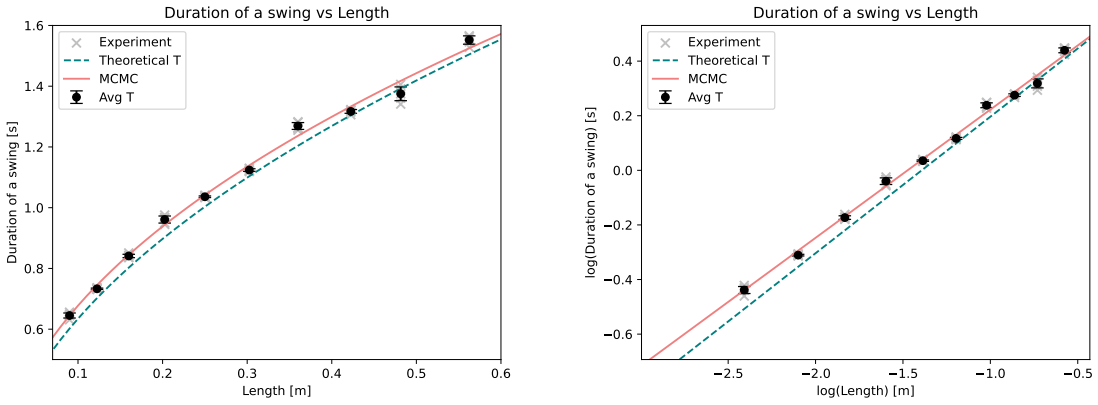

Figure 36: MCMC擬合結果（左圖爲線性擬合；右圖爲對數擬合）

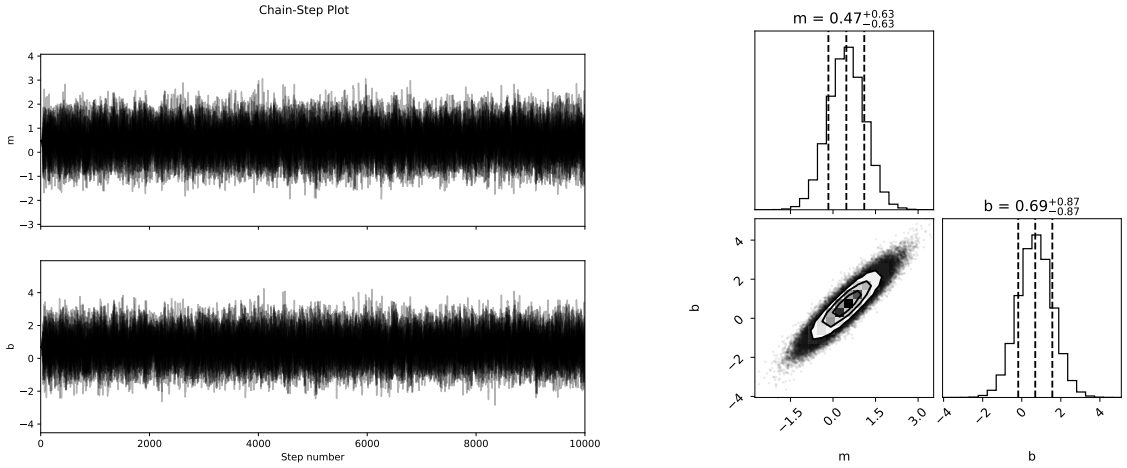

Figure 37: Chain-step plot and posterior corner plot

4.6.1 MCMC fitting 數據分析

如Fig.36所示，我們使用另一種方法–MCMC（馬可夫鏈蒙地卡羅）方法做非線性擬合；左圖爲線性尺度下的擺長與週期關係，右圖則爲對數尺度。

MCMC擬合結果（紅線）整體貼合實驗點，與實驗error bar（黑色）一致性良好；相較於理論曲線（虛線），MCMC 模型更能捕捉到實驗數據的微小偏差，尤其在短擺長處擁有更高的吻合度，可能是因爲誤差在小擺長處較小，因此權重較大，MCMC 更傾向貼近這些資料點。

在對數-對數圖中，MCMC擬合同樣呈現高度線性，與實驗數據的對數趨勢相符，強化了模型合理性。

MCMC不僅給出最佳擬合參數，也能產生參數的後驗分佈，進一步評估其不確定性，使整體擬合更具統計意義，因此我們生成Fig.37來驗證其擬合解果的可靠性。

在左圖中，兩條鏈分別對應參數m與b的歷程變化，顯示出已達到穩定波動狀態，無明顯飄移或趨勢，表示馬可夫鏈已充分混合並收斂，擬合過程穩定。

右圖爲後驗分佈圖，說明參數的不確定性已被資料良好約束。聯合分佈圖中呈現出輕微正相關的橢圓形，亦符合物理模型中擬合參數可能的耦合特性。

4.6.2 MCMC fitting 誤差討論

誤差可能來自：

1. 參數不確定性來源：
   MCMC 本身會輸出參數的機率分佈，若分佈過寬，代表數據不足以嚴格約束模型；反之若過窄，可能暗示模型過於自信，未考慮潛在系統誤差。

2. 模型選擇的依賴性：
   MCMC 依賴於設定的模型形式（如冪次關係），若模型假設本身有偏誤，則即使後驗收斂良好，也可能是「錯模型的好擬合」。

3. error bar可信度影響擬合精度：
   若誤差估計不足（如未反映實驗者反應時間、系統延遲等），會使後驗分佈過窄，導致參數誤判或過度擬合。

改善的方法：

1. 加入參數後驗分佈圖與可信區間：
   展示如a, b的後驗分佈，可更具體呈現參數的不確定性，也能對比理論值是否落在可信區間內。

2. 提升誤差估計的可靠性：
   以每個擺長點進行多次測量（取平均與標準差），減少系統誤差未反映於σ的問題。

4.7 比較不同方法

```
plt.scatter(np.log(pendulum_length_matrix), np.log(period_1swing.flatten()), s=50, marker='x', color='silver', label='Experiment')
plt.errorbar(np.log(pendulum_length), np.log(avg_1period), yerr=std_1period_log, fmt='o', label='Avg T', color='k', capsize=5)
plt.plot(np.log(l_axis), linear_function(np.log(l_axis), *propt), color='orangered',linestyle='dashed',
         label=f'curve_fit')
plt.plot(np.log(l_axis), np.log(t_ideal), label='Theoretical T', color='teal', linestyle='--')
plt.plot(np.log(l_axis), np.dot(np.vander(np.log(l_axis), 2), w), linestyle="--", color='slateblue',label="LS")
plt.plot(np.log(l_axis), linear_function(np.log(l_axis), *optimized_params), linestyle="--", color='goldenrod', label="Chi-squared")
plt.plot(np.log(l_axis), (m_mcmc*np.log(l_axis)+b_mcmc), color='lightcoral',linestyle='--', label=f'MCMC')
plt.xlabel('log(Length) [m]')
plt.ylabel('log(Duration of a swing) [s]')
plt.title('Duration of a swing vs Length')
plt.xlim(left=np.log(0.05), right=np.log(0.65))
plt.ylim(bottom=np.log(0.5), top=np.log(1.7))
plt.legend()
plt.savefig('./figures/all_log.pdf', transparent=True)
plt.show()
```

Figure 38: 比較不同方法程式碼(1)

最後，我們比較每個方法的差異，分別繪製出最小平方法、卡方檢定、MCMC方法的線性擬合結果比較；輸出結果見Fig.41左圖。

```
plt.scatter(pendulum_length_matrix, period_1swing.flatten(), s=50, marker='x', color='silver', label='Experiment')
plt.errorbar(pendulum_length, avg_1period, yerr=std_1period, fmt='o', label='Avg T', color='k', capsize=5)
plt.plot(l_axis, np.e**linear_function(np.log(l_axis), *propt), color='orangered',linestyle='dashed',
         label=f'curve_fit')
plt.plot(l_axis, t_ideal, label='Theoretical T', color='teal', linestyle='--')
plt.plot(l_axis, np.e**np.dot(np.vander(np.log(l_axis), 2), w), linestyle="--", color='slateblue', label="LS")
plt.plot(l_axis, np.e**linear_function(np.log(l_axis), *optimized_params), linestyle="--", color='goldenrod', label="Chi-squared")
plt.plot(l_axis, np.e**(m_mcmc*np.log(l_axis)+b_mcmc), color='lightcoral',linestyle='--', label=f'MCMC')
plt.xlabel('Length [m]')
plt.ylabel('Duration of a swing [s]')
plt.title('Duration of a swing vs Length')
plt.xlim(left=0.07, right=0.6)
plt.ylim(bottom=0.5, top=1.6)
plt.legend()
plt.savefig('./figures/all.pdf', transparent=True)
plt.show()
```

Figure 39: 比較不同方法程式碼(2)

此爲三種方法的對數擬合結果比較；輸出結果見Fig.41右圖。

```
print(f'Ideal g = {g0.value:.3f} m/s^2')
print("-------------------------------------")
print(f'Fitted g (curve_fit) = {g_fitted:.3f} m/s^2')
print(f'Error (curve_fit) = {np.abs(g_fitted - g0.value) / g0.value * 100:.3f}%')
print("-------------------------------------")
print(f'Fitted g (LS) = {np.e**(2*(np.log(2*np.pi) - b_ls)):.3f} m/s^2')
print(f'Error (LS) = {np.abs(np.e**(2*(np.log(2*np.pi) - b_ls)) - g0.value) / g0.value * 100:.3f}%')
print("-------------------------------------")
print(f'Fitted g (Chi-squared) = {np.e**(2*(np.log(2*np.pi) - b_chisq)):.3f} m/s^2')
print(f'Error (Chi-squared) = {np.abs(np.e**(2*(np.log(2*np.pi) - b_chisq)) - g0.value) / g0.value * 100:.3f}%')
print("-------------------------------------")
print(f'Fitted g (MCMC) = {np.e**(2*(np.log(2*np.pi) - b_mcmc)):.3f} m/s^2')
print(f'Error (MCMC) = {np.abs(np.e**(2*(np.log(2*np.pi) - b_mcmc)) - g0.value) / g0.value * 100:.3f}%')
```

Figure 40: 比較不同方法程式碼(3)

此爲三種方法推論出之重力加速度值。輸出得：

- Ideal $g_0 = 9.807\,m/s^2$
- $g_{(\texttt{curve\_fit})} = 9.807\,m/s^2$
  $Error_{(\texttt{curve\_fit})} = 0.823\%$
- $g_{(LS)} = 9.969\,m/s^2$
  $Error_{(LS)} = 1.660\%$
- $g_{(Chi-squared)} = 9.969\,m/s^2$
  $Error_{(Chi-squared)} = 1.660\%$
- $g_{(MCMC)} = 9.887\,m/s^2$
  $Error_{(MCMC)} = 0.823\%$

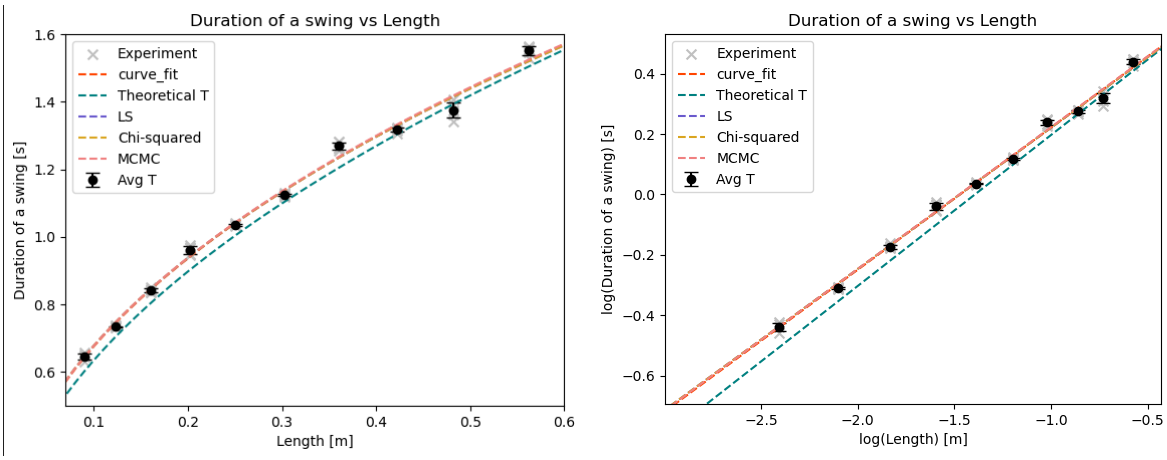

Figure 41: 各方法比較之結果；左圖爲線性擬合結果；右圖爲對數擬合結果

4.7.1 比較所有結果－數據分析

見Fig.41，結果顯示各方法的擬合曲線接與平均值非常接近，尤其是curve_fit和MCMC方法。右圖採用對數-對數圖表示資料和理論的線性關係，我們做出的圖顯示實驗資料大致服從$\log(T) \propto \frac{1}{2}\log(L)$的關係，理論模型有良好的預測性。

而因爲我們在χ^2 fitting雖有加權，但因爲數據的標準差差異不大，因此在實作上幾乎等同於未加權最小平方法。

4.7.2 比較所有結果－誤差討論

計算出之重力加速度誤差皆小於2%，我們認爲在可接受範圍內；LS和χ^2值方法誤差較大，可能是因爲在擬合過程中未考慮各資料點的不確定性（即週期的標準差），等於假設所有資料點的誤差相同，可能導致擬合結果對誤差較大的資料點過度信任，進而影響最終的斜率與g值；反之，curve_fit與MCMC方法納入誤差加權機制，能更客觀反映實驗數據的實際變異性，因此其誤差相對較小。

改進方法：

1. 誤差加權擬合：
   使用如 weighted least squares 或加權卡方分析，可更充分反映實驗誤差並提升擬合準確度。

2. 增加測量次數：
   對每個擺長進行更多次的測量，有助於提升統計可靠度。

實驗產生之系統誤差和隨機誤差，由於本次我們沒有實際做實驗，因而無法排除。

5 問題討論

5.1 根據實驗數據的擬合結果所計算出的χ^2值是多少？如何利用χ^2值判定擬合的好壞？如果值偏差較大，可能有哪些原因？

1. 我們透過程式碼所計算出的χ^2值爲：0.15251356436613928
2. 參考預習問題第四題及eq.(16)，χ^2表示的是測量值與預測值的加權平方差，誤差σ_i越小，偏差造成的影響越大；而應當透過自由度來分析使用χ^2 fitting擬合的好壞。
   自由度定義爲：衡量進行統計估計時有多少獨立的數據點或參數。
   數學表達：
   $$\nu = N - p \tag{11}$$
   其中ν爲自由度；N爲資料點之數量；p爲擬合模型中參數的數量。 比較每個自由度對應的χ^2值：
   $$\chi^2{}_{reduced} = \frac{\chi^2}{\nu} = \frac{\chi^2}{N-p} \tag{12}$$
   這樣可以去除不同資料量或擬合參數量的影響，更直觀的表示出來
   - 理想情況：
     在理想狀態下，每個自由度對應的卡方值會接近1，表示實驗值與觀察值間的差異和誤差範圍相應，無過多偏差。
   - 過度擬合：
     若每個自由度對應的 χ^2 值顯著小於1，表示模型過度擬合資料，模型過於複雜，亦可能對資料的雜訊進行擬合，進而導致過度擬合的情況。
   - 欠擬合：
     若每個自由度對應的 χ^2 值遠大於1，表示模型無法適配欲擬合之資料，可能原因爲模型過於簡單，或數據的變化沒被捕捉到。
3. 若χ^2值偏差較大，可能的原因如下：
   (a) 模型選擇不合適
   (b) 誤差範圍估算錯誤
   (c) 資料本身雜訊過多

5.2 在數據擬合過程中,測量誤差 (error bar) 的值會如何影響χ^2值和最終擬合的重力加速度g？

1. 同樣參考eq.(16)，σ_i即表示數據的測量誤差，由此公式可見，σ_i會直接影響每個數據點對χ^2的貢獻；當誤差較小（也就是σ_i較小）時，術據點之偏差（$y_i - f(x_i)$）在χ^2中的權重會變大；反之誤差變大時，此偏差將對χ^2影響較小，導致整體的χ^2更容易變小。
2. 對最終擬合結果（重力加速度g）的影響：
   - 當誤差較小時，數據點的偏差會在擬合過程會得到更大影響，模型將更顯著的調整參數，以減少誤差；使得最終擬合結果更爲精確。
   - 當誤差較大時，數據點的偏差對擬合的影響較小，擬合過程會更仰賴於較大的誤差範圍，使最終擬合結果可能產生較大的不確定性。

5.3 殘差圖是否呈現隨機分佈？是否觀察到某種趨勢（例如隨擺長增加或減少）？如果存在系統性誤差，可能的原因是什麼？如何改進實驗設計來減少這類誤差？

根據殘差圖Fig.26顯示，殘差接分布於殘差值= 0之附近，因此無法從殘差圖中觀察並推斷出殘差具有任何趨勢關係。因此我們可以暫時推論殘差屬於機率分布，與擺長並沒有關聯。若要驗證推論，則可以透過實際實驗取得數據，確認不同擺長下的殘差圖是否依舊符合隨機分布。

系統性誤差可能來源：

1. 單擺擺動時可能受到空氣阻力影響。
   解決方式：在眞空環境下實驗。

2. 量測誤差。
   解決方式：使用精確度更高的量測工具，包含更準確的計時器、量角器等等，或是透過機器、電腦操作計時和調整單擺擺放角度，避免人爲操作產生之誤差。

5.4 χ^2擬合的結果與單純使用最小二乘法（least square fitting）擬合相比有何不同？χ^2擬合的優勢在哪裡？適合用在什麼實驗？

最小二乘法 (least square fitting)的核心概念是找出一條線，使得數據點與這條線垂直距離的平方和能達到最小值，即是殘差平方總和的最小化。

最小二乘法假設： 誤差是隨機分布、正態分布，並且每個點的誤差大小皆一樣大（稱爲等變異性）、資料點之間相互獨立。最小二乘法較明顯的問題是其對異常值較敏感，因此若是數據之誤差值較大，其擬合結果可能偏差。

而χ^2擬合則會考慮到測量誤差，如果數據點內的量測誤差大小不同，使用χ^2擬合可確保誤差較大之數值不會對整體數據造成太明顯的影響。χ^2值大表示觀察到的結果與期望值有較大差異，不確定度的較大數據之殘差會降低對χ^2值的貢獻，反之不確定不小的數據之殘差則會提高對χ^2值的貢獻。若測量誤差相同，χ^2擬合結果會等於最小二乘法。

同時χ^2擬合可以透過檢測P值確認擬合的可信度。故χ^2擬合適用於誤差來源較多的實驗，如單擺實驗或者光譜實驗，透過使用χ^2擬合，可以較有效地降低誤差帶來的干擾。

6 總結

本實驗以驗證單擺週期與擺長之經典關係爲目標，並利用 Chi-square 擬合方法分析實驗數據，進而反推出重力加速度值。Chi-square 擬合的一大優點爲其可根據資料的誤差給予不同權重，使誤差較小的資料點對模型更具影響力。在本次實作中，我們透過最小化卡方統計量，求得擬合參數$m = 0.4802$、$b = 0.7927$，計算得$g = 9.969\,m/s^2$，與理論值$g = 9.807\,m/s^2$相比誤差爲 1.66%。然而，$\chi^2 = 0.1525$ 遠小於自由度（8），顯示模型或誤差估計可能存在過度樂觀的情形。從殘差圖觀察，資料點大致隨機分佈，無明顯系統性誤差，顯示擬合結果具統計合理性。此外，我們亦將Chi-square 擬合與其他方法（如最小平方法、`curve_fit`與MCMC）進行比較，發現考慮誤差加權的擬合方法（如 `curve_fit`與MCMC）在精確度與穩定性上表現更佳，凸顯了誤差估計與模型選擇在實驗分析中的重要性。

7 分工

洪懌平：coding、前言
黃巧涵：`curve_fit`、LS數據分析及誤差討論、問題三四
洪瑜：χ^2 fitting、MCMC數據分析及誤差討論、問題一二