Experimental Physics (II)
Experiment Notebook

Fundamental Python
Fitting Data of Simple Pendulum

Group 2

洪　瑜 B125090009

黃巧涵 B122030003

洪懌平 B102030019

2025/04/01

1 Preparatory Question

1.1 Under the small-angle approximation, why can the relation between period T and pendulum length L be expressed as $T = 2\pi\sqrt{L/g}$? Please explain the derivation process of this formula.

Using Newton's Second Law and Lagrangian Equation:

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

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

Suppose $\theta \sim 0$, i.e., under the small-angle approximation, $sin\theta \sim \theta$, then Eq. 2 can be simplified to:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\theta = 0 \tag{3}$$

可視爲簡諧運動的微分方程

$$\theta(t) = Asin(\omega t) + Bcos(\omega t) \tag{4}$$

where,

$$\omega = \sqrt{\frac{g}{L}} \tag{5}$$

$$T = \frac{2\pi}{\omega} = 2\pi\sqrt{\frac{L}{g}} \tag{6}$$

T爲系統完成一次完整振動所需的時間，即爲週期

1.2 Through logarithmic transformation, the relationship between the period of a simple pendulum and its length can be converted into a linear relationship. Write out this transformation process and explain why such a transformation can help us use linear fitting to determine the gravitational acceleration g.

將Eq.6兩邊取對數：

$$\ln T = \ln(2\pi) + \frac{1}{2}\ln g \tag{7}$$

令

$$y = \ln T, \quad x = \ln L \tag{8}$$

$$A = \ln(2\pi) - \frac{1}{2}\ln g \tag{9}$$

可將Eq.9改寫爲：

$$y = \frac{1}{2}x + A \tag{10}$$

斜率：

$$m = \frac{1}{2} \tag{11}$$

可將實驗數據用線性回歸求斜率，而

$$A = \ln(2\pi) - \frac{1}{2}\ln g \tag{12}$$

$$\ln g = 2(\ln(2\pi) - A) \tag{13}$$

擬合後的截距A可求重力加速度g：

$$g = exp\left(2\ln(2\pi) - 2A\right) \tag{14}$$

1.3 What is the basic principle of least square fitting? Is **`curve_fit`** initially based on least square fitting?

1. 最小二乘法是用來找到一組最佳參數使模型$f(x)$與觀測數據(x_i,y_i)的偏差最小化。 概念爲殘差平方和

$$S = \sum_i (y_i - f(x_i))^2 \tag{15}$$

y_i爲測量值、$f(x_i)$爲模型預測值、$y_i - f(x_i)$爲殘差。將模型參數優化使S最小，可找出最佳擬合結果。

2. 是的，`scipy.optimize.curve_fit`預設用非線性最小二乘法擬合數據，找最佳參數，使殘差平方最小；若數據有誤差，可透過`sigma`參數指定權重，`curve_fit`改用加權最小二乘法。

1.4 What is the basic principle of chi-square (χ^2) fitting? How does it differ from least square fitting?

1. 用來最小化卡方統計量χ^2

$$\chi^2 = \sum_i \frac{(y_i - f(x_i))^2}{\sigma_i^2} \tag{16}$$

y_i爲測量值、$f(x_i)$爲模型預測值、σ_i爲數據點的標準差。 這個方法適用當數據有不同測量誤差時，因爲它會給誤差較小的數據較高的權重，誤差較大的數據較低的權重。

2. 最小二乘法假設所有數據點的測量誤差相同，不考慮個別點的不確定性。 χ^2擬合考慮測量誤差，誤差小的數據點影響較大，誤差大的數據點影響較小。 若測量誤差相同，χ^2擬合等於最小二乘法。

1.5 How to estimate χ^2? Please write the formula and explain the meaning of each variable.

$$\chi^2 = \sum_i \frac{(y_i - f(x_i))^2}{\sigma_i^2} \tag{17}$$

χ^2爲卡方統計量（衡量擬合優劣的指標）；y_i爲第i個數據點的實驗測量值；$f(x_i)$爲第i個數據點的模型預測值；σ_i爲第i個數據點的測量誤差；N爲數據點總數。計算殘差$y_i - f(x_i)$並平方，再除以誤差平方（σ_i^2），加總全部即爲χ^2

意義：

1. χ^2值越小，擬合效果越好
2. χ^2接近N，則模型擬合合理
3. $\chi^2 \gg N$，表模型和數據不符
4. $\chi^2 \ll N$，可能誤差過大或過度擬合

1.6 What are the two general types of errors? Which type of error can be reduced by repeatedly conducting experiments

1. 系統誤差：可預測且固定的誤差。
   (a) 儀器誤差：測量設備本身的限制或校準問題
   (b) 方法誤差：測量方式的限制
   (c) 環境誤差：外在因素的影響
2. 隨機誤差：不可預測且隨機變動的誤差。
   (a) 讀數誤差：測量值因主觀判讀不同
   (b) 外界干擾：例如空氣流動影響擺動時間測量
   (c) 儀器靈敏度的限制：測量儀器的解析度有限，導致測量值有微小變動

隨機誤差可以透過多次實驗取平均值來降低影響，因爲它的變動是隨機的，多次測量後平均值會更接近真實情況；但系統誤差無法透過多次實驗消除，需要透過改進儀器、調整測量方向，或校正數據來修正。

2 Experimental steps, preliminary results, and analysis

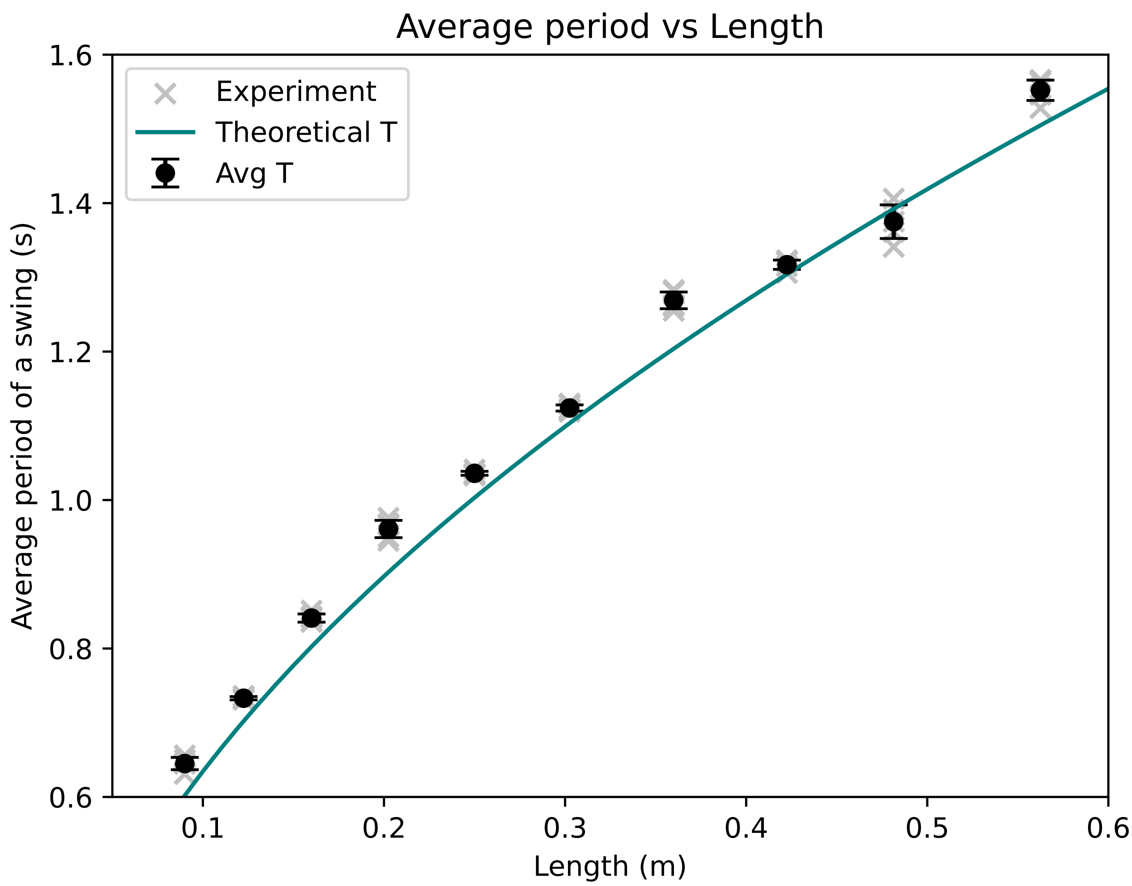

Figure 1: Experiment data

2.1 curve_fit

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

Figure 2: curve_fit(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 3: curve_fit(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.6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 4: curve_fit(3)

繪製原始尺度的時間 vs. 擺長圖，已轉爲線性尺度（fig.6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 5: curve_fit(4)

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

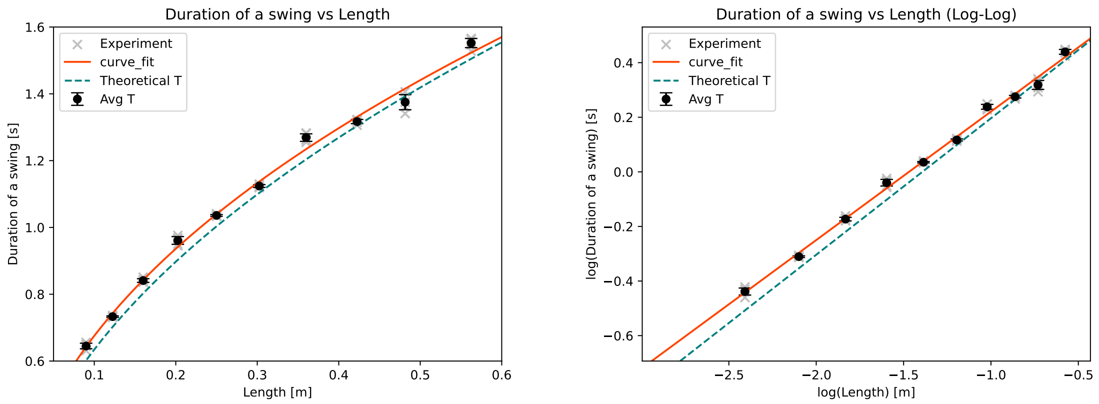

Figure 6: `curve_fit`

2.2 Least-squared 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 7: LS(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 8: LS(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.11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 9: LS(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.11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 10: LS(4)

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

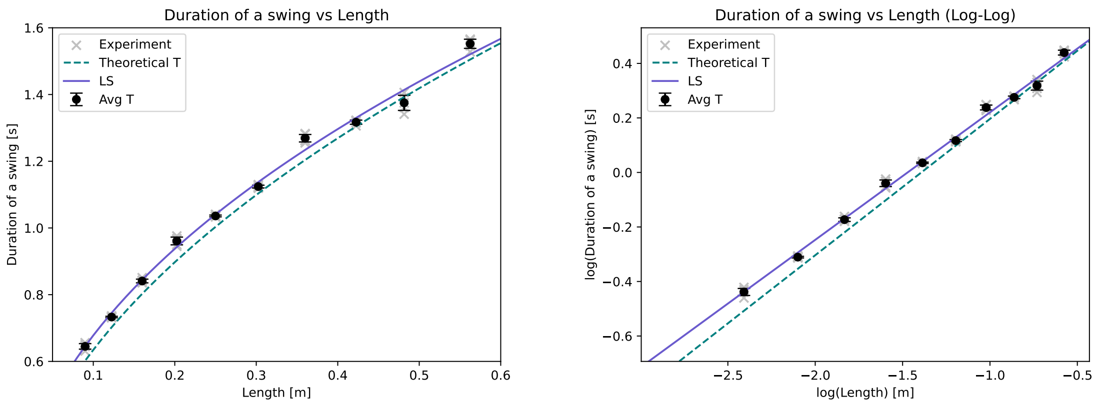

Figure 11: Least-squared fitting

2.3 χ^2 fitting

```
# 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 12: χ^2 fitting(1)

這部分是先定義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a=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a、b顯示出來。

```
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 13: χ^2 fitting(2)

這部分使用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.16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), 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 14: χ^2 fitting(3)

這部分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.16左圖）。

```python
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 15: χ^2 fitting(4)

這部分透過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$g = 9.969(\frac{m}{s^2})$，與理論的$g_0 = 9.807\frac{m}{s^2}$誤差1.660%。

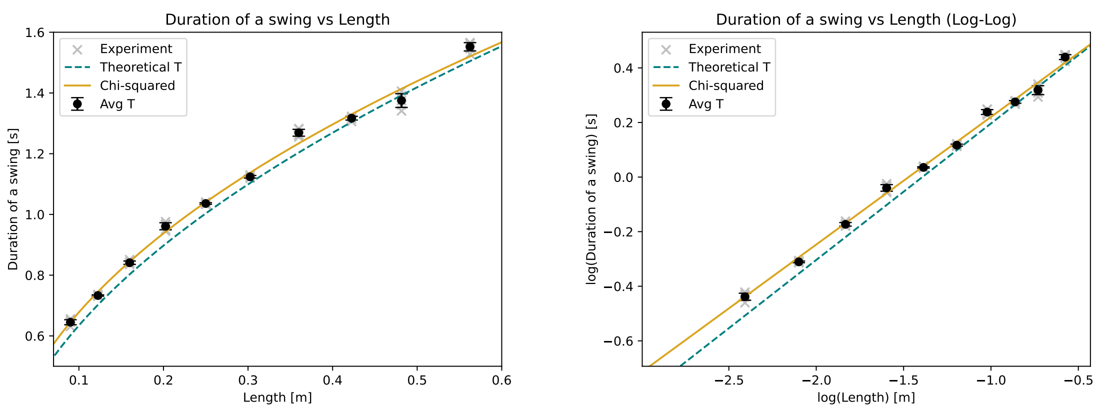

Figure 16: χ^2 fitting

2.4 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 17: MCMC(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。

```python
# 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 18: MCMC(2)

```python
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 19: MCMC(3)

fig.18和fig.19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採樣氣後執行。

```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 20: MCMC(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 21: MCMC(5)

fig.20和fig.21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）。

此爲診斷MCMC計算後的工具之一（fig.22）。

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.26右圖）。

```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 22: MCMC(6)

```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 23: MCMC(7)

```
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 24: MCMC(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.26左圖）。

```
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 25: MCMC(9)

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

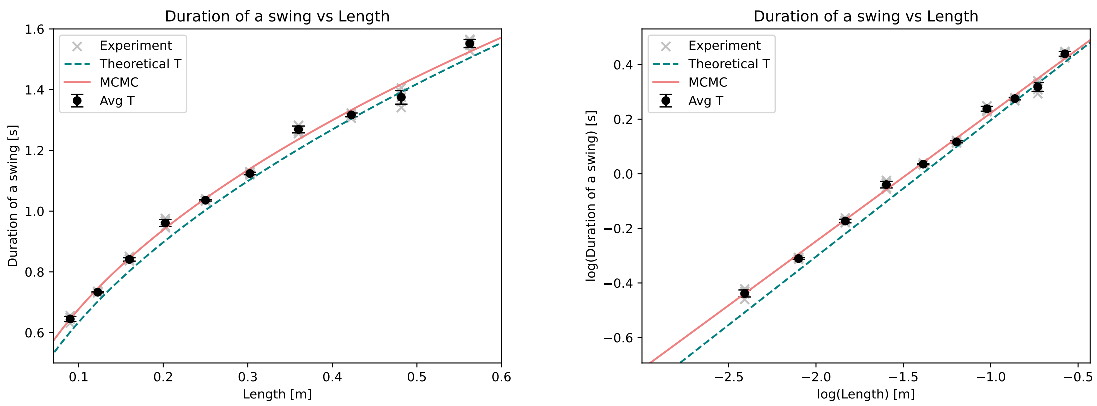

Figure 26: MCMC

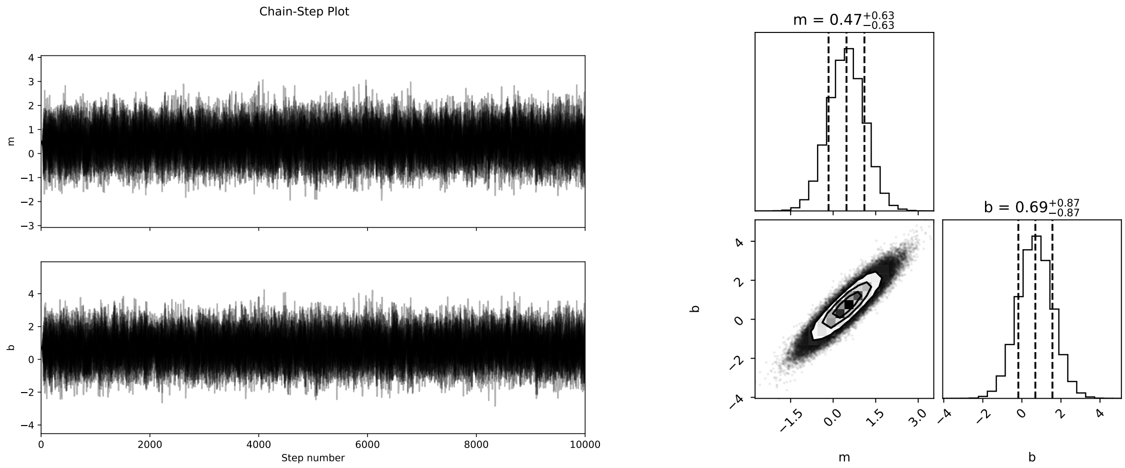

Figure 27: Chain-step plot and posterior corner plot

2.5 Comparison of all methods

```
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 28: Comparison of all methods(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。

```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), 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 29: Comparison of all methods(2)

此爲三種方法的對數擬合結果比較。

```python
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 30: Comparison of all methods(3)

此爲三種方法推論出之重力加速度值。計算得：

- Ideal $g = 9.807 m/s^2$
- $g(curve\_fit) = 9.807 m/s^2$
  $Error(curve\_fit) = 0.823\%$
- $g(LS) = 9.969 m/s^2$
  $Error(LS) = 1.660\%$
- $g(Chi - squared) = 9.969 m/s^2$
  $Error(Chi - squared) = 1.660\%$
- $g(MCMC) = 9.887 m/s^2$
  $(MCMC) = 0.823\%$

初步分析：

1. 方法比較：
   curve_fit和MCMC兩種方法的擬合結果是9.887m/s，誤差 0.823%，而LS和 Chi-squared兩種方法的擬合結果是9.969m/s，誤差1.660%。

2. 擬合誤差分析：
   curve_fit和MCMC方法給出的誤差較小（0.823%），說明這兩種方法較爲精確，且與理論值的差異較小；LS 和 Chi-squared 方法的擬合誤差較大（1.660%），可能性爲這兩種方法在這個問題中可能不如curve_fit或 MCMC方法適用，亦或是這些方法的假設或過程導致了稍微較大的誤差。

3. 初步結論：
   curve_fit和MCMC擬合結果一致，且與理論值接近，說明它們可能能夠更準確地捕捉系統的行爲。

4. 改進的可能性：
   可以嘗試調整LS和Chi-squared方法的參數，或者檢查數據是否符合這些方法的假設。

3 具體說明除错方法、可能遇到的問題

1. curve_fit 擬合
   問題：初始參數敏感，容易發散或收斂至局部極值。
   解決：提供合理的初始猜測值（可先手動擬合估算）；使用全局優化避免局部極值；檢查協方差矩陣評估參數不確定性。

2. 卡方計算
   問題：數據誤差估計不準確，導致卡方值失眞。
   解決：確保誤差來源（如測量誤差、系統誤差）正確建模，或使用誤差重標定（error rescaling）。

3. 最小二乘法
   問題：對異常値敏感，擬合結果可能偏差。
   解決：使用加權最小二乘法或魯棒回歸（如RANSAC）降低異常值影響。

4. MCMC擬合
   問題：收斂速度慢或陷入局部極值。
   解決：調整步長、預燒期（burn-in），或改用更高效的採樣算法（如NUTS）。

5. 共通問題

   (a) 模型偏差（如小角度近似不適用大擺角）：
       解決：改用 精確模型（如橢圓積分）或引入非線性修正項。
   (b) 數值不穩定性（如除零、溢出）：
       解決：對參數施加物理限制，或使用數值穩健算法。
   (c) 過擬合：
       解決：交叉驗證、使用正則化。