實驗物理學（二）
實驗日誌

Fundamental Python
Chi-square fitting - 2

Group 2

洪　瑜 B125090009
黃巧涵 B122030003
洪懌平 B102030019

2025/05/06

1 實驗步驟、初步結果

1.1 Practice 1

1. Generate 2000 sets of data points defined by a function $y = a * x + b +$ noise with $x =$ `np.linspace`$(1, 10, 10)$.
2. The noise is sampled from a normal distribution with mean $= 0$ and standard deviation $= \sigma_0 = 3$.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.stats import chi2
```

```
#設定參數
#自己設定
true_a = 2.0
true_b = 1.0
#題目要求
sigma0 = 3.0
x = np.linspace(1, 10, 10)
num_sets = 2000
```

Figure 1: practice 1-程式碼(1)

3. Fit the data points set by set with the function $y = a * x + b$ using `curve_fit`.

```
#擬合函數
def linear_func(x, a, b):
    return a * x + b

#儲存變數
a_list = []
b_list = []
fit_results = []
all_data = []
chi2_values = []
```

Figure 2: practice 1-程式碼(2)

4. Now calculate the χ^2 for each set of data points. The standard deviation σ_j may vary slightly but remains close to σ_0.

5. $$\chi_i^2 = \sum_{j=1}^{10} \frac{(\text{data\_point}_{i,j} - \text{fitted\_result}_{i,j})^2}{\sigma_j^2} \quad (i = 1 \sim 2000)$$

   where i is the index of data set and j is the index of x value in one data set. ($\text{len}(\chi^2) = 2000$)

```python
#產生資料&擬合
for i in range(num_sets):
    noise = np.random.normal(0, sigma0, size=len(x))
    y = true_a * x + true_b + noise
    all_data.append(y)

    popt, _ = curve_fit(linear_func, x, y)
    fit_y = linear_func(x, *popt)
    fit_results.append(popt)
    #為practice 3做準備
    a_list.append(popt[0])
    b_list.append(popt[1])
    #計算
    residuals = y - fit_y
    chi2_i = np.sum((residuals / sigma0)**2)
    chi2_values.append(chi2_i)

    # 印出全部結果
    print(f"第 {i+1} 組：a = {popt[0]:.4f}, b = {popt[1]:.4f}, χ² = {chi2_i:.4f}")
```

Figure 3: practice 1-程式碼(3)

6. You might find out that this χ^2 is just the sum of the square of the residuals divided by the standard deviation, and it is analogous to previous practice: summing up the χ^2 of each normal distribution sample.

7. Plot all data points you have generated by $y = a * x + b + \text{noise}$.

```python
# 畫出資料點
plt.figure(figsize=(10, 5))
for i in range(100):
    plt.plot(x, all_data[i], 'o', color='gray', alpha=0.3)
plt.title(' y = ax + b + noise')
plt.xlabel('x')
plt.ylabel('y')
plt.savefig("output1_1.pdf", transparent=True)
plt.show()
```

Figure 4: practice 1-程式碼(4)

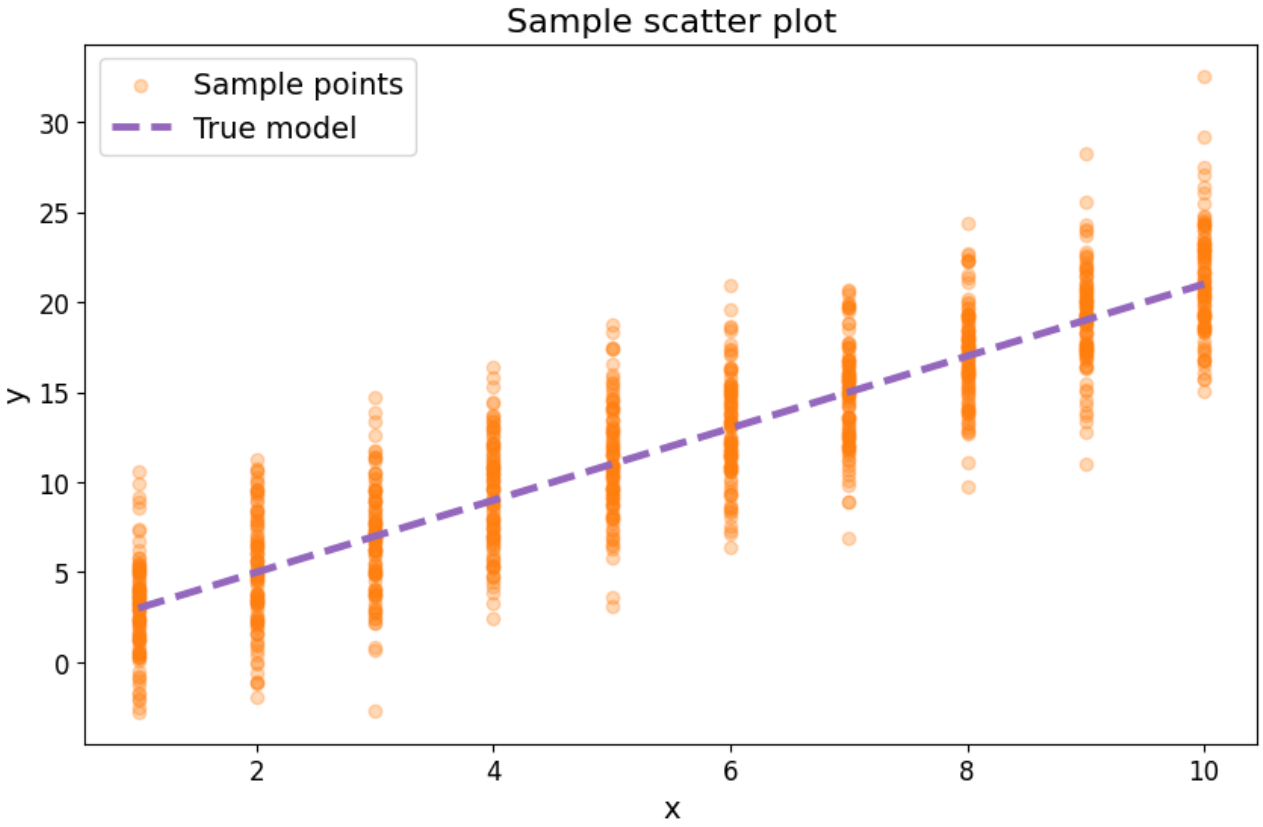

Figure 5: practice 1-output(1)

8. Plot the histogram of these χ^2 obtained from each set of data points. (A set of data points is a 10 points line with noise)

```
#chi square直方圖
plt.figure(figsize=(8, 5))
plt.hist(chi2_values, bins=50, density=True, alpha=0.7, label='χ²')
x_chi = np.linspace(0, 50, 500)
plt.plot(x_chi, chi2.pdf(x_chi, df=8), 'r-', label='degree of freedom=8')
plt.title('DOF=8')
plt.xlabel('chi square')
plt.ylabel('PDF')
plt.legend()
plt.savefig("output1_2.pdf", transparent=True)
plt.show()
```

Figure 6: practice 1-程式碼(5)

9. Compare the histogram with the chi-square distribution.

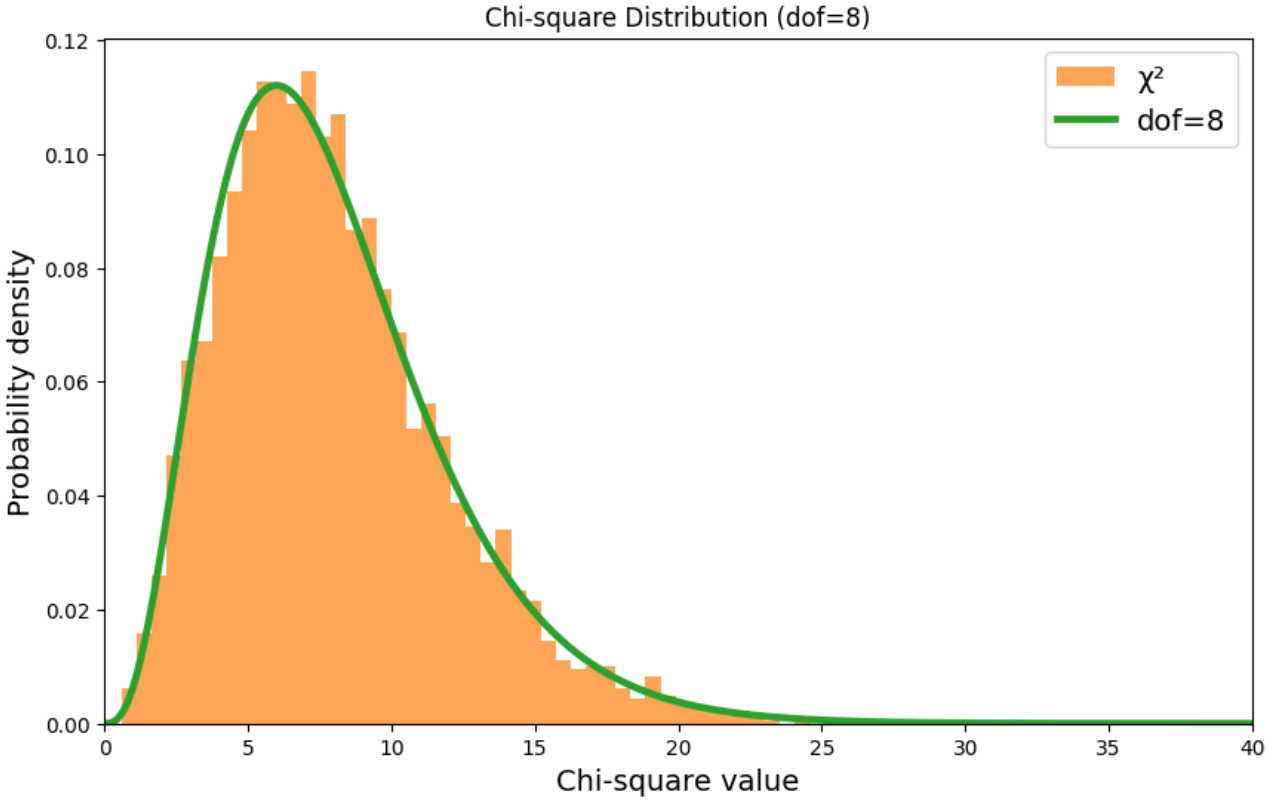

Figure 7: practice 1-output(2)

1.2 Practice 2

1. Show the histogram of a and b in **Practice 1** fitted in separated five sections shown in Figure 1.

```
#畫出每一區的a和b直方圖
fig, axs = plt.subplots(2, 5, figsize=(20, 8))

for i in range(5):
    axs[0, i].hist(a_groups[i], bins=20, alpha=0.7, color='skyblue')
    axs[0, i].axvline(true_a, color='r', linestyle='--', label='True a')
    axs[0, i].set_title(f'Group {i+1} (a)')
    axs[0, i].set_xlabel('a')
    axs[0, i].set_ylabel('Count')

    axs[1, i].hist(b_groups[i], bins=20, alpha=0.7, color='lightgreen')
    axs[1, i].axvline(true_b, color='r', linestyle='--', label='True b')
    axs[1, i].set_title(f'Group {i+1} (b)')
    axs[1, i].set_xlabel('b')
    axs[1, i].set_ylabel('Count')


plt.tight_layout()
plt.savefig("output2_1.pdf", transparent=True)
#plt.axvline(np.mean(a_groups[i]), color='orange', linestyle='--', label='Mean a')
plt.show()
```

Figure 8: practice 2-程式碼(1)

2. Each section contains 400 sets of data points, grouped as follows: 1–400, 401–800, 801–1200, 1201–1600, and 1601–2000, ordered by χ^2 value.

3. Remember that the dataset or the fitted result is not sorted by the χ^2 value at the beginning.

```
#假設chi2_values和fit_results是前面已算好的list
chi2_values = np.array(chi2_values)
fit_results = np.array(fit_results)

#根據chi square值排序，取得排序後的索引
sorted_indices = np.argsort(chi2_values)

#將排序後的擬合參數依區段分成五區（每區400組）
num_per_group = 400
a_groups = []
b_groups = []

for i in range(5):
    #取出這一區的index
    idx = sorted_indices[i * num_per_group : (i + 1) * num_per_group]
    a_groups.append(fit_results[idx, 0])  #取出a值
    b_groups.append(fit_results[idx, 1])  #取出b值
```

Figure 9: practice 2-程式碼(2)

4. You'll need to get those a and b values sorted by the χ^2 value and choose certain indexes that are in the section to plot the histogram.

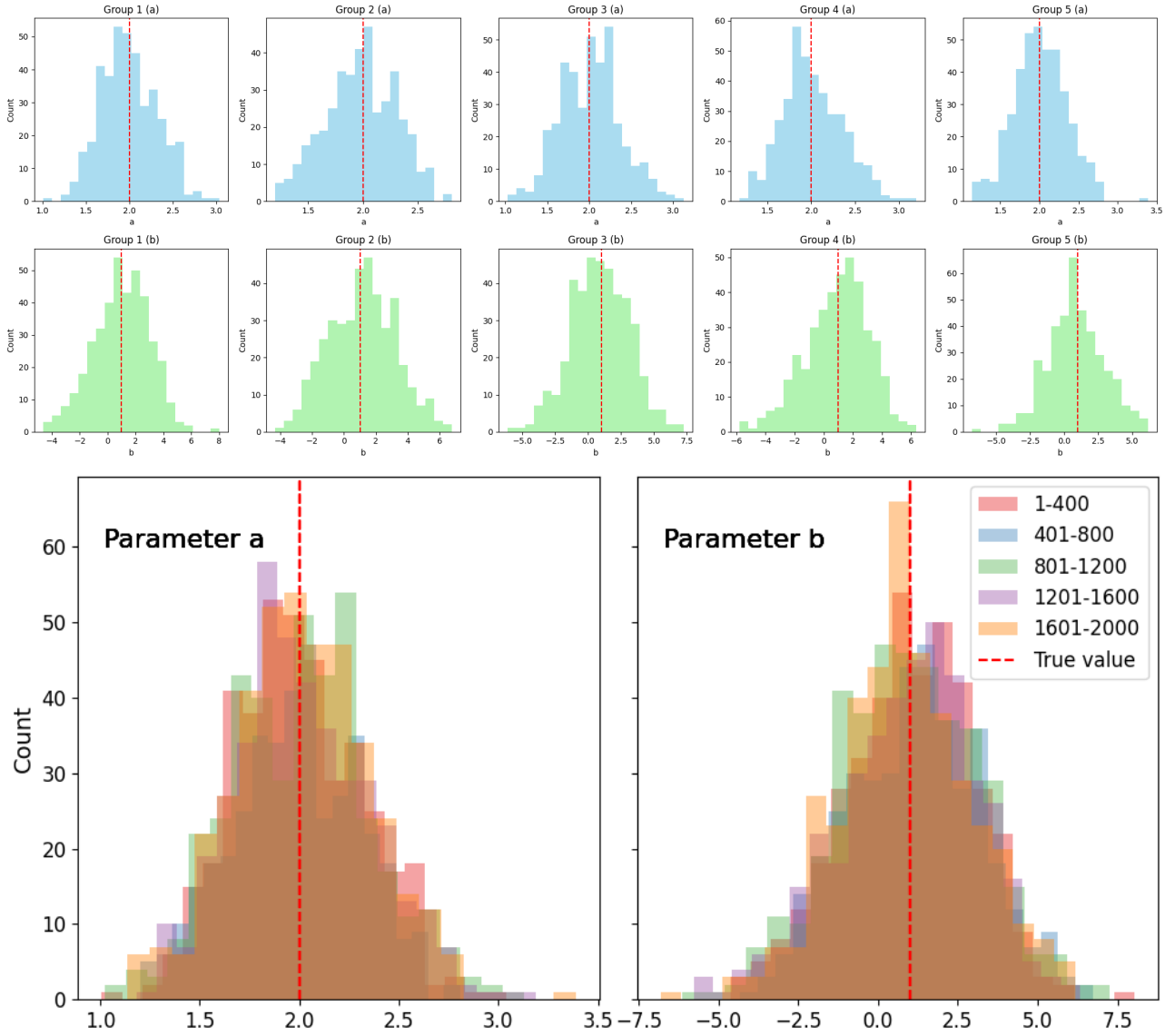

Figure 10: practice 2-output(1)

1.3 Practice 3

1. By using the data points generated in **Practice 1**, get the mean and standard deviation of these data points.

2. Fit these mean value of y with the function $y = a*x+b$ using `curve_fit`, and remember to set `absolute_sigma=True`.

```
a_list = np.array(a_list)
b_list = np.array(b_list)
```

```
#對每個x值，計算y的平均與標準差
y_all = np.array(all_data)  #shape = (2000, 10)
y_mean = np.mean(y_all, axis=0)  #對2000組取平均
y_std = np.std(y_all, axis=0, ddof=1)  #標準差

#x軸（每組共用）
x = np.linspace(1, 10, 10)

#使用y_mean擬合線性模型
def linear_model(x, a, b):
    return a * x + b
```

```
params, cov_matrix = curve_fit(linear_model, x, y_mean, sigma=y_std, absolute_sigma=True)
a_fit, b_fit = params

print(f"線性擬合參數: a = {a_fit:.3f}, b = {b_fit:.3f}")
print("共變異數矩陣:")
print(cov_matrix)
#[0, 0] represents the variance of a (slope)
#[1, 1] represents the variance of b (intercept)
#[0, 1] or [1, 0] represents the covariance between a and b, indicating how changes in a relate to changes in b.
#This indicates that when slope a increases, intercept b tends to decrease – a typical compensation relationship between linear parameters.
```

Figure 11: practice 3-程式碼(1)

3. By using the result in **Practice 1**, you can show the histogram of those a and b values via `plt.hist` and `plt.hist2d` methods. (2000 sets of data points)

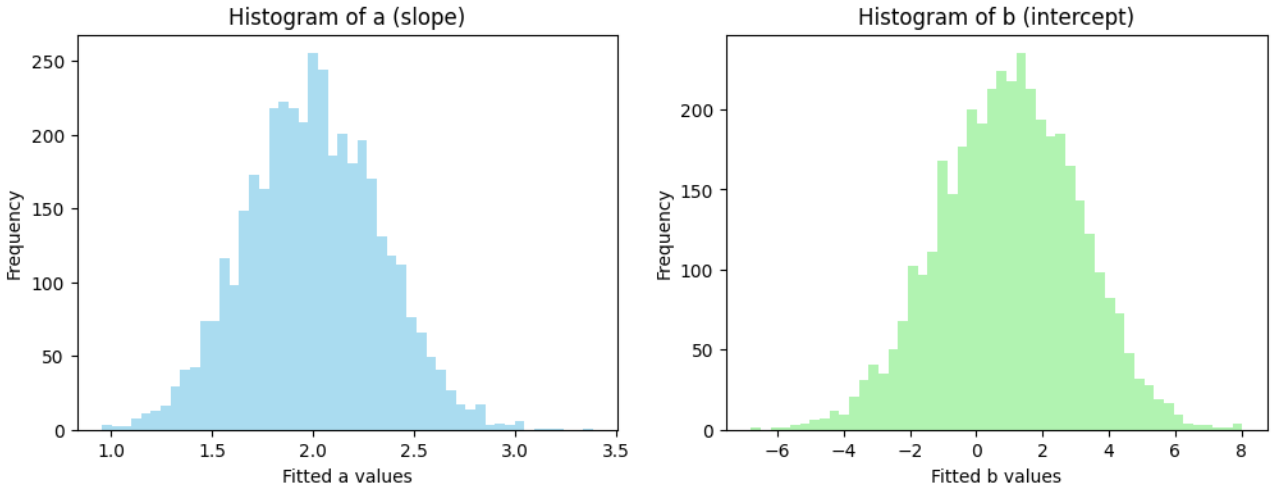

Figure 12: practice 3-output(1)

4. Compare the standard deviation and correlation coefficient obtained from the fitted covariance matrix with those derived from statistical graphs of a and b.

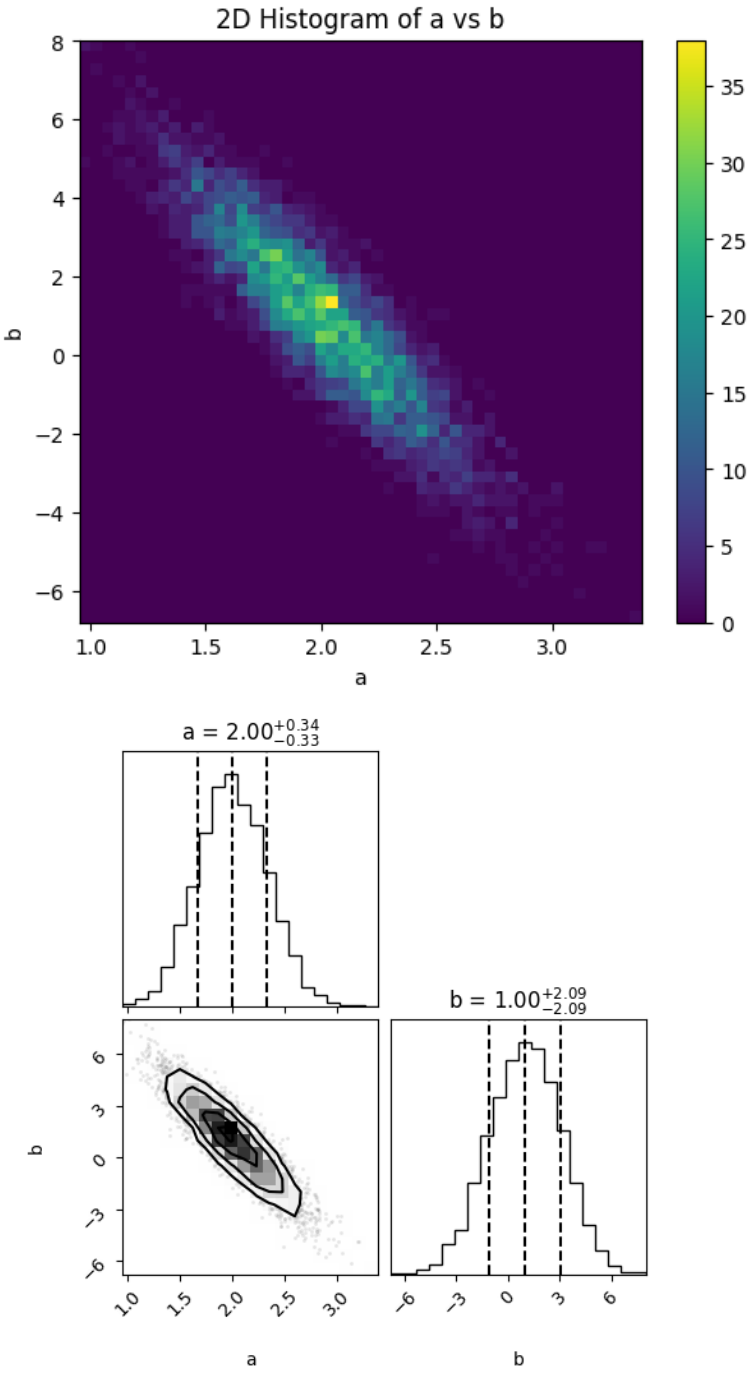

Figure 13: practice 3-output(2)

5. Perform a linear fit on the 2D histogram of parameters a and b to obtain the slope, which characterizes specific properties helpful in calculating the correlation coefficient.

6. Ensure the definition of those quantities you are calculating is correct.

```python
#繪製a與b的直方圖與2D直方圖
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.hist(a_list, bins=50, alpha=0.7, color='blue', edgecolor='black')
plt.xlabel("Fitted a values")
plt.ylabel("Frequency")
plt.title("Histogram of a (slope)")
#plt.savefig("output.3_1.pdf", transparent=True)

plt.subplot(1, 2, 2)
plt.hist(b_list, bins=50, alpha=0.7, color='green', edgecolor='black')
plt.xlabel("Fitted b values")
plt.ylabel("Frequency")
plt.title("Histogram of b (intercept)")
plt.savefig("output.3_1.pdf", transparent=True)
plt.show()
```

```python
#計算共變異數與相關係數
cov_ab = np.cov(a_list, b_list)
corr_ab = np.corrcoef(a_list, b_list)

print("Covariance matrix of [a, b]:")
print(cov_ab)
print("Correlation coefficient matrix of [a, b]:")
print(corr_ab)

#擬合a vs b的關係線，觀察是否線性
from scipy.stats import linregress
slope, intercept, r_value, p_value, std_err = linregress(a_list, b_list)

print(f"Linear fit b = m * a + c:\nSlope = {slope:.3f}, Intercept = {intercept:.3f}")
print(f"Correlation coefficient (r) = {r_value:.3f}, R² = {r_value**2:.3f}")
```

Figure 14: practice 3-程式碼(2)

2 初步分析

2.1 Practice 1

Sample Scatter Plot

Fig. 5 是由true model (紫色虛線，$y = 2x + 1$) 與100組資料組成的散點圖。由於數據中加入隨機誤差，各數據點具有不同的noise值，故可以從Fig. 5 中看出，數據點都接近眞實線，但有不同程度的擺動。有些資料的雜訊較大，所以距離眞實線的偏差較大，但整體趨勢仍貼近眞實線。

Chi-square Histogram

Fig. 7 代表每一組擬合出來的 χ^2 值的統計分佈，綠色理論曲線是自由度dof=8的卡方分布 χ^2 曲線。由於每組有 10 個資料點，並且擬合了兩種參數，故自由度算法爲 10-2=8。透過結果可以看出，中心大概在 8 附近，符合dof=8 之期望值。透過 Fig. 7 可以看出直方圖與理論曲線高度重合，顯示每組擬合後的殘差平方和經過標準化後符合卡方分布。

2.2 Practice 2

Fig. 10上面一排是 a 的直方圖，下面一排是 b 的直方圖。每張子圖中都有一條紅色虛線，爲眞實值 a=2.0、b=1.0。χ^2值小的組別中可以看出a與b的分布比較集中，而且也比較接近眞實值。χ^2值大的組別則會發現a與b的分布較不集中，分布變寬，代表誤差較大，有些明顯偏離眞實值。可以得知，資料所受隨機誤差較小時，擬合出的結果會更貼近眞實值，且noise較大的時候，a與b的直方圖就會變寬(不集中)且會偏離。

2.3 Practice 3

Fig. 12 爲擬合出的斜率分布（藍色）以及擬合出的截距分布（綠色）。兩者皆大致呈現鐘型曲線，且皆有自然、合理的波動，接近常態分佈。

Fig. 13 上方橫軸爲a的取值，縱軸爲b的取值，而顏色深淺則代表出現的次數。從圖中可看出，散佈的形狀爲傾斜的橢圓形，沿著「負斜率」的方向拉長。可看出a 和 b 之間存在負相關，即a的增加會使b減少。

Fig. 13 黑白部分上方爲a的一維直方圖，右側爲b的一維直方圖，左下角爲a和b的聯合分布。可透過上方與左下看出a、b各自的分布範圍和中心，左下方則能觀測聯合分佈呈現負傾斜橢圓的情形。再次看出a和b單獨皆呈常態分布，而當a與b一起變化時，會有明顯負相關。

3 具體說明實驗遇到的問題，或分析可能的問題

3.1 Color choices in Figure 10

To overlap the histograms of the grouped parameters' distributions (Fig. 10 top panel) with a pair of figures (Fig. 10 bottom panel), we encountered a problem about the color choices: how to make the five colors be distinguished and the overlapping area not ugly? We then tried various sets of colors and asked ChatGPT for recommendations. Finally, we adjusted the `alpha` argument in the `plt.hist` to alter the transparency of each color, and the results were ideal.

3.2 Corner plot for better visualization

In Fig. 12 and 13 (top panel), they roughly shared the information of how the fitted parameters were distributed. Therefore, we referred to the visualization method of MCMC fitting, which conventionally uses a "corner plot" to showcase how the parameters are distributed and how each pair of parameters correlates. `corner` python module was then introduced to produce the bottom panel of Fig. 13.