實驗物理學（二）
實驗報告

Fundamental Python
Chi-square fitting - 2

Group 2

洪　瑜 B125090009
黃巧涵 B122030003
洪懌平 B102030019

2025/05/06

摘要

This week's experiment investigates the statistical principles underlying linear curve fitting using Python simulations, focusing on the chi-square (χ^2) test, goodness of fit, and parameter correlation. In Practice 1, we generated multiple noisy datasets based on a linear model and applied least-squares fitting to each, evaluating the fit using reduced chi-square statistics. Practice 2 grouped the results by chi-square values to analyze how noise affects parameter estimation, revealing that datasets with lower noise yield parameter distributions closer to actual values. Adjustments were made to improve grouping by using varying noise levels. Practice 3 further explored the covariance and correlation between parameters a and b, confirming a strong negative correlation through covariance matrix analysis and 2D histograms. Overfitting was illustrated using high-degree polynomials, which fit training data well but performed poorly on new data, highlighting the dangers of excessive model complexity. This study deepens our understanding of fitting reliability and offers practical insights into using statistical tools in physical experiments.

1 前言

In this week's coding exercises, we performed three practices to (1) connect the concept of χ^2 with curve fitting and understand the degree of freedom, (2) clarify the concept of goodness of fit, and (3) understand the covariance matrix and the correlation coefficient. Moreover, we investigated the effect of overfitting by simply modifying some existing codes.

The following subsections briefly introduce the concept of **goodness of fit** and the concept of **covariance matrix and correlation coefficient**.

1.1 Goodness of Fit

In data analysis and model fitting, it's essential to evaluate how well a proposed model explains the observed data. One standard method for this is the reduced chi-square statistic, denoted as χ^2_ν, which provides a normalized measure of the discrepancy between observed values and model predictions.

The reduced chi-square is defined as:

$$\chi^2_\nu = \frac{\chi^2}{\nu} = \frac{1}{N-p}\sum_{i=1}^{N}\left(\frac{y_i - f(x_i)}{\sigma_i}\right)^2 \tag{1}$$

where

- y_i are the experimental data points
- $f(x_i)$ are the model predictions with parameters x_i
- σ_i are the uncertainties in the experiments
- N is the number of data points and p is the number of model's parameters

To interpret the evaluation of χ^2_ν is an approach to testify the goodness of fitting:

- $\chi^2_\nu \approx 1$: The model fits the data, and the uncertainties are likely well estimated.
- $\chi^2_\nu > 1$: The model does not explain the data within the given uncertainties (possibly poor fit or underestimated errors).
- $\chi^2_\nu < 1$: The data scatter less than expected (possibly overestimated errors or overfitting).

1.2 Covariance Matrix and Correlation Coefficient

In data analysis, particularly in the context of parameter estimation and model fitting, the covariance matrix and uncertainty of each parameter are essential concepts used to assess the reliability and relationship of the parameters being estimated.

The covariance matrix is a square matrix that measures the covariance (i.e., how two variables change together) between each pair of parameters in a multidimensional dataset. It is advantageous when multiple parameters are estimated simultaneously.

For example, if we consider a two-parameter model with parameters a and b, the covariance matrix is

$$\mathrm{pcov} = \begin{pmatrix} \sigma_a^2 & \mathrm{cov}(b,a) \\ \mathrm{cov}(a,b) & \sigma_b^2 \end{pmatrix} \tag{2}$$

In the covariance matrix, the diagonal elements (σ_a^2 and σ_b^2) represent the variance of each parameter, and the off-diagonal elements ($\mathrm{cov}(b,a)$ and $\mathrm{cov}(a,b)$) are the covariance between each pair of parameters.

The correlation coefficient (often denoted as R) is a normalized version of covariance, defined as:

$$R_{ab} = \frac{\mathrm{cov}(a,b)}{\sigma_a \sigma_b} \tag{3}$$

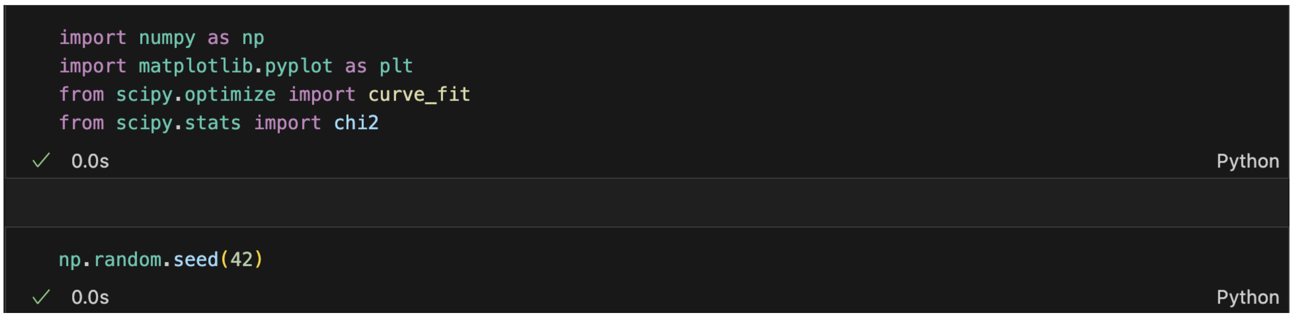
- $R = -1$: Perfect negative linear relation
- $R = +0$: No linear correlation.
- $R = +1$: Perfect positive linear relation

Moreover, the coefficient of determination, R^2, is the square of the correlation coefficient when doing simple linear regression. R^2 means:

- $R_{ab}^2 = 1$: Perfect fit.
- $R_{ab}^2 = 0$: No fit.
- $R_{ab}^2 = k$: k% of the variance in b is explained by a.

2 實驗步驟

2.1 Practice 1



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

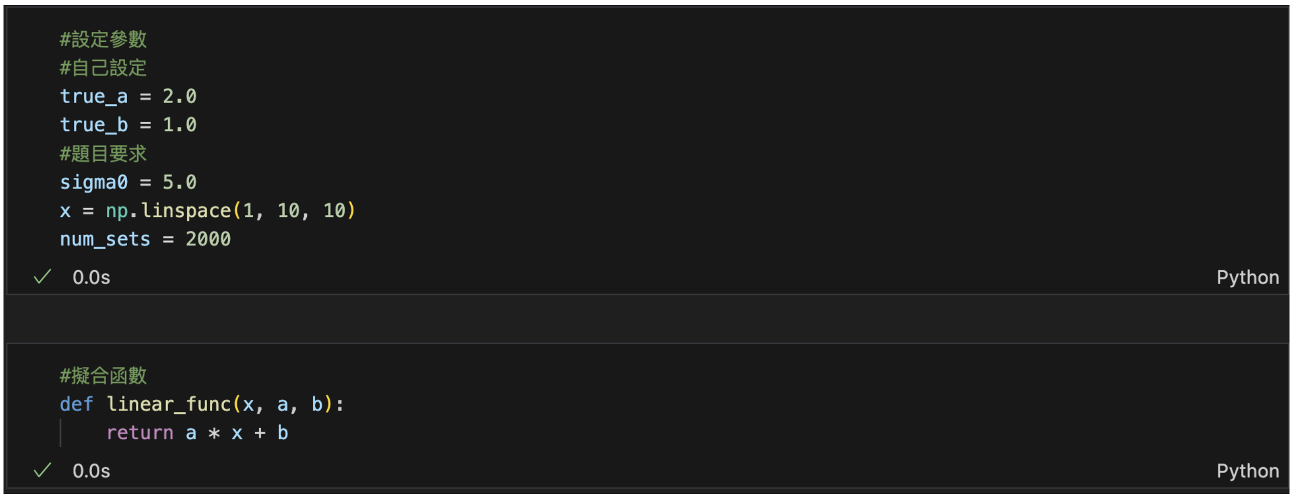
✓ 0.0s Python

```
np.random.seed(42)
```

✓ 0.0s Python

Figure 1: Import packages and fix the random seed to 42

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



```
#設定參數
#自己設定
true_a = 2.0
true_b = 1.0
#題目要求
sigma0 = 5.0
x = np.linspace(1, 10, 10)
num_sets = 2000
```

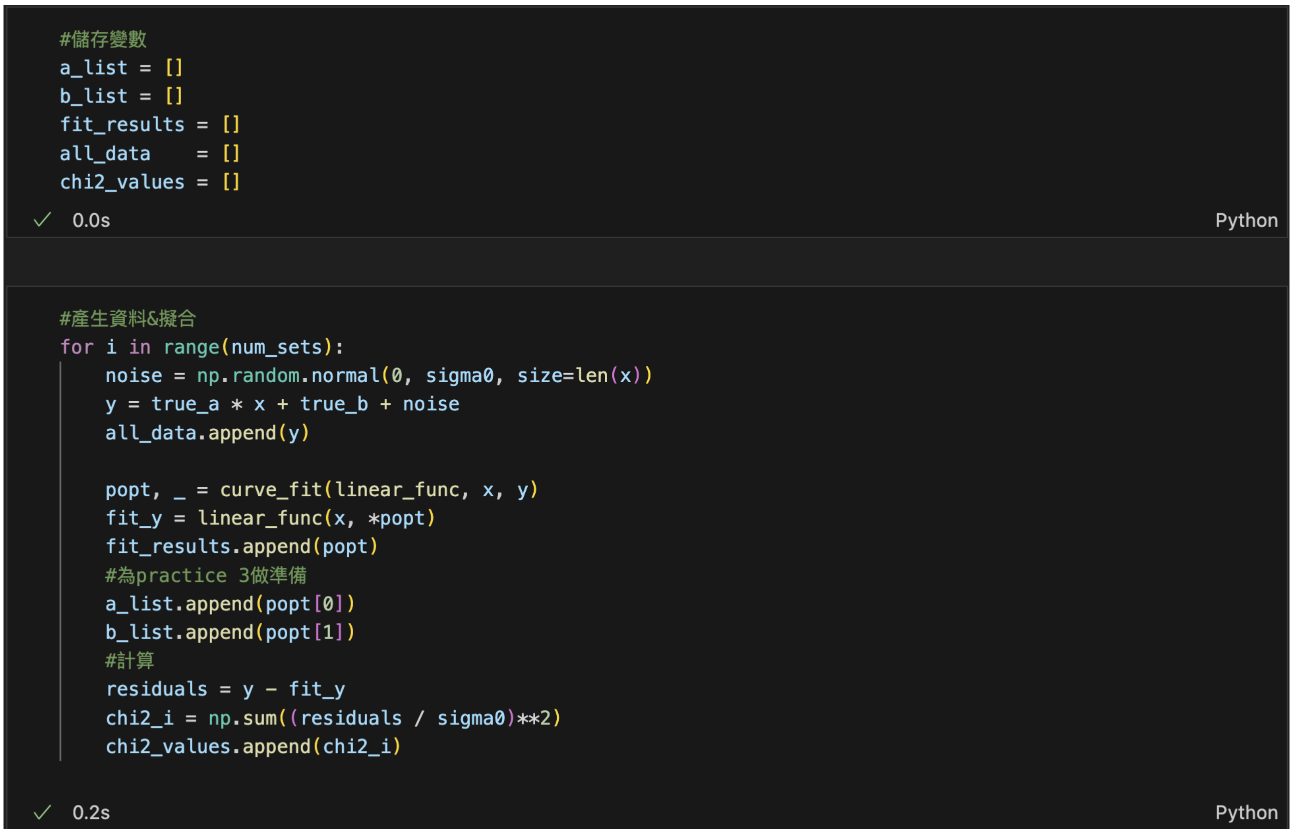
✓ 0.0s Python

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

✓ 0.0s Python

Figure 2: Generate samples with assigned mean, noise, and dataset size.

3. Fit the data points set by set with the function $y = a * x + b$ using `curve_fit`.
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. $\chi_i^2 = \sum_{j=1}^{10} \frac{(\text{data\_point}_{i,j} - \text{fitted\_result}_{i,j})^2}{\sigma_j^2}$ $(i = 1 \sim 2000)$ where i is the index of data set and j is the index of x value in one data set. $(\text{len}(\chi^2) = 2000)$
6. You might find out that this χ^2 is just the sum of the squares 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 $y = a * x + b + \text{noise}$.



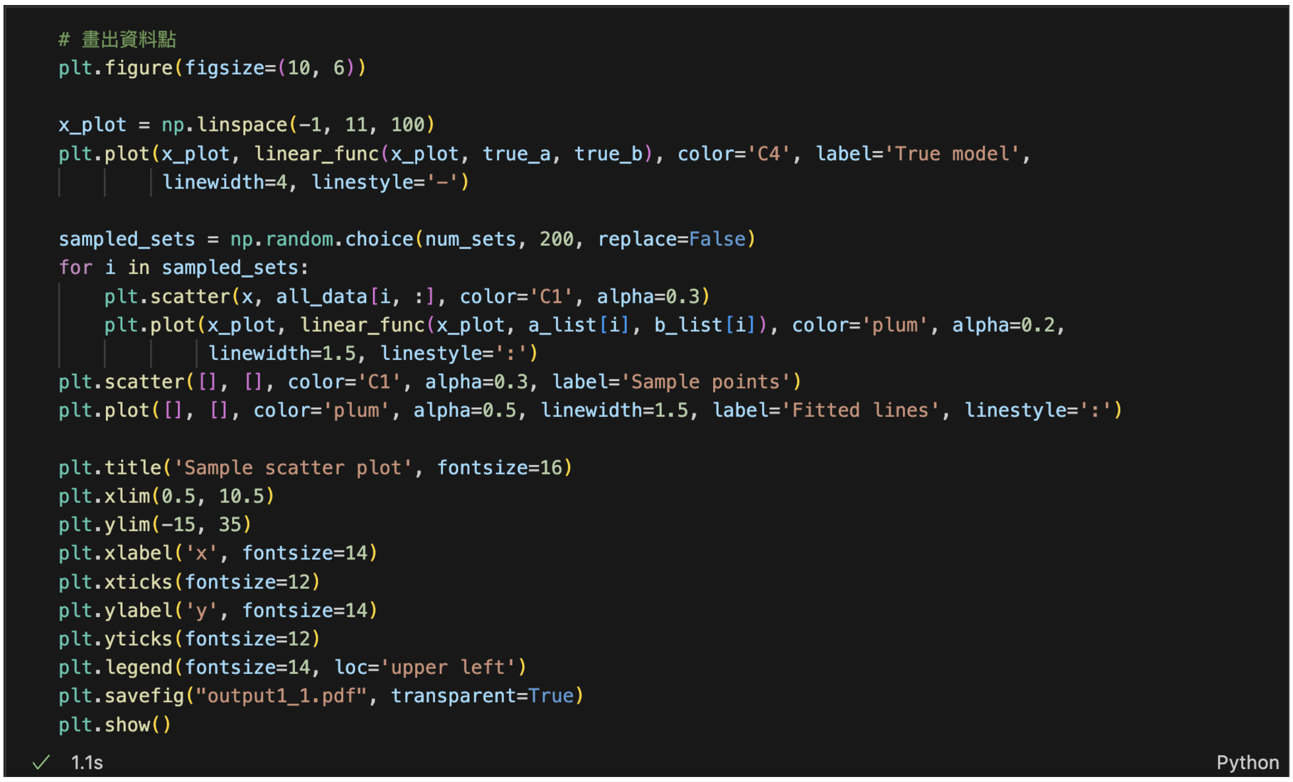
```python
#儲存變數
a_list = []
b_list = []
fit_results = []
all_data    = []
chi2_values = []
```

✓ 0.0s Python

```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)
```

✓ 0.2s Python

Figure 3: Fit the data points set by set and calculate the χ^2 values.



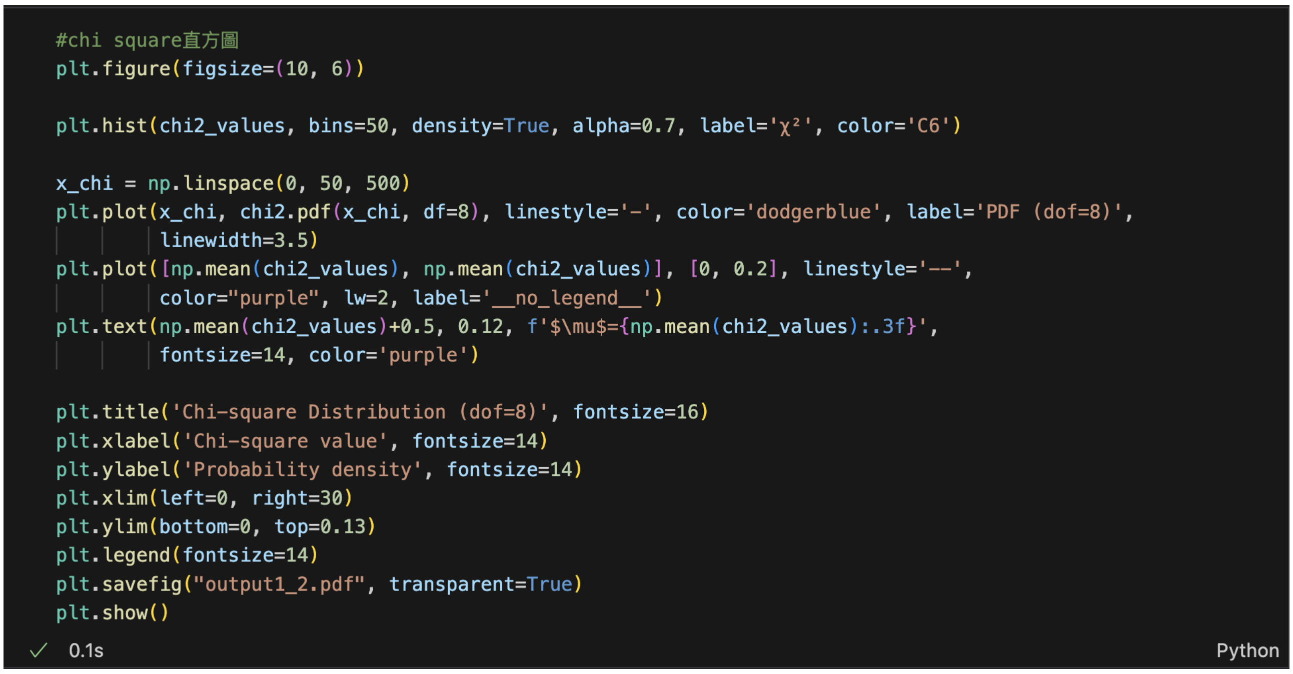
```python
# 畫出資料點
plt.figure(figsize=(10, 6))

x_plot = np.linspace(-1, 11, 100)
plt.plot(x_plot, linear_func(x_plot, true_a, true_b), color='C4', label='True model',
            linewidth=4, linestyle='-')

sampled_sets = np.random.choice(num_sets, 200, replace=False)
for i in sampled_sets:
    plt.scatter(x, all_data[i, :], color='C1', alpha=0.3)
    plt.plot(x_plot, linear_func(x_plot, a_list[i], b_list[i]), color='plum', alpha=0.2,
                linewidth=1.5, linestyle=':')
plt.scatter([], [], color='C1', alpha=0.3, label='Sample points')
plt.plot([], [], color='plum', alpha=0.5, linewidth=1.5, label='Fitted lines', linestyle=':')

plt.title('Sample scatter plot', fontsize=16)
plt.xlim(0.5, 10.5)
plt.ylim(-15, 35)
plt.xlabel('x', fontsize=14)
plt.xticks(fontsize=12)
plt.ylabel('y', fontsize=14)
plt.yticks(fontsize=12)
plt.legend(fontsize=14, loc='upper left')
plt.savefig("output1_1.pdf", transparent=True)
plt.show()
```

✓ 1.1s Python

Figure 4: Plot all data points

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)

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



```python
#chi square直方圖
plt.figure(figsize=(10, 6))

plt.hist(chi2_values, bins=50, density=True, alpha=0.7, label='χ²', color='C6')

x_chi = np.linspace(0, 50, 500)
plt.plot(x_chi, chi2.pdf(x_chi, df=8), linestyle='-', color='dodgerblue', label='PDF (dof=8)',
         linewidth=3.5)
plt.plot([np.mean(chi2_values), np.mean(chi2_values)], [0, 0.2], linestyle='--',
         color="purple", lw=2, label='__no_legend__')
plt.text(np.mean(chi2_values)+0.5, 0.12, f'$\mu$={np.mean(chi2_values):.3f}',
         fontsize=14, color='purple')

plt.title('Chi-square Distribution (dof=8)', fontsize=16)
plt.xlabel('Chi-square value', fontsize=14)
plt.ylabel('Probability density', fontsize=14)
plt.xlim(left=0, right=30)
plt.ylim(bottom=0, top=0.13)
plt.legend(fontsize=14)
plt.savefig("output1_2.pdf", transparent=True)
plt.show()
```

Figure 5: Plot the histogram of the χ^2 values and compare to the PDF of chi-square distribution

2.2 Practice 2

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

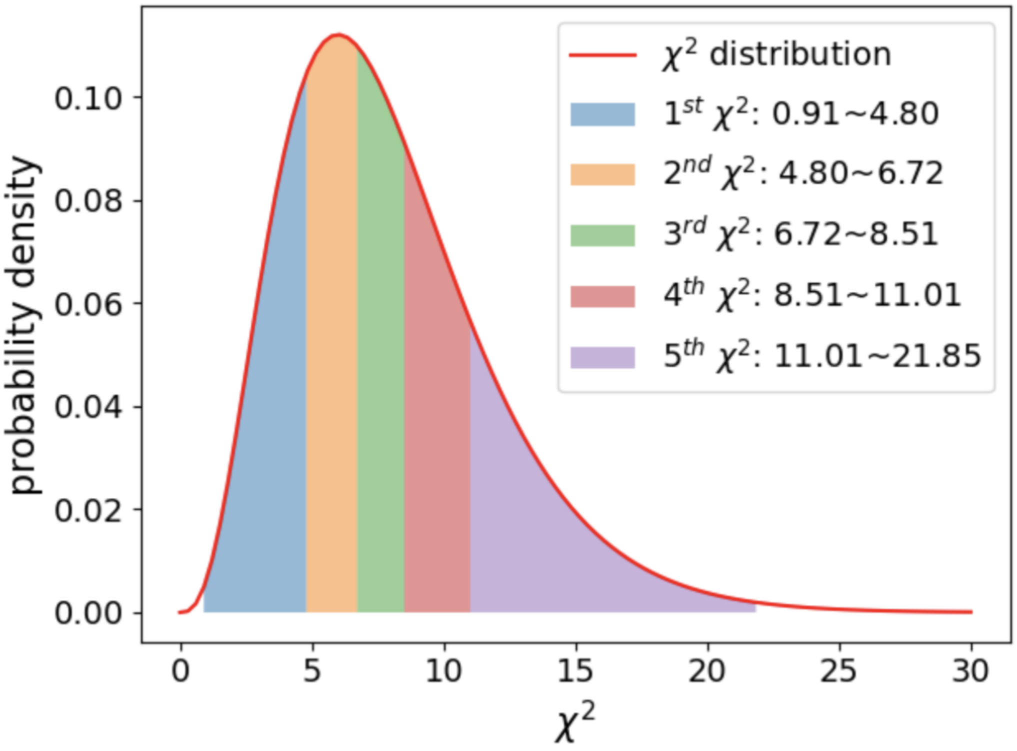
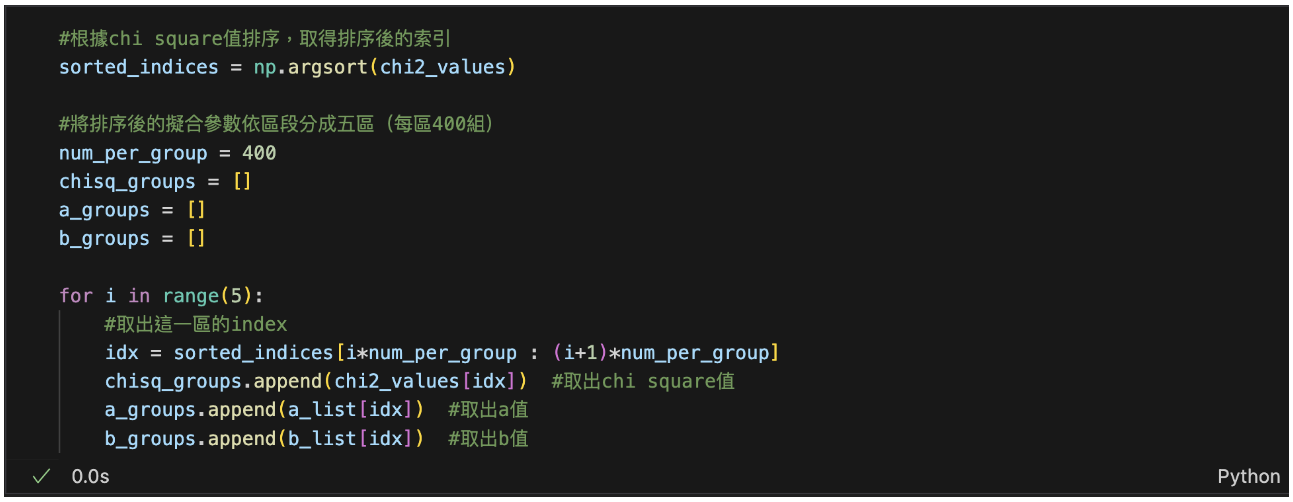


Figure 6: "Figure 1" in the course material.

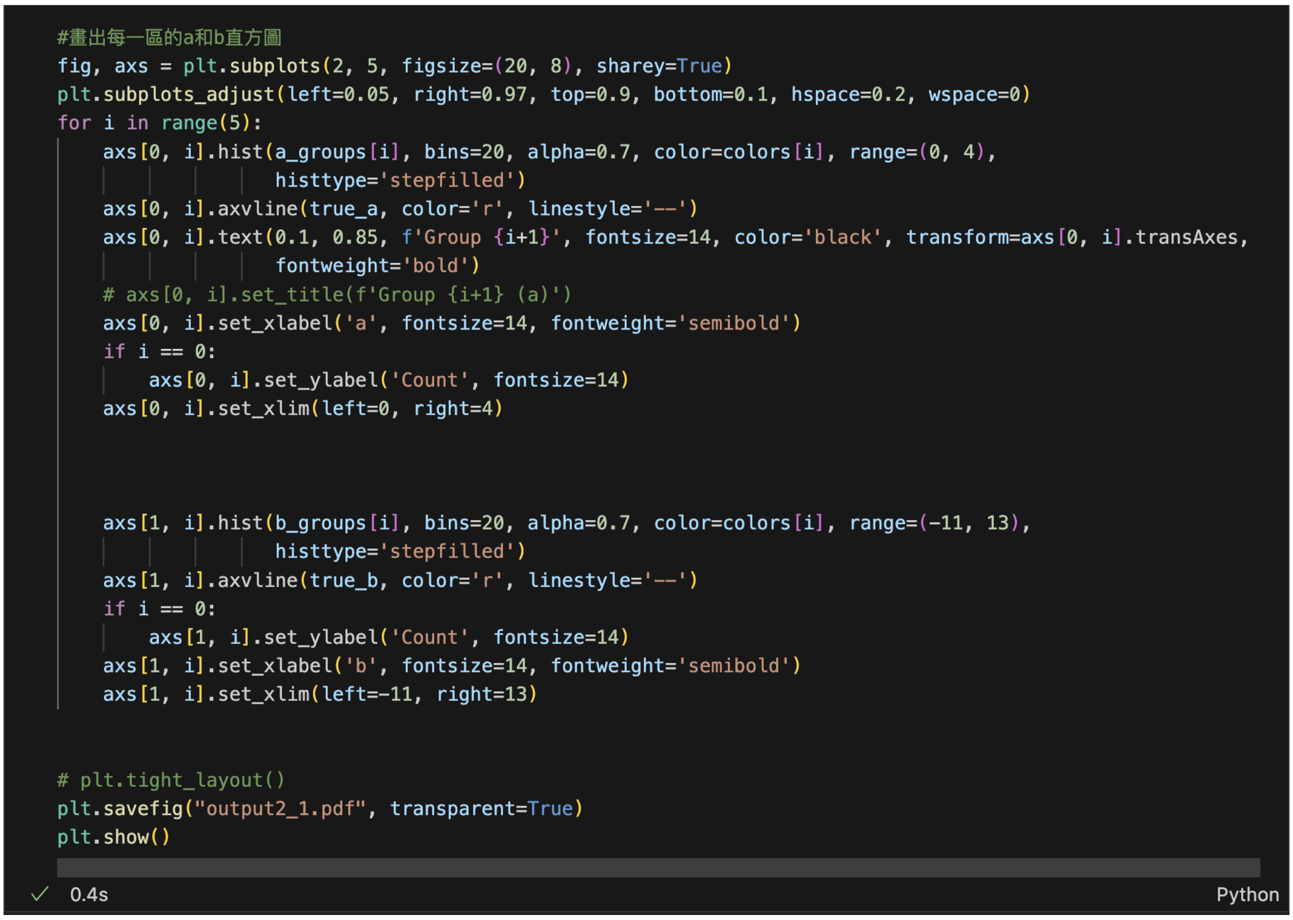
2. Each section contains 400 sets of data points, grouped as follows: 1–400, 401–800, 801–1200, 1201–1600, and 1601–2000, ordered by χ^2 value.
3. You'll need to get those a and b values sorted by the χ^2 value and choose certain indexes that are in the section to plot the histogram.



```python
#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
chisq_groups = []
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]
    chisq_groups.append(chi2_values[idx])  #取出chi square值
    a_groups.append(a_list[idx])  #取出a值
    b_groups.append(b_list[idx])  #取出b值
```

✓ 0.0s Python

Figure 7: Group data points into 5 groups according to their chi-square value



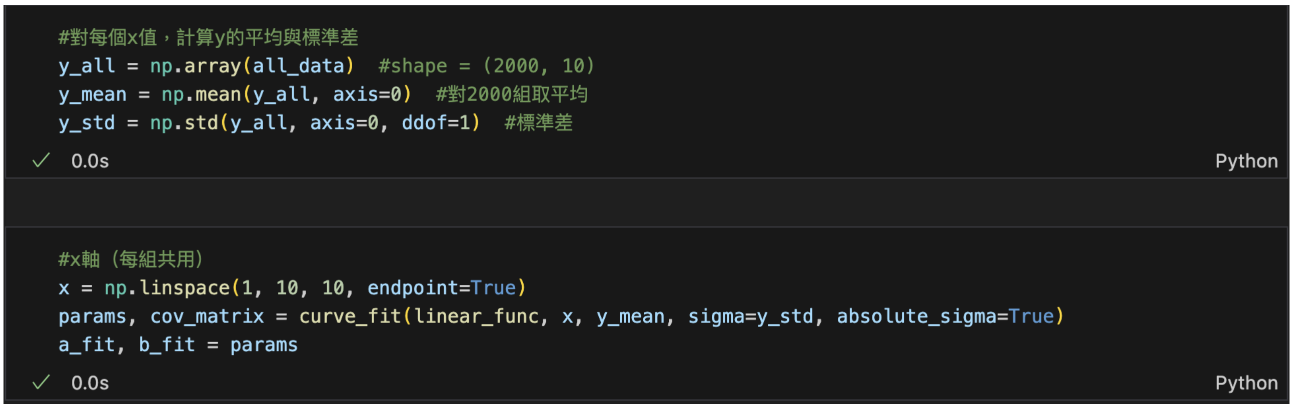
```python
#畫出每一區的a和b直方圖
fig, axs = plt.subplots(2, 5, figsize=(20, 8), sharey=True)
plt.subplots_adjust(left=0.05, right=0.97, top=0.9, bottom=0.1, hspace=0.2, wspace=0)
for i in range(5):
    axs[0, i].hist(a_groups[i], bins=20, alpha=0.7, color=colors[i], range=(0, 4),
                   histtype='stepfilled')
    axs[0, i].axvline(true_a, color='r', linestyle='--')
    axs[0, i].text(0.1, 0.85, f'Group {i+1}', fontsize=14, color='black', transform=axs[0, i].transAxes,
                   fontweight='bold')
    # axs[0, i].set_title(f'Group {i+1} (a)')
    axs[0, i].set_xlabel('a', fontsize=14, fontweight='semibold')
    if i == 0:
        axs[0, i].set_ylabel('Count', fontsize=14)
    axs[0, i].set_xlim(left=0, right=4)



    axs[1, i].hist(b_groups[i], bins=20, alpha=0.7, color=colors[i], range=(-11, 13),
                   histtype='stepfilled')
    axs[1, i].axvline(true_b, color='r', linestyle='--')
    if i == 0:
        axs[1, i].set_ylabel('Count', fontsize=14)
    axs[1, i].set_xlabel('b', fontsize=14, fontweight='semibold')
    axs[1, i].set_xlim(left=-11, right=13)


# plt.tight_layout()
plt.savefig("output2_1.pdf", transparent=True)
plt.show()
```

✓ 0.4s Python

Figure 8: Plot the histograms of a and b in five groups

2.3 Practice 3

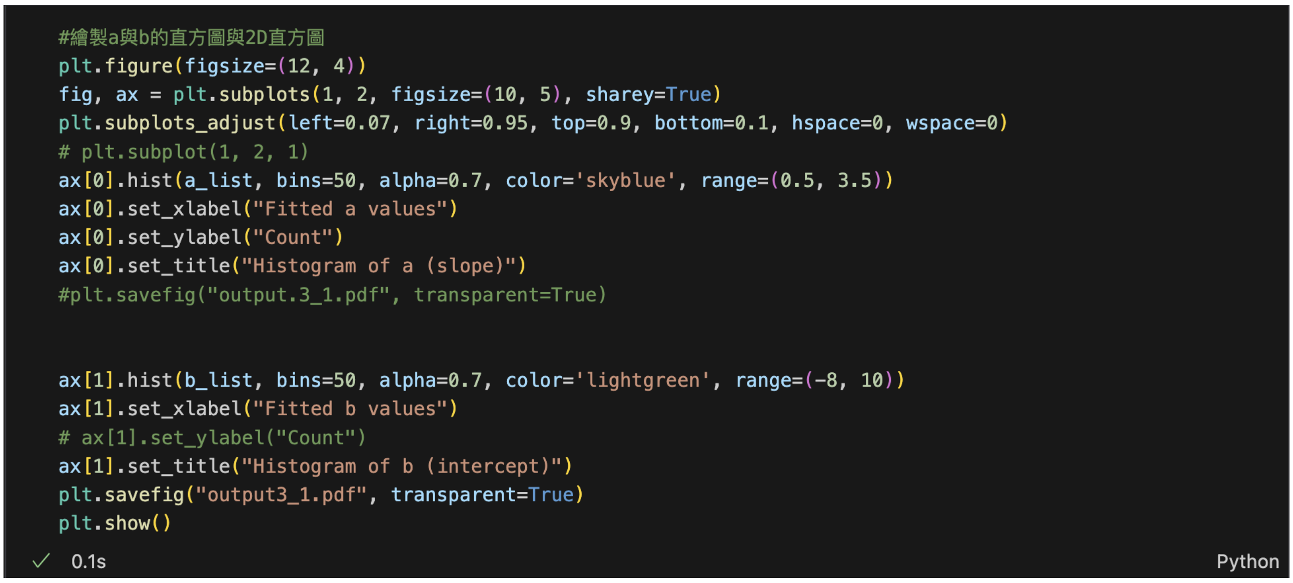
1. By using the data points generated in **Practice 1**, get the mean and standard deviation of these data points.
2. Fit these mean value of y with the function $y = a * x + b$ using `curve_fit`, and remember to set `absolute_sigma=True`.



```
#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
✓ 0.0s                                                    Python

#x軸 (每組共用)
x = np.linspace(1, 10, 10, endpoint=True)
params, cov_matrix = curve_fit(linear_func, x, y_mean, sigma=y_std, absolute_sigma=True)
a_fit, b_fit = params
✓ 0.0s                                                    Python
```

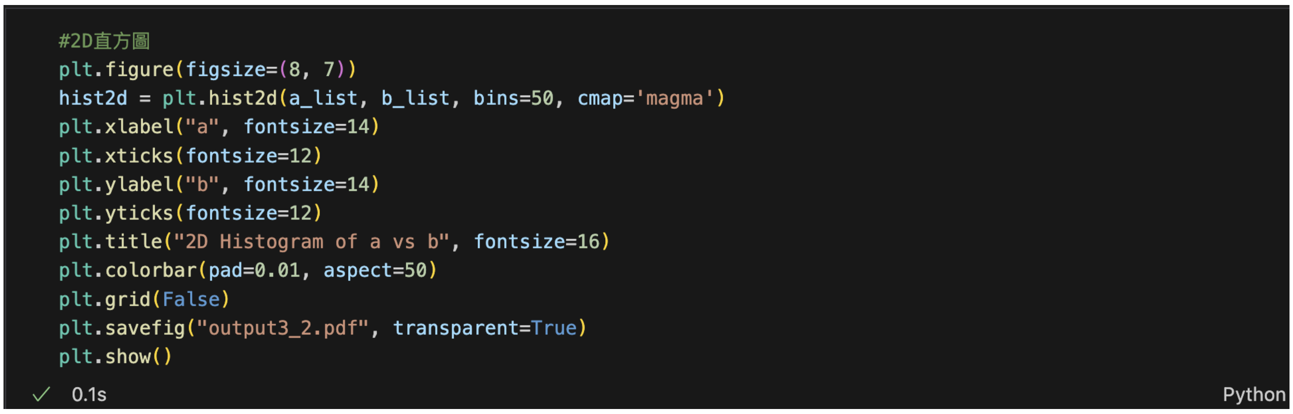
Figure 9: Calculate the mean and standard deviation of the data points and fit them with `curve_fit`

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)



```
#繪製a與b的直方圖與2D直方圖
plt.figure(figsize=(12, 4))
fig, ax = plt.subplots(1, 2, figsize=(10, 5), sharey=True)
plt.subplots_adjust(left=0.07, right=0.95, top=0.9, bottom=0.1, hspace=0, wspace=0)
# plt.subplot(1, 2, 1)
ax[0].hist(a_list, bins=50, alpha=0.7, color='skyblue', range=(0.5, 3.5))
ax[0].set_xlabel("Fitted a values")
ax[0].set_ylabel("Count")
ax[0].set_title("Histogram of a (slope)")
#plt.savefig("output.3_1.pdf", transparent=True)


ax[1].hist(b_list, bins=50, alpha=0.7, color='lightgreen', range=(-8, 10))
ax[1].set_xlabel("Fitted b values")
# ax[1].set_ylabel("Count")
ax[1].set_title("Histogram of b (intercept)")
plt.savefig("output3_1.pdf", transparent=True)
plt.show()
✓ 0.1s                                                    Python
```

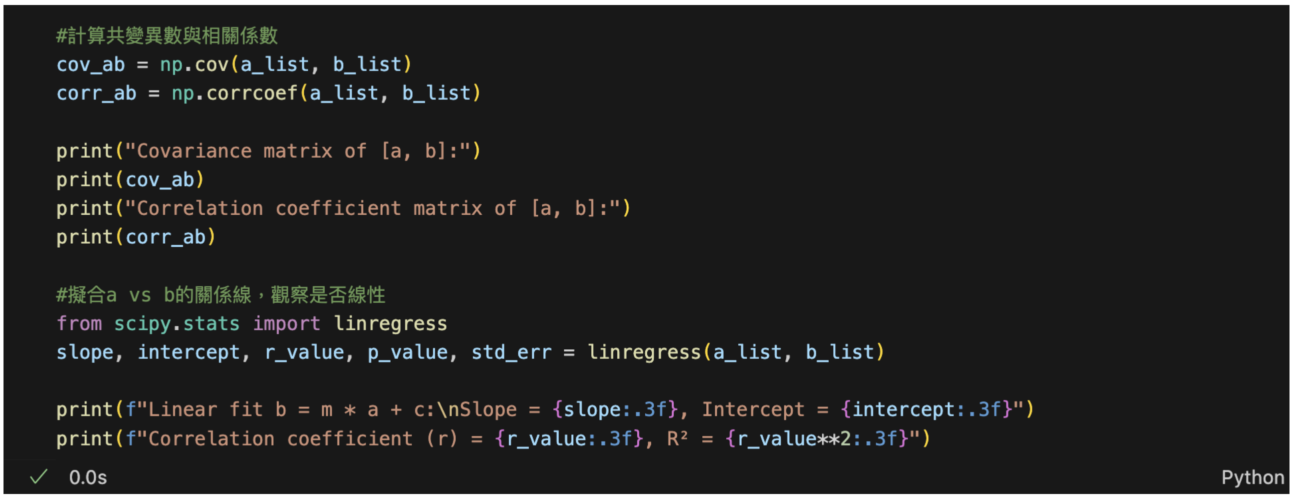
Figure 10: 1D histogram of the fitted results



```python
#2D直方圖
plt.figure(figsize=(8, 7))
hist2d = plt.hist2d(a_list, b_list, bins=50, cmap='magma')
plt.xlabel("a", fontsize=14)
plt.xticks(fontsize=12)
plt.ylabel("b", fontsize=14)
plt.yticks(fontsize=12)
plt.title("2D Histogram of a vs b", fontsize=16)
plt.colorbar(pad=0.01, aspect=50)
plt.grid(False)
plt.savefig("output3_2.pdf", transparent=True)
plt.show()
```

Figure 11: 2D histogram of the fitted results

4. Compare the standard deviation and correlation coefficient obtained from the fitted covariance matrix with those derived from statistical graphs of a and b.
5. Perform a linear fit on the 2D histogram of parameters a and b to obtain the slope, which characterizes certain properties useful for calculating the correlation coefficient.
6. Ensure the definition of those quantities you are calculating is correct.
7. Finally, you should be able to find that everything is consistent with the covariance matrix. (The values will be close to each other)



```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 12: Calculating the correlation coefficient by performing a linear fit on parameters a and b

3 實驗數據與分析

3.1 Practice 1

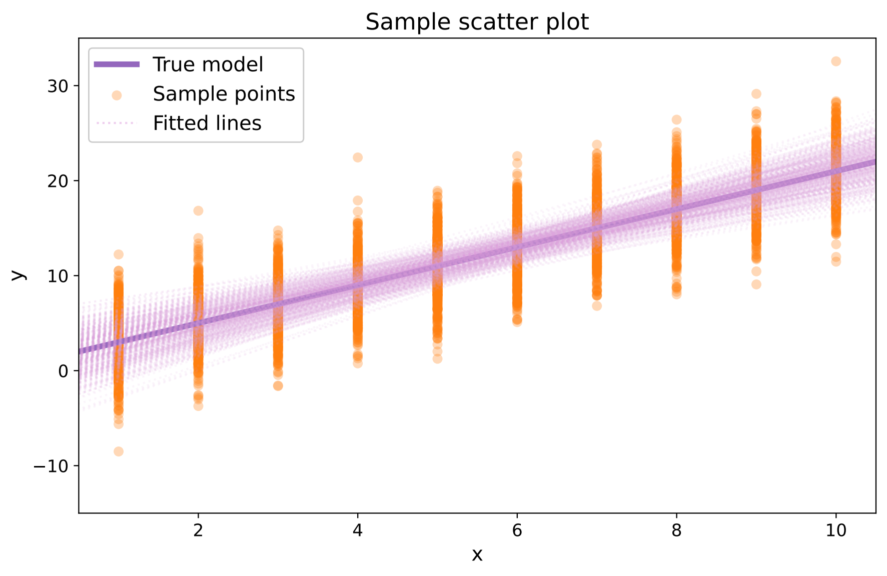


Figure 13: Sample points of our data (plotted only 500 sets for representative)

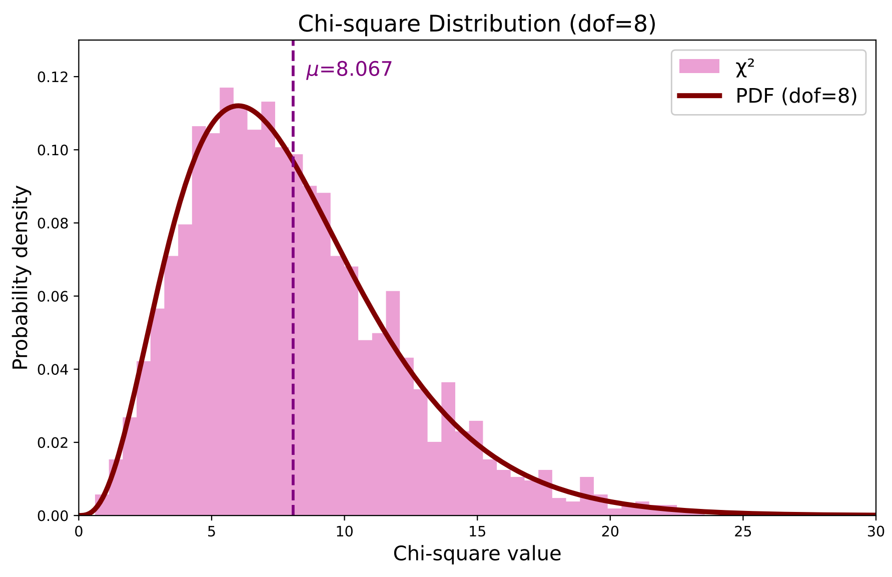


Figure 14: Histogram of the chi-square distributions

Resembling last week's practices, the experimental dataset is sampled from a linear model $y = ax + b$ with a Gaussian noise $\sigma_0 = 3$. For clarification, Fig. 13 randomly plots 500 sets out of the 2000 datasets (orange scattered dots). The model and the fitted curves, generated by `curve_fit` with the linear model, are also displayed in Fig. 13 (purple solid and dotted lines, respectively). The means and the standard deviations of the data are listed in the Table. 1.

The histogram of chi-square values is shown in Fig. 14, overlapped with the probability density function (PDF) of the chi-square distribution with degrees of freedom (dof) equal to 8. The dof ν is assigned to be $N - p = 10 - 2 = 8$ (Sec. 1.1). As expected, the mean chi-square value of 8.067 approximates the degrees of freedom, and the probability density function aligns well with the chi-square distribution.

3.2 Practice 2

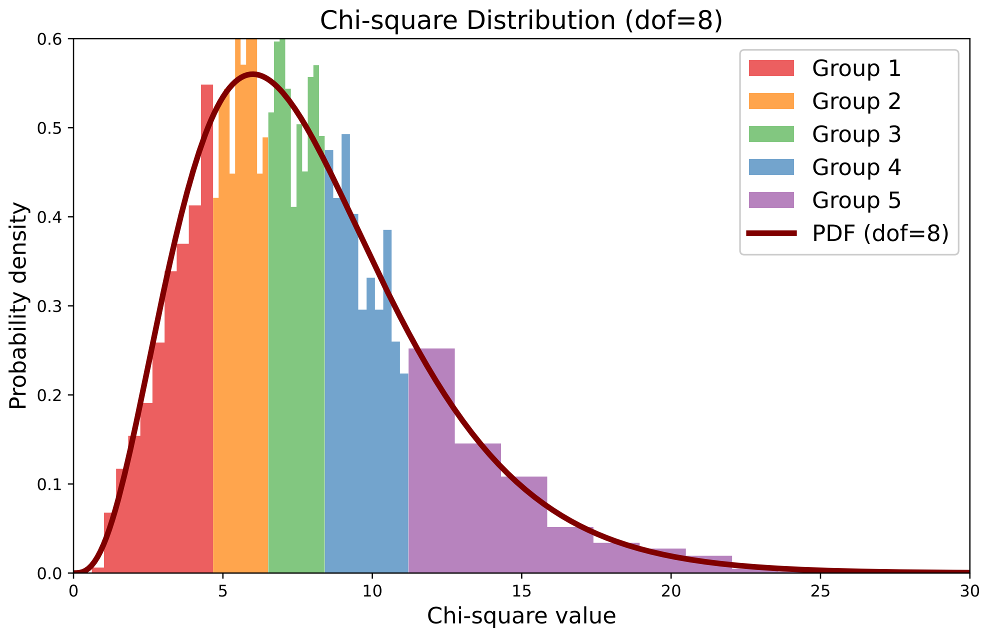


Figure 15: Chi-square distribution separated into five groups according to their chi-square values

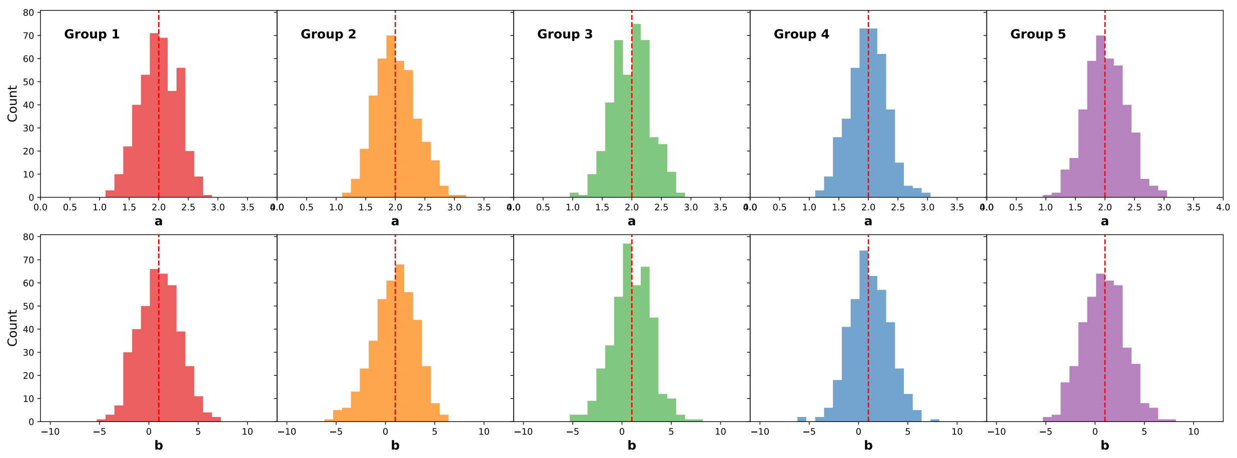


Figure 16: Histograms of a and b

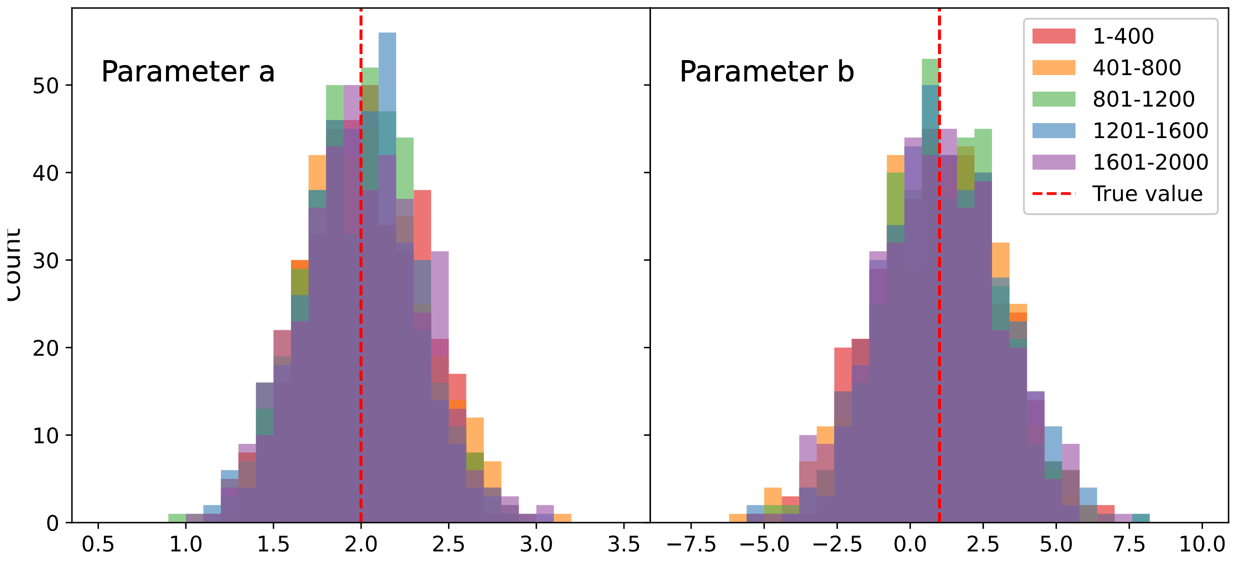


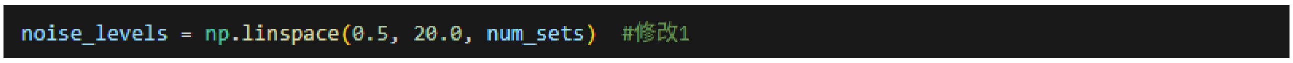
Figure 17: Overlap the five histograms of Fig. 16

可從Fig.15 ∼ Fig.17可見結果不合預期，分組差異不明顯。
分析原因如下：

- 原本每組資料之標準差固定爲3.0，導致資料的誤差相同，無法用χ^2分辨擬合好壞。
- χ^2之計算會將殘差標準化，所以每一組都使用對應的σ計算χ^2，當誤差太大時，其會使公式分母變大，使χ^2看起來還是很小。

我們透過以下方式改進：

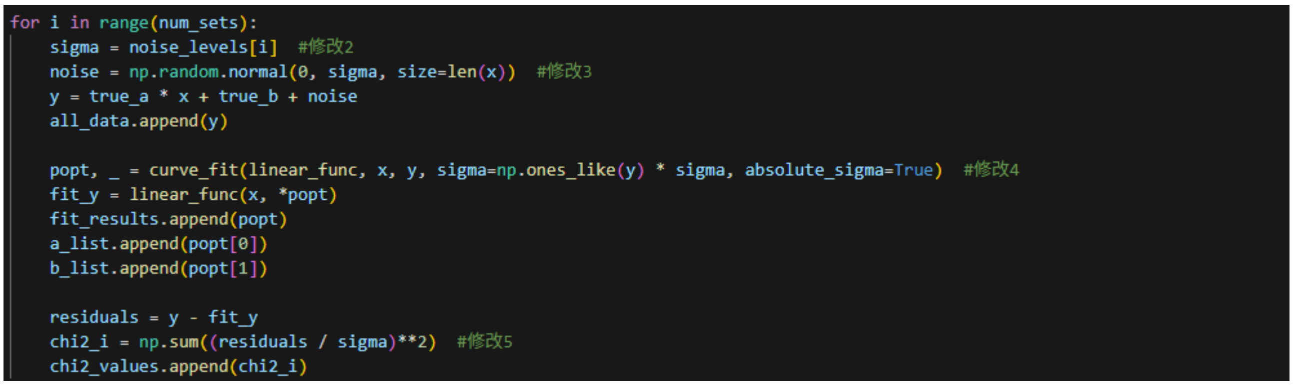
- 讓每組資料的標準差不同，提升數據的變異性。



```
noise_levels = np.linspace(0.5, 20.0, num_sets)  #修改1
```

Figure 18: 修改後程式碼(1)

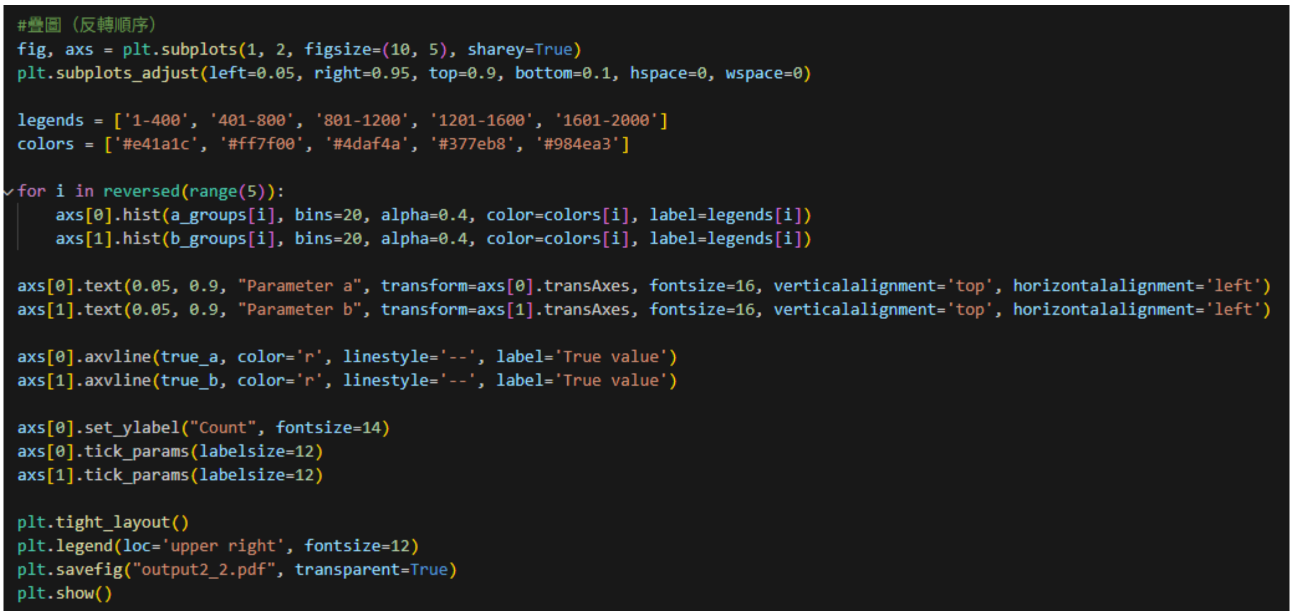
- 不使用χ^2分組，改成使用σ分組。



```
for i in range(num_sets):
    sigma = noise_levels[i]  #修改2
    noise = np.random.normal(0, sigma, size=len(x))  #修改3
    y = true_a * x + true_b + noise
    all_data.append(y)

    popt, _ = curve_fit(linear_func, x, y, sigma=np.ones_like(y) * sigma, absolute_sigma=True)  #修改4
    fit_y = linear_func(x, *popt)
    fit_results.append(popt)
    a_list.append(popt[0])
    b_list.append(popt[1])

    residuals = y - fit_y
    chi2_i = np.sum((residuals / sigma)**2)  #修改5
    chi2_values.append(chi2_i)
```

Figure 19: 修改後程式碼(2)

- 修改疊圖順序，使結果更明顯。



```python
#疊圖（反轉順序）
fig, axs = plt.subplots(1, 2, figsize=(10, 5), sharey=True)
plt.subplots_adjust(left=0.05, right=0.95, top=0.9, bottom=0.1, hspace=0, wspace=0)

legends = ['1-400', '401-800', '801-1200', '1201-1600', '1601-2000']
colors = ['#e41a1c', '#ff7f00', '#4daf4a', '#377eb8', '#984ea3']

for i in reversed(range(5)):
    axs[0].hist(a_groups[i], bins=20, alpha=0.4, color=colors[i], label=legends[i])
    axs[1].hist(b_groups[i], bins=20, alpha=0.4, color=colors[i], label=legends[i])

axs[0].text(0.05, 0.9, "Parameter a", transform=axs[0].transAxes, fontsize=16, verticalalignment='top', horizontalalignment='left')
axs[1].text(0.05, 0.9, "Parameter b", transform=axs[1].transAxes, fontsize=16, verticalalignment='top', horizontalalignment='left')

axs[0].axvline(true_a, color='r', linestyle='--', label='True value')
axs[1].axvline(true_b, color='r', linestyle='--', label='True value')

axs[0].set_ylabel("Count", fontsize=14)
axs[0].tick_params(labelsize=12)
axs[1].tick_params(labelsize=12)

plt.tight_layout()
plt.legend(loc='upper right', fontsize=12)
plt.savefig("output2_2.pdf", transparent=True)
plt.show()
```

Figure 20: 修改後程式碼(3)

修改後輸出結果如下：

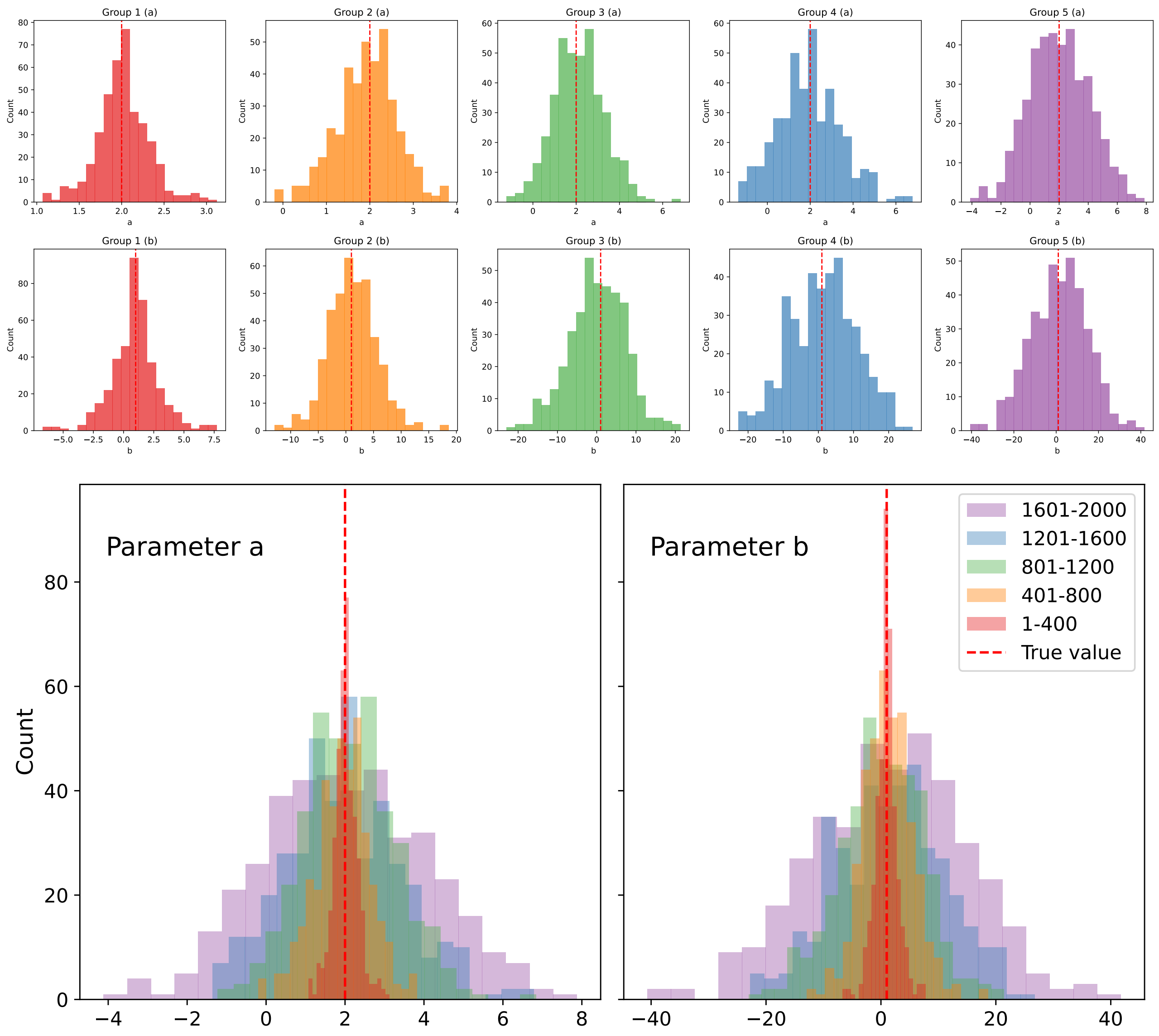


Figure 21: 修改後a和b的直方圖與其疊圖

將數據根據誤差分成五組時，可觀察到擬合參數a與b的分布隨著誤差增大而變得分散。*Group* 1（誤差較低）分布最集中；*Group* 5（誤差較高）則分布較廣，成功呈現預期的趨勢。

3.3 Practice 3

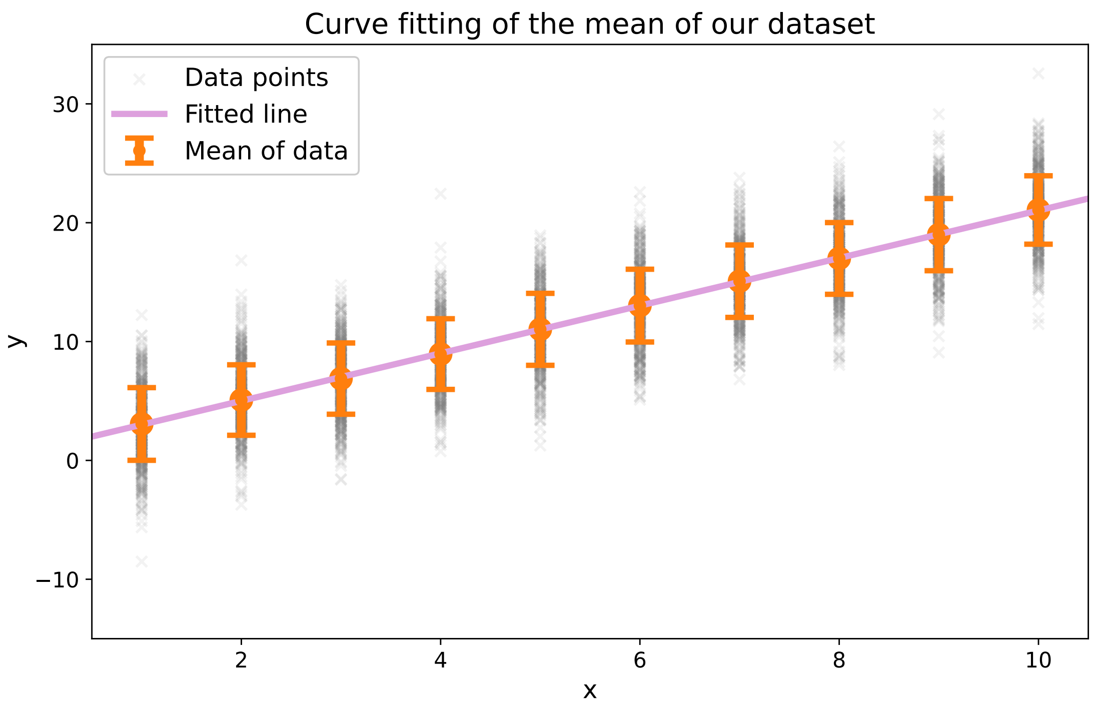


Figure 22: Curve fitting of the mean of the dataset (The fitted line is $y = 2.003x + 1.002$)

Table 1: The mean and standard deviation of data

x	y_{model}	μ_{data}	σ_{data}
1	3	3.07	3.06
2	5	5.08	2.97
3	7	6.88	3.00
4	9	8.95	2.97
5	11	11.03	3.03
6	13	13.02	3.06
7	15	15.08	3.05
8	17	16.99	3.02
9	19	19.00	3.04
10	21	21.07	2.88

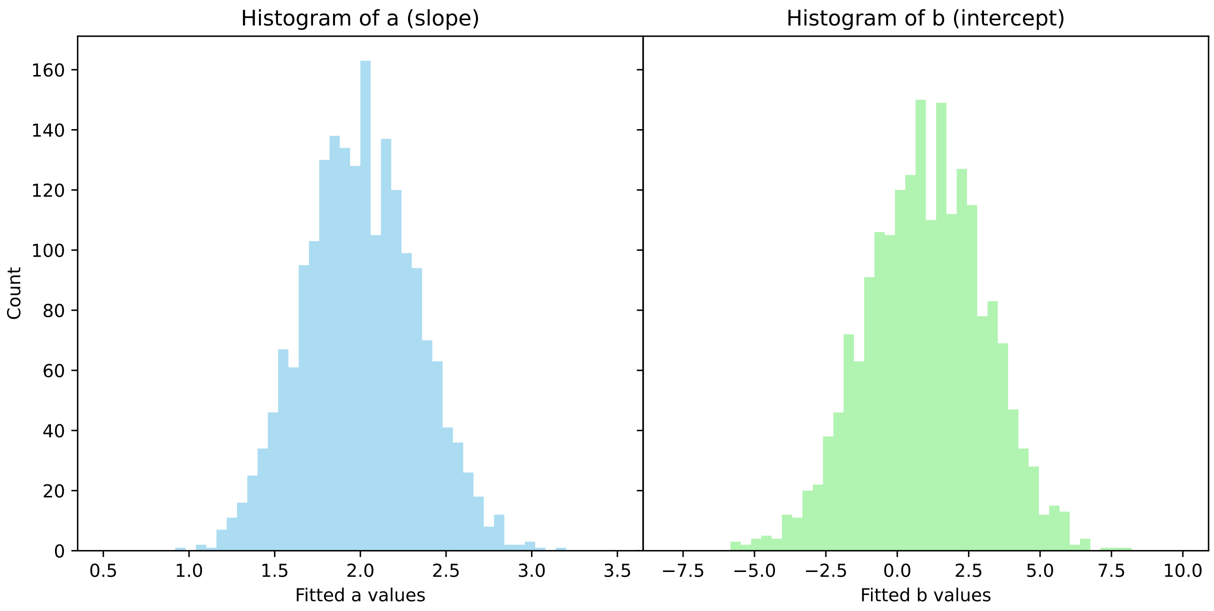


Figure 23: 1D Histogram of fitted a and b.

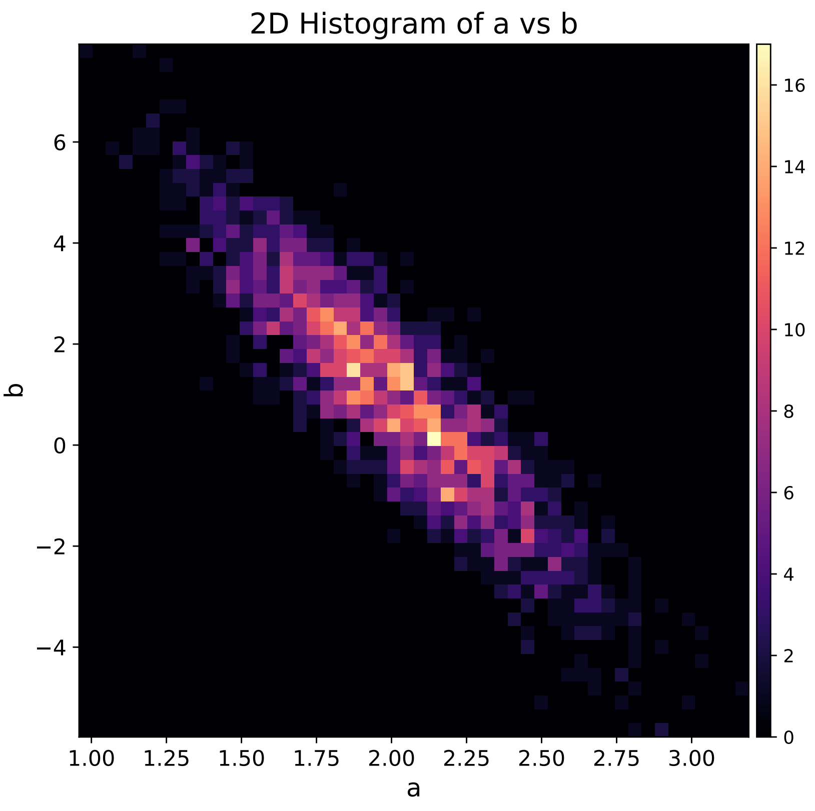


Figure 24: 2D Histogram of fitted a and b.

從Fig.22以及Table.1可看出資料的標準差σ大約爲3，與原始設定的數值相近，顯示出資料的noise設定是合理的。再使用這10個平均資料點加權線性擬合，以每點的標準差作爲權重，可得出結果爲$a = 2.003$、$b = 1.002$，相當接近原始設定值（$a = 2.0$、$b = 1.0$）。

我們將2000組擬合結果的 a 和 b 數值分別繪製成直方圖（Fig.23）。可以觀察到 a 的分布大致呈現高斯形態，平均值接近2；b 的分布稍微比較散一點，平均值接近1。此外，從Fig.24的2D直方圖可以看到資料點分佈是一個傾斜的橢圓形，顯示兩數值呈現負相關。即當 a 值增加時， b 會適度減少以確保整理擬合的結果良好，這就是補償效應；a與b的共變異數矩陣以及相關係數矩陣中， $r = -0.893$ 、$R^2 = 0.797$，也證明了二者之間的強烈負相關性。

4 問題討論

4.1 Practice 1

- The term **model** is often used in the statement of goodness of fit. What does **model** mean in this case?

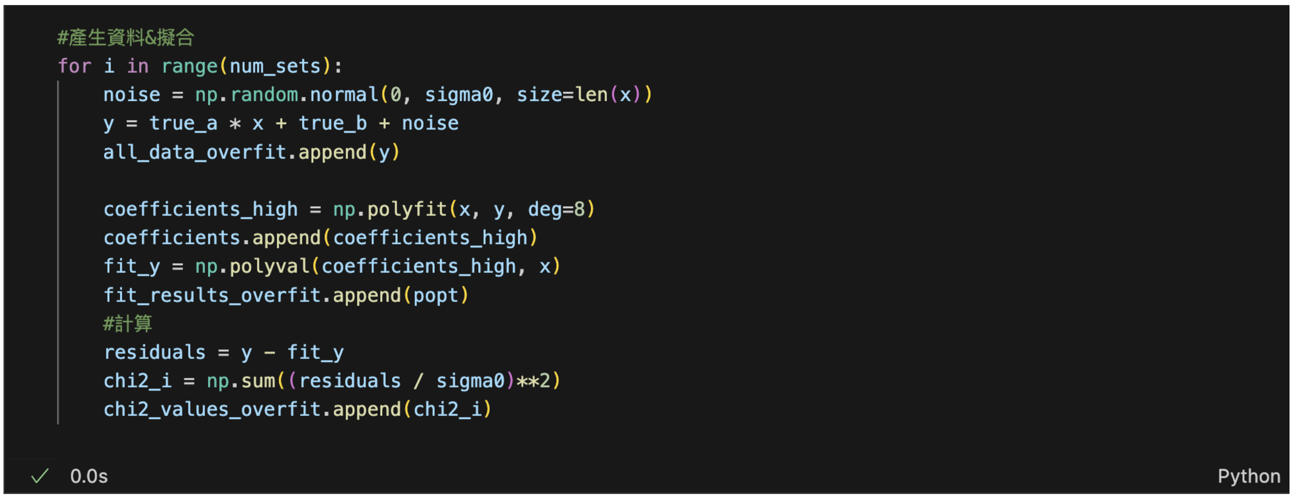
  Model 在這裡是描述數據在生成時所遵循之機制的數學函數，在本次實驗中，我們令數據從$y = a * x + b$這個基礎下生成數據點，並透過LS fitting和χ^2 fitting來找到擬合參數a、b。

- State what **overfitting** is.
  Overfitting（過度擬合）是一種機器學習時出現的不理想行爲。Overfitting 發生可能的原因包含模型沒有一般化（資料筆數太少），或是機器學習過程中學到了資料中包含的 noise (模型自由度較高)，導致雖然在舊的模型上運作可以得到好的表現，但若是套用到新的數據上，可能會有不好的表現。的時候。因此在將新的數據加入資料集時，overfitting會導致在資料判斷上出錯。在 Practice 1 中以 $y = ax + b + noise$的線性模型產生資料，並加入標準差$_0 = 3$的noise，但因爲合理使用線性模型配上線性資料，所以擬合出的結果與理論預期高度一致，沒有觀察到明顯的overfitting情況。

- Design an experiment to illustrate the **overfitting** phenomenon.
  To illustrate the overfitting phenomenon, it's crucial to assume a model with a sufficiently large number of parameters to fit our data. In addition, the number of parameters cannot be higher than the data size since the reduced χ^2 (χ^2_ν hereafter) is required to evaluate the effectiveness of overfitting quantitatively (According to Eq. 1, $\nu = N - p = 0$ makes χ^2_ν undefined). In our case, the data size is `len(x)=10`. The best choice of the model, consequently, is the 8th-degree polynomial, which includes nine free parameters.

- By making slight adjustments to your code, you can effectively achieve this goal.
  We merely modified the `curve_fit` section in Fig. 3 to `np.polyfit` to fit our data with an 8th-degree polynomial. The fitted coefficients are saved for plotting and reproducing. The reason why we don't continue using `curve_fit` is due to simplicity and ignoring the effect of uncertainties.

- Describe your observations based on the modifications made.
  As Fig. 26 shows, the fitted curves (green dotted lines) are messier and more complicated than Fig. 13. Those curves seem forced to approach the data points, and once the fitted models perform outside the range of the data (the leftmost and rightmost parts of the plot), the curves blow up and down drastically. These phenomena indicate that the models may:



```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_overfit.append(y)

    coefficients_high = np.polyfit(x, y, deg=8)
    coefficients.append(coefficients_high)
    fit_y = np.polyval(coefficients_high, x)
    fit_results_overfit.append(popt)
    #計算
    residuals = y - fit_y
    chi2_i = np.sum((residuals / sigma0)**2)
    chi2_values_overfit.append(chi2_i)
```

✓ 0.0s Python

Figure 25: Same coding part as Fig. 4 but the 8th-degree polynomial is used to be the fitting model.

– Fit overly well within the data, minimizing the least-squares significantly.

– Fail to extrapolate and predict functionally outside the range of the fitting data, limiting the utility a model means to provide.

The evaluation of χ_ν^2 can be used to distinguish the difference between the goodness of fit (Sec. 1.1). In this case, the just-fitted model is the linear function (Sec. 3.1), and the overfitted model is the 8th-degree polynomial. As the two reduced chi-square distributions overlay in Fig. 27, several characteristics can be found:

– The shape of the χ_ν^2 by linear model is compact, and its peak and mean are in the vicinity of the just-fit standard ($\chi_\nu^2 = 1$), i.e., the linear model may be the just-fitted model.

– The distribution of χ_ν^2 by 8th-degree polynomial (the overfitted model) morphs extremely right-skewed from that of the linear model, suggesting that the vast majority of the 8th-degree polynomials are overfitted to the data points. This is concluded by the fact that the errors are known and assumed, expelling the possibility of overestimation (Sec. 1.1).

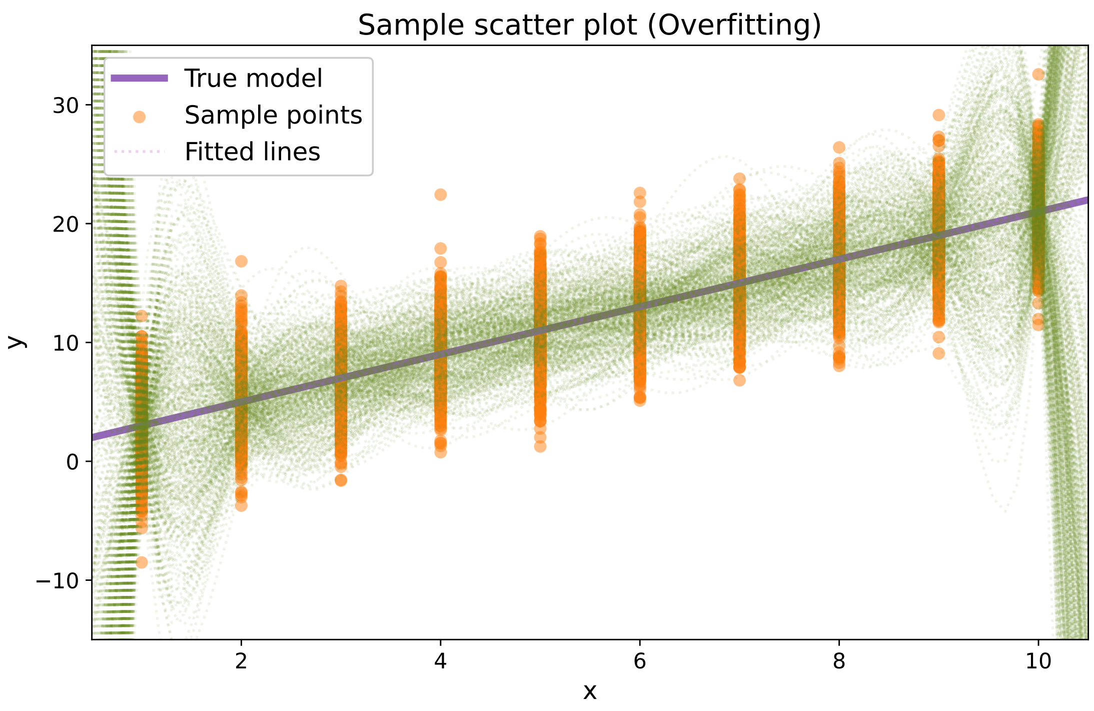


Figure 26: Same plot as the Fig. 13 but the fitted lines are 8th-degree polynomials.

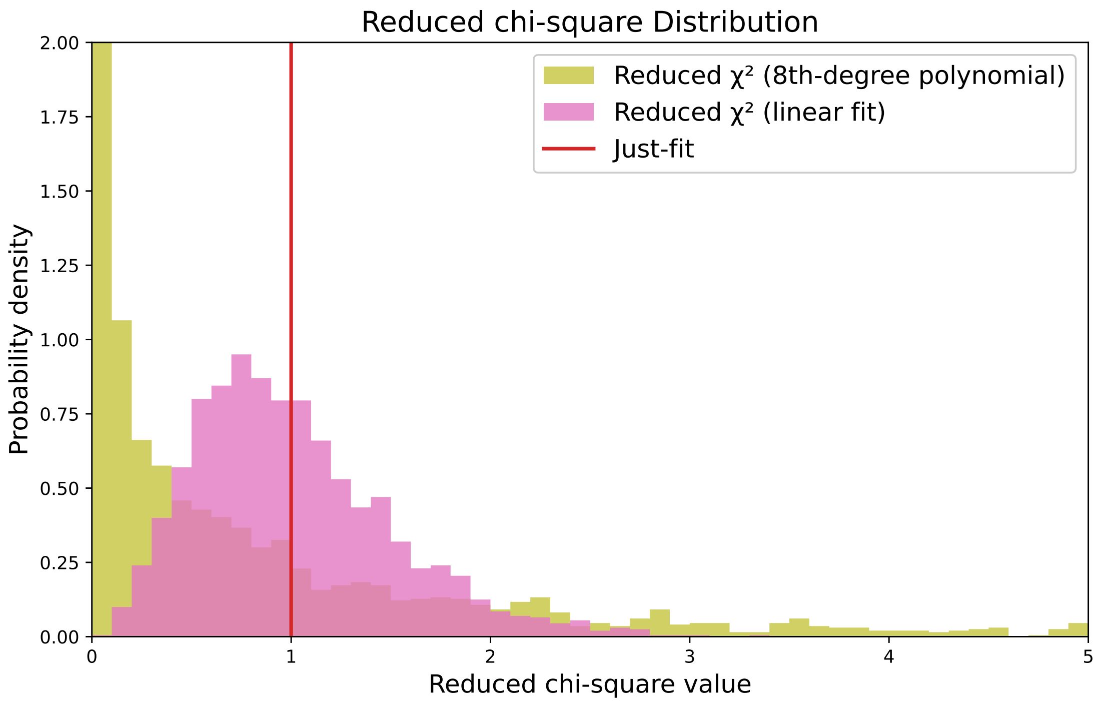


Figure 27: Reduced χ^2 distributions comparison between the two fitting functions, linear functions, and 8th-degree polynomials

- Will this phenomenon occur when analyzing the data from the General Physics Laboratory experiments, such as the pendulum or gravity experiment?

  如果我們已經有理論公式（例如單擺實驗公式：$T = 2\pi\sqrt{\frac{L}{g}}$）可讓我們做數據擬合時，過擬合的結果相對不容易出現，因爲所選模型是使用物理理論所推論的結果，較不會出現數據擬合到誤差的情況。

- Can we determine if the model is overfitting the data points obtained by unknown functions? Design your own experiment to prove your point of view.
  我們可透過**殘差分析**與**測試資料的誤差比較**判斷模型是否發生過擬合。

  1. 殘差分析：我們可藉由Fig.28看出殘差沒有趨勢，無過擬合或欠擬合的情況發生。

```python
plt.figure(figsize=(10, 6))
for i in range(5):
    y = all_data[i]
    popt = fit_results[i]
    fit_y = linear_func(x, *popt)
    residuals = y - fit_y
    plt.plot(x, residuals, marker='o', linestyle='-', alpha=0.7, label=f'Set {i+1}')

plt.axhline(0, color='black', linestyle='--', linewidth=2)
plt.title('Residuals of First 5 Fits', fontsize=16)
plt.xlabel('x', fontsize=14)
plt.ylabel('Residual (y - fit)', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True, linestyle='--', alpha=0.5)
plt.savefig("output1_3_residuals.pdf", transparent=True)
plt.show()
```

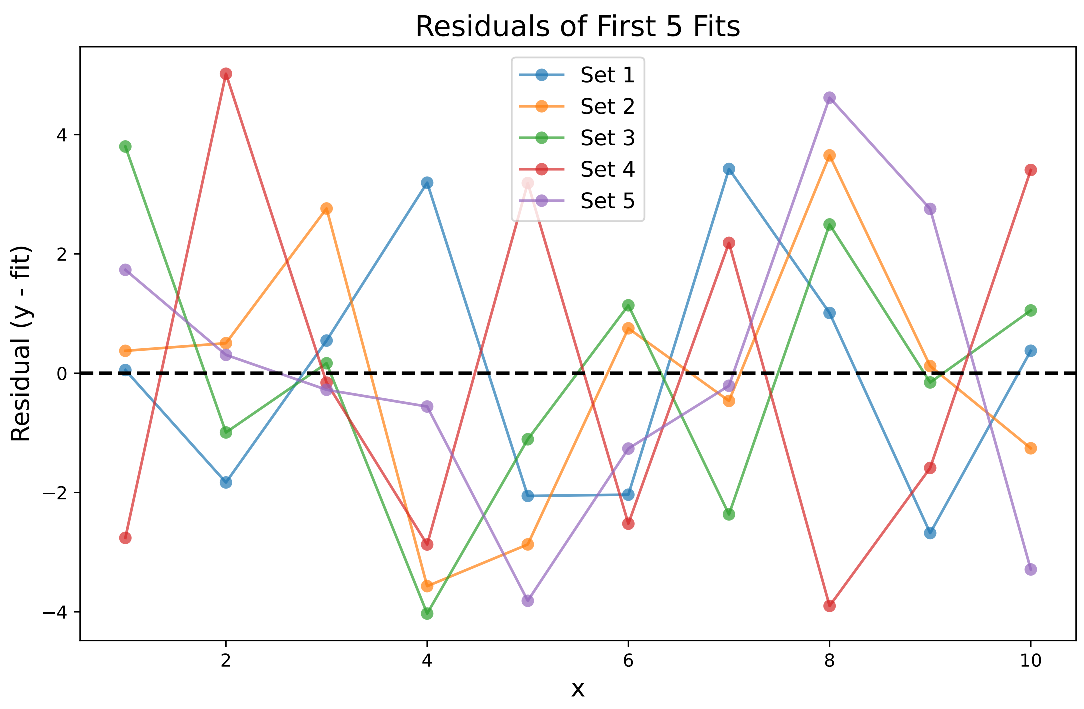
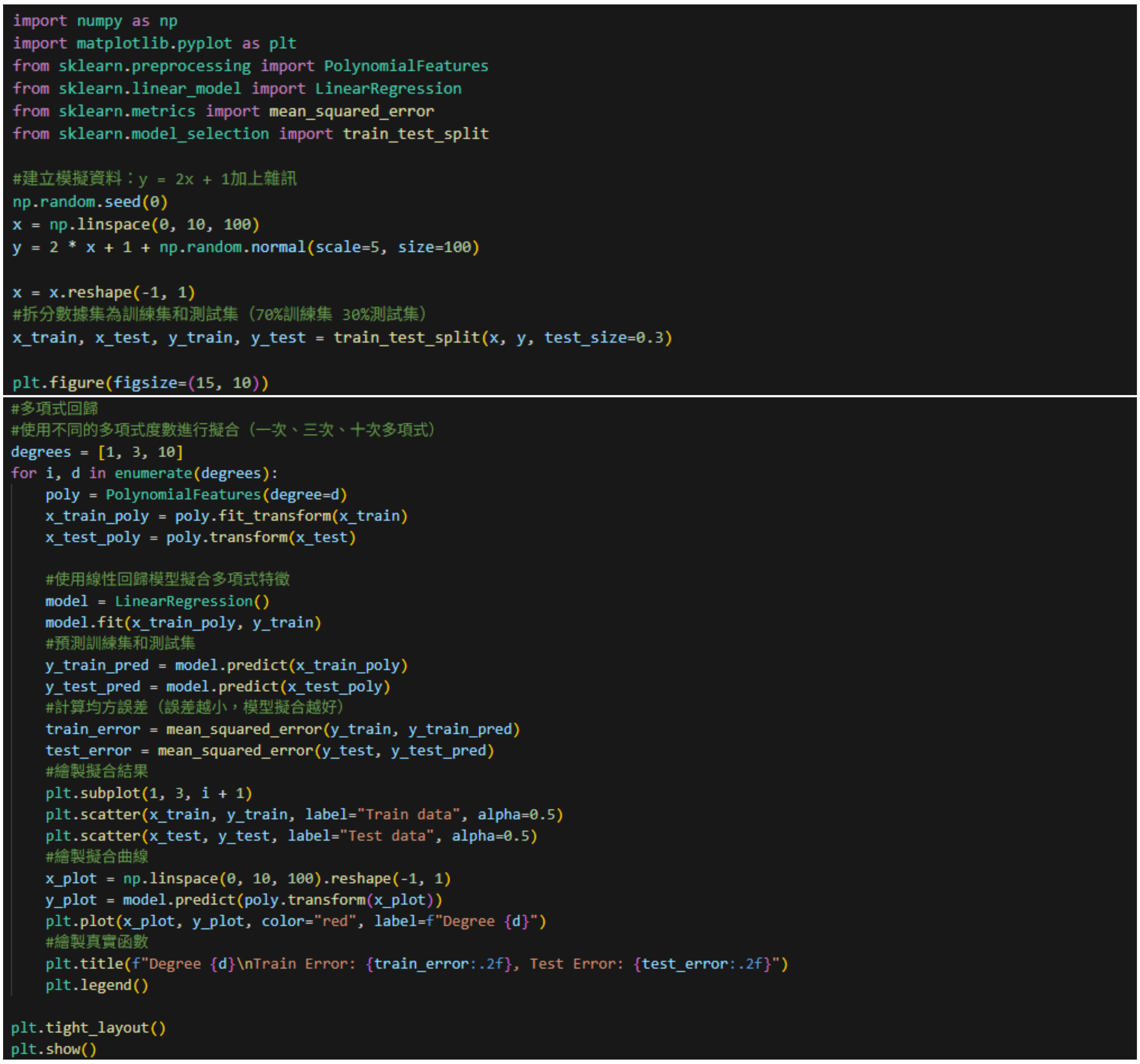


Figure 28: practice 1之殘插圖程式碼和輸出結果

2. 測試資料的誤差比較：透過先產生一組訓練資料，其為一簡單的函數加上雜訊；然後使用不同的model來擬合產生之資料，最後建立一組測試資料來比較模型在未知數據上的誤差；由此可知，當模型在訓練資料上誤差較小、而測試資料上之誤差很大，可說明實驗發生過擬合。



```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

#建立模擬資料：y = 2x + 1加上雜訊
np.random.seed(0)
x = np.linspace(0, 10, 100)
y = 2 * x + 1 + np.random.normal(scale=5, size=100)

x = x.reshape(-1, 1)
#拆分數據集為訓練集和測試集（70%訓練集 30%測試集）
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)

plt.figure(figsize=(15, 10))
#多項式回歸
#使用不同的多項式度數進行擬合（一次、三次、十次多項式）
degrees = [1, 3, 10]
for i, d in enumerate(degrees):
    poly = PolynomialFeatures(degree=d)
    x_train_poly = poly.fit_transform(x_train)
    x_test_poly = poly.transform(x_test)

    #使用線性回歸模型擬合多項式特徵
    model = LinearRegression()
    model.fit(x_train_poly, y_train)
    #預測訓練集和測試集
    y_train_pred = model.predict(x_train_poly)
    y_test_pred = model.predict(x_test_poly)
    #計算均方誤差（誤差越小，模型擬合越好）
    train_error = mean_squared_error(y_train, y_train_pred)
    test_error = mean_squared_error(y_test, y_test_pred)
    #繪製擬合結果
    plt.subplot(1, 3, i + 1)
    plt.scatter(x_train, y_train, label="Train data", alpha=0.5)
    plt.scatter(x_test, y_test, label="Test data", alpha=0.5)
    #繪製擬合曲線
    x_plot = np.linspace(0, 10, 100).reshape(-1, 1)
    y_plot = model.predict(poly.transform(x_plot))
    plt.plot(x_plot, y_plot, color="red", label=f"Degree {d}")
    #繪製真實函數
    plt.title(f"Degree {d}\nTrain Error: {train_error:.2f}, Test Error: {test_error:.2f}")
    plt.legend()

plt.tight_layout()
plt.show()
```

Figure 29: 測試資料的誤差比較之程式碼（詳情看註解）

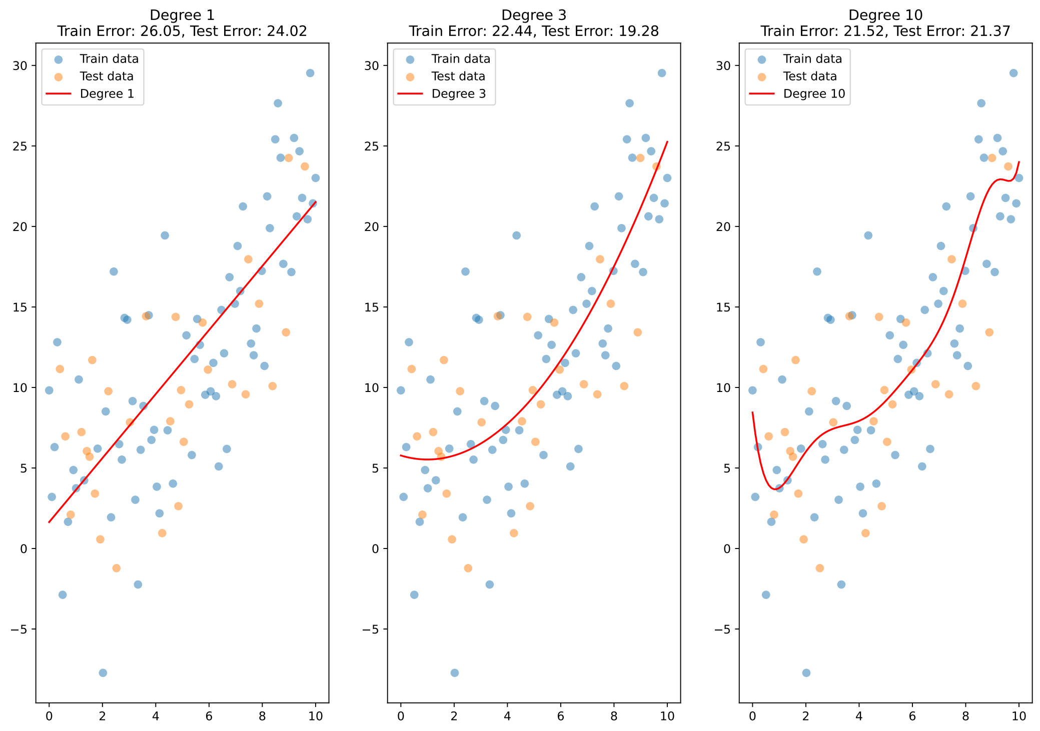


Figure 30: 測試資料的誤差比較結果

(a) 使用線性回歸模型（Fig.30左圖）
   – 訓練誤差爲26.05、測試誤差爲24.02，可看出單一直線沒辦法更細節的去描繪數據的分布。
   – 結果：模型過於簡單，發生欠擬合；訓練與測試誤差皆很大。
(b) 使用三次多項式模型（Fig.30中圖）
   – 訓練誤差爲22.44、測試誤差爲19.28，模型有畫出資料的趨勢。
   – 結果：訓練與測試誤差降低，且差距不大，可知結果爲good fit。
(c) 使用十次多項式模型（Fig.30右圖）
   – 訓練誤差爲21.52、測試誤差爲21.37，模型太過複雜，有許多波動。
   – 結果：發生過擬合，雖然在訓練誤差是最低的，但測試誤差上升表示訓練誤差所擬合到的是雜訊而不是數據的趨勢。

4.2 Practice 2

- Will a and b be closer to what you originally used to generate the data points in the section with the smaller χ^2 value?
  在χ^2值比較小的情況下，a和b會更接近於原始的a與b的值。這是因爲χ^2值小代表資料中的隨機誤差比較小，擬合效果比較好。在χ^2值小的組別中，a和b的直方圖分布相較於χ^2值大的組別窄、集中，中心值也比較接近眞實設定值，資料點與理想直線趨勢接近，誤差小，較能忠實反映眞實趨勢。因此使用χ^2 值比較小的數據就能得出較不受誤差干擾、類似於原始數據的數值。

- Is it appropriate to assess the goodness of fit based solely on the accuracy of the fitted a and b values?
  Goodness of fit應以統計方法爲主，僅依賴擬合參數的數值準確度可能會產生極大誤差。因爲參數數值接近眞實值也不代表整體的擬合情況好，如果資料的殘差分佈不符合模型假設，像是有系統誤差或者非高斯noise，整體擬合結果可能會是不好的。且單從參數值會無法看出部分問題，像次資料如果系統性的偏移或異常點，就會產生問題。而且在實際應用中，有可能會不知道眞正的a和b的值，所以也無法單純靠比對參數來評估擬合好壞。從Practice 2的結果也可以看出，即使部分的a和b的數值接近眞實值，但資料隨機誤差較大的區段擬合的不確定性依然很高，分布也會明顯變寬，因此最好還是根據統計量來評估（像是χ^2值、殘差分析或是p-value），在不知道眞實參數的時候也還是能分辨出模型是否合理。

4.3 Practice 3

- Explain how you calculate the correlation coefficient from a and b values.
  在本次實驗中，我們用了以下三種方法來計算a和b之correlation coefficient：

  1. 使用共變異數矩陣計算：
     我們從`curve_fit`函數中獲得之covarianve matrix爲：
     $$covarianve\ martix = \begin{bmatrix} \sigma_a^2 & cov(a,\ b) \\ cov(b,\ a) & \sigma_b^2 \end{bmatrix} \approx \begin{bmatrix} 0.110 & -0.620 \\ -0.620 & 4.375 \end{bmatrix}$$
     而根據相關係數之定義：
     $$R = \frac{cov(a,\ b)}{\sigma_a \sigma_b}$$
     帶入數值：
     $$R = \frac{-0.620}{\sqrt{0.110} \cdot \sqrt{4.375}} \approx -0.893$$
  2. 使用`numpy.corrcoef()`：
     透過計算所有a、b值之相關係數矩陣再度確認兩者具高度負相關。
     $$covarianve\ martix = \begin{bmatrix} 1 & -0.893 \\ -0.893 & 1 \end{bmatrix}$$

其中矩陣內各數字表示：

位置	數學式	意義
$[0,0]$	$\mathrm{corr}(a,a)=1$	a與自己的相關係數（恆爲1）
$[0,1]$	$\mathrm{corr}(a,b)=r$	a與b之相關係數（我們欲求之）
$[1,0]$	$\mathrm{corr}(b,a)=r$	b與a之相關係數（和 $[0,1]$ 相同）
$[1,1]$	$\mathrm{corr}(b,b)=1$	b與自己的相關係數（恆爲1）

Table 2: 相關係數矩陣中各元素的意義

3. 利用線性回歸分析$b=m\cdot a+c$：
   我們將a、b視爲二維資料點做線性回歸，所得斜率爲-5.624、截距爲12.242、相關係數$r=-0.893$，$R^2=0.797$表示約有79.7%之變異可藉由線性關係解釋。

此三種方式皆表示a和b之間存在負線性關係，這結果與Fig.24之2D直方圖之結果相符。

5 總結

本實驗再次以數值模擬方式，深入探討線性曲線的擬合過程，並對於擬合中的擬合參數a與b與χ^2 distribution的關係，以及透過設計實驗，觀察並了解overfitting的發生和造成的結果。 Practice 1裡，我們建立了產生資料、擬合資料、以及卡方計算等等的基本流程，並驗證了 χ^2 值符合理論的χ^2 distribution，確定了擬合結果良好。Practice 2 中，我們依照 χ^2 值的大小將資料分群，並透過分群發現了擁有較小 χ^2 數值的組別，參數（a與b）的分佈皆較擁有較大 χ^2 數值的組別集中，數值比較接近真實值，而χ^2 數值較大的組別則會看出參數分布更加發散、偏移。Practice 3 中以統計方法計算參數的平均值、標準差、共變異數與相關係數，再以 2D 分佈圖與線性回歸驗證在直線擬合中， a的增加會使 b 需要相應地減小，a 與 b 的相關係數約爲-0.963，呈高度負相關。我們也做了線性回歸分析，得到負斜率回歸線，明確證明了兩者間的負向線性關係。在 overfitting 的探討實驗中，我們以8次多項式擬合小量資料，觀察到擬合曲線在資料範圍外劇烈擺動，以及 reduced χ^2 分布右偏、拉長等特徵，驗證高自由度的模型在資料有限時容易過度擬合的問題。 本實驗再次深化了我們對曲線擬合的理解，也爲將來實驗分析提供實用的統計與實作方法。

6 分工

- 洪瑜： 實驗分析、問題討論
- 黄巧涵： 實驗分析、問題討論
- 洪懌平： 實驗分析、問題討論

7 Appendix

7.1 Source code

- https://github.com/hyp0515/exp_phy_ii/tree/main/may6