實驗物理學（二）
實驗日誌

Fundamental Python
Chi-square fitting - 1

Group 2

洪　瑜 B125090009
黃巧涵 B122030003
洪懌平 B102030019

2025/04/29

1 實驗步驟、初步結果

Practice 1

1. Generate 20 sets of data points defined by a function $y = ax + b +$ noise with `x = np.linspace(1, 10, 10)`.

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

np.random.seed(42)
```

✓ 0.6s Python

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

```python
n_sample    = 10
n_set       = 20

x = np.linspace(1, 10, n_sample, endpoint=True)
x_set = np.tile(x[np.newaxis, :], (n_set, 1))
```

✓ 0.0s Python

Figure 2: Initialize x

2. The noise is sampled from a normal distribution with mean = 0 and standard deviation $= \sigma_0 = 3$.

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

✓ 0.0s Python

```python
m = 3
k = 1
mean    = 0
sigma_0 = 3 * np.ones((n_set, n_sample))

noise       = np.random.normal(mean, sigma_0, x_set.shape)
y_true      = linear(x_set, m, k)
y_original  = y_true + noise
```

✓ 0.0s Python

Figure 3: Initialize original y

3. Change the σ_0 of y at $x = 4, 5, 6, 7$ to $\sigma_0 = 10$.

```
sigma_0_adjust = sigma_0.copy()
sigma_0_adjust[:, 3:7] = 10

y_adjust = linear(x_set, m, k) + np.random.normal(mean, sigma_0_adjust, x_set.shape)
```
✓ 0.0s Python

Figure 4: Adjust the σ_0 of y at $x = 4, 5, 6, 7$

4. You might see that the deviation of the data points at these x values is larger than the others.

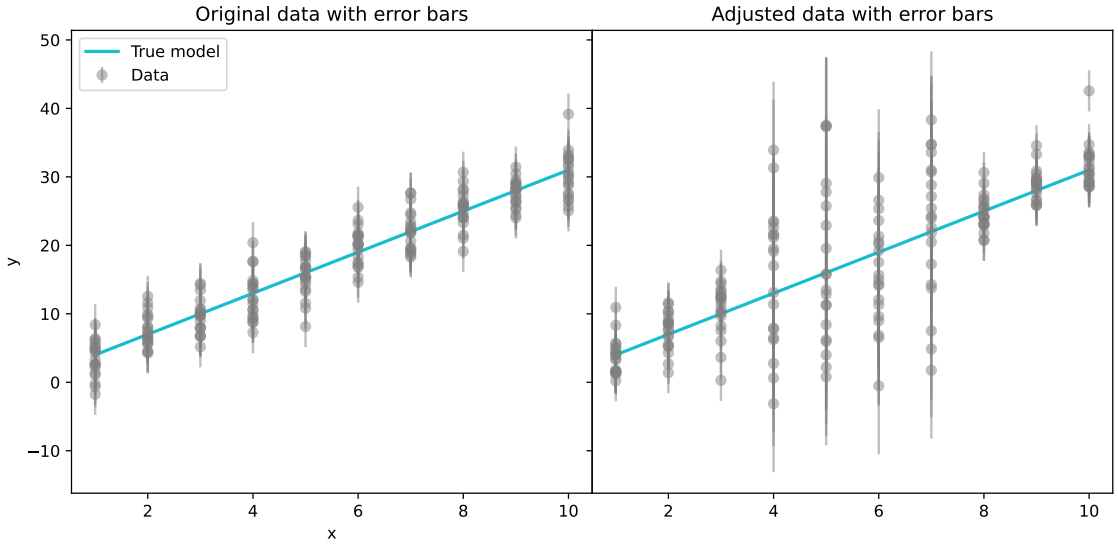

Figure 5: Comparison between the original and σ_0-adjusted y data points.

5. Get the *mean* and *standard deviation* of the data points (mean value of y might be close to $a * x + b$).

```
mean_y_adjust = np.mean(y_adjust, axis=0)
sigma_y_adjust = np.std(y_adjust, axis=0)
```
```
y_true: [ 4.  7. 10. 13. 16. 19. 22. 25. 28. 31.]
mean: [ 3.7317675   7.51018254 10.62186273 15.3878198  17.27131913 15.90692865
 22.85163387 24.72823459 28.93321474 31.62547308]
==========
std_true: [ 3.  3.  3. 10. 10. 10. 10.  3.  3.  3.]
std: [ 2.57154411  2.70636339  3.81006406  9.85004994 11.60101412  7.44949283
 10.06736762  2.48776362  2.23224275  3.11042094]
```

Figure 6: Get the *mean* and *standard deviation* of the data points

6. Fit these mean values of y with the function $y = a * x + b$ using `curve_fit`.
7. Compare the difference between the least-square fitting method and the chi-square fitting method by calculating the χ^2 value.

```
parms_chisq, cov_chisq = curve_fit(linear, x_set.flatten(), y_adjust.flatten(),
                                   sigma=sigma_0_adjust.flatten(), absolute_sigma=True)
parms_uncertainty_chisq = np.sqrt(np.diag(cov_chisq))
parms_ls, cov_ls = curve_fit(linear, x_set.flatten(), y_adjust.flatten())
parms_uncertainty_ls = np.sqrt(np.diag(cov_ls))
```

Figure 7: Fit these mean values of y with the linear function and

```
Chi-square fitting
m: 3.036747091742827 ± 0.07597980536937424
k: 1.156118712173278 ± 0.4953643977053076
chisq: 0.48853880032044295
============================================
Least-square fitting
m: 2.985233255792696 ± 0.16710284738728395
k: 1.438060752192655 ± 1.0368459071172862
chisq: 0.44692727280046485
```

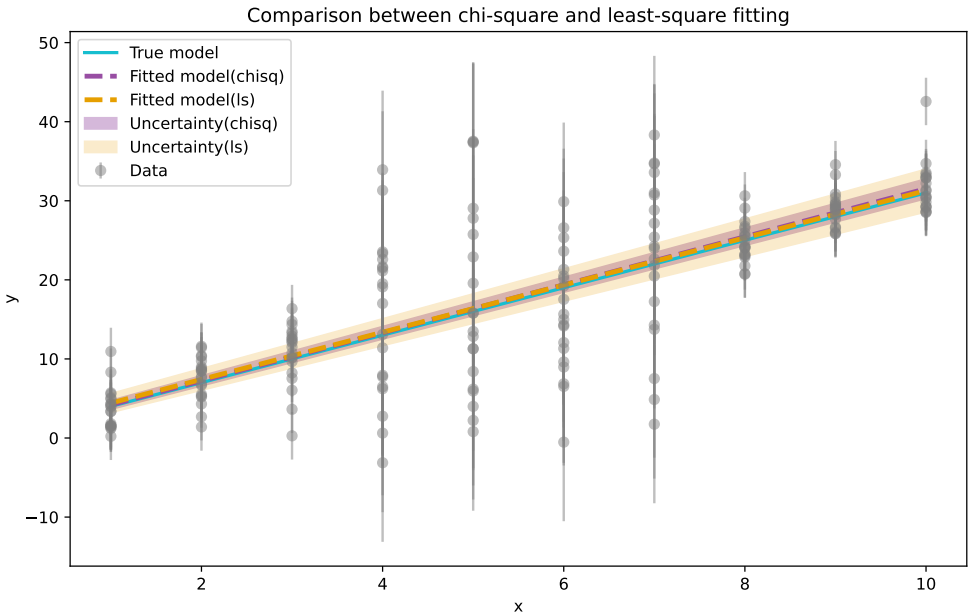

Figure 8: Comparison between least-square fitting and chi-square fitting.

Practice 2

1. Generate a 1-d array with 500 elements, each element is sampled from a normal distribution with mean = 0 and standard deviation = 1.

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

# 固定隨機數種子
np.random.seed(0)
```

Figure 9: Import packages and fix the random seed to 42

2. Calculate the chi-square value of each data point: chisq = $\frac{(\text{data\_point}-\mu)^2}{\sigma}$ You'll find out $\sigma \approx 1, \mu \approx 0$. (Note that `len(chisqr)` = 500 in this case. You might feel confused about the definition of chisqr here, but just accept it for now. You can think about what it probably means in the fitting process you've done before.)

```python
data = np.random.normal(0, 1, 500)

#計算每個點的卡方值(自由度=1)
chisqr = (data - 0)**2 / 1  # μ=0, σ=1
```

Figure 10: Initialize 500 elements and calculate the chi-square value.

3. Plot the histogram of these chi-square values.
4. Compare the histogram with the chi-square distribution with degree of freedom = 1.

```python
#畫直方圖
plt.figure(figsize=(10, 6))
plt.hist(chisqr, bins=50, density=True, alpha=0.6, label='Simulated Data')

#畫理論自由度1的卡方分布
x_vals = np.linspace(0, 10, 500)
plt.plot(x_vals, chi2.pdf(x_vals, df=1), label='Chi-square Distribution (dof=1)', color='red')

plt.xlim(left=0)
plt.xlabel('Chi-square value')
plt.ylabel('Probability Density')
plt.title('Chi-square Distribution (dof=1)')
plt.legend()
plt.savefig('./fig/chi_square_dof1.pdf', transparent=True)
plt.close('all')
```

Figure 11: Compare the histogram with the chi-square distribution with dof=1.

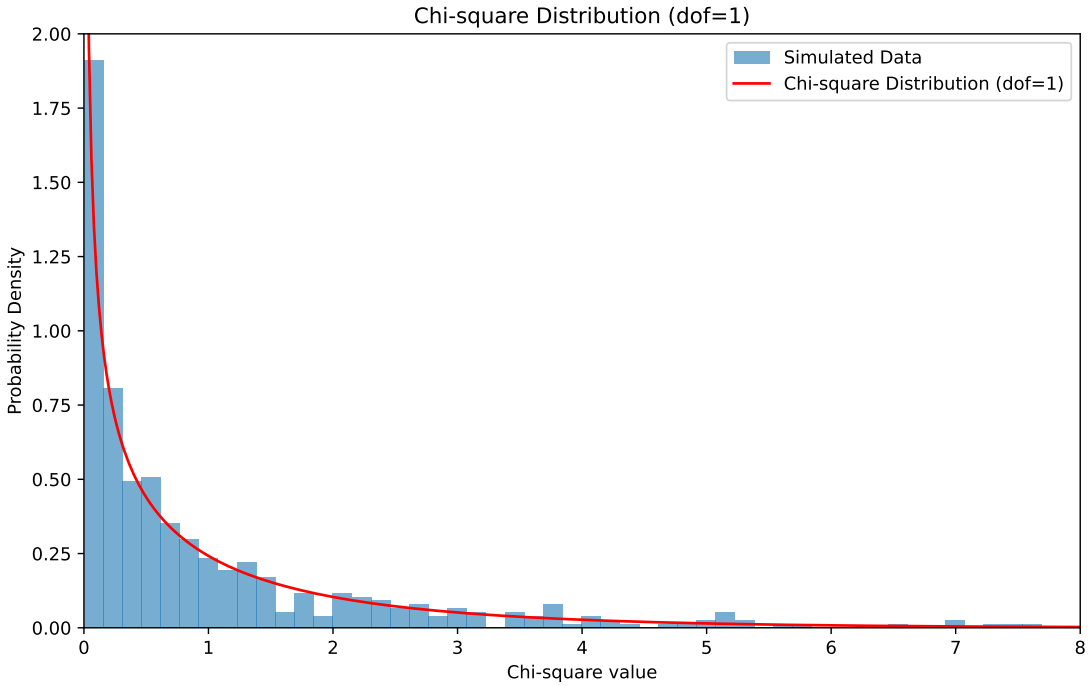

Figure 12: Histogram of the calculated chi-square values of the 500 elements with comparison of chi-square distribution with the degree of freedom = 1

5. Create several (2 to 10) 1-d arrays similar to the ones above, each array containing 500 elements sampled from a normal distribution with mean = 0 and standard deviation = 1. Then, calculate the chi-square value of each data point in each array and sum them up to get a new 1-d array that also contains 500 elements representing the sum of chi-square values. (e.g., array1 = $[1, 2, 3, \ldots, (500^{\text{th}}$ value$)]$, array2 = $[4, 5, 6, \ldots, (500^{\text{th}}$ value$)]$, sum_array = $\left(\frac{\text{array1}}{\sigma_1}\right)^2 + \left(\frac{\text{array2}}{\sigma_2}\right)^2$, `len(sum_array) = 500`)
6. Plot the histogram of these chi-square values (`sum_array`).
7. Compare the histogram with the chi-square distribution with degree of freedom equal to the number of 1-d arrays (2∼10) you generated.

```python
# 設定自由度範圍
dofs = range(2, 11)
# 建立subplots 3*3
fig, axes = plt.subplots(3, 3, figsize=(10, 9), sharex=True, sharey=True)
# 調整子圖間距
plt.subplots_adjust(left=0.03, right=0.97, top=0.95, bottom=0.05, wspace=0.0, hspace=0.0)
x_vals = np.linspace(0, 30, 1000)
# 遍歷每個自由度與子圖
for idx, dof in enumerate(dofs):
    row = idx // 3
    col = idx % 3
    ax = axes[row, col]

    # 模擬卡方資料
    sum_chisqr = np.zeros(500)
    for _ in range(dof):
        data = np.random.normal(0, 1, 500)
        sum_chisqr += data**2

    # 畫直方圖
    ax.hist(sum_chisqr, bins=50, density=True, alpha=0.6, label='Simulated Data')
    # 畫理論曲線
    ax.plot(x_vals, chi2.pdf(x_vals, df=dof), color='red', label='Chi-square PDF')

    ax.set_xlim(0, 30)
    ax.set_title(f'dof = {dof}')
    if dof == 4:
        ax.legend()
    if dof == 8:
        ax.set_xlabel('Chi-square value')
        ax.set_ylabel('Density')

plt.suptitle('Chi-square Distributions (dof = 2 to 10)', fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.96])  # 調整 layout 以容納 suptitle
# plt.show()
plt.savefig('./fig/output_2_1.pdf', transparent=True)
```

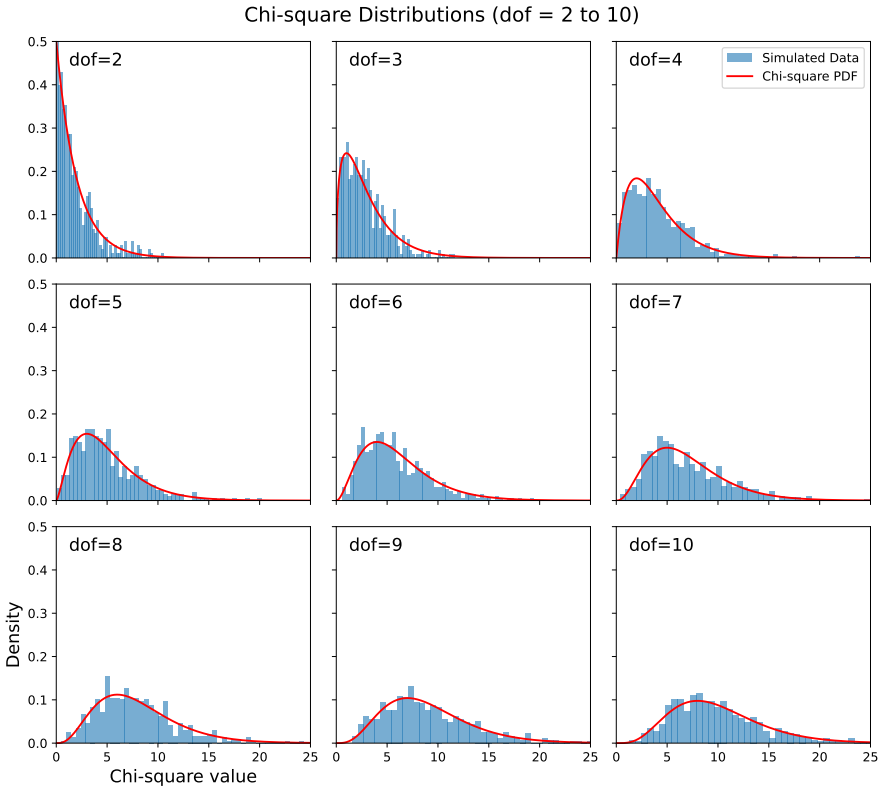

Figure 13: Compare the histogram with the chi-square distribution with different degrees of freedom.

2 初步分析

Practice 1

資料爲y=a*x+b加上高斯分布生成的雜訊，在x=4-7間加入較大的誤差（Fig.5），而在Fig.8中可見這個區間的error bars明顯變長，表不確定度變大（也可由比較黃色框框和紫色框框看見），符合理論預期。

而延續上次與前次練習，可透過reduced χ^2來判斷擬合的好壞，我們的自由度爲$\nu = 10 - 2 = 8$，各擬合方法的χ^2_ν如下：

- Least-square fitting：

$$\chi^2_{\nu,\,LS} = \frac{0.489}{8} \approx 0.061$$

- Chi-square fitting：

$$\chi^2_{\nu,\,chisq} = \frac{0.447}{8} \approx 0.056$$

先前所推測的：

- $\chi^2_\nu \approx 1$爲理想
- $\chi^2_\nu \ll 1$爲過擬合
- $\chi^2_\nu \gg 1$爲欠擬合

的結果可推論出：雖兩擬合結果皆遠小於1，雖差異很小，但χ^2 fitting的數值較接近1，顯示其較符合預期。

Practice 2

由Fig.12 可見，模擬結果與理論曲線近乎重合；並且由Fig.13 可知，當自由度上升時，曲線的峰值將會往右偏移，且涵蓋範圍更廣，可推測自由度與χ^2有關聯。

3 具體說明實驗遇到的問題，或分析可能的問題

1. 在練習二中，PDF(probability density function)的線一開始輸出的時候不會跑到底（x=30的位置）。
   解法：調整參數，從原本`x_vals = np.linspace(0, 10, 1000)`修正成`x_vals = np.linspace(0, 30, 1000)`
   補充：參數的意義
   `np.linspace(a, b, c)`是用來產生一個等間距的數值序列，三個參數的意義：
   - a：序列的起始值（包含這個數字）
   - b：序列的結束值（包含這個數字）
   - c：在這個範圍內總共產生1000個等距的點

   我們改變參數b，使結果更符合預期。
2. 在練習二中，若將圖片分別繪出時，將會產生9個視窗，略顯煩躁。
   解法：透果引入`matplotlib.pyplot.subplots()`這個程式碼，匯集九張圖片，使結果輸出更爲清爽。