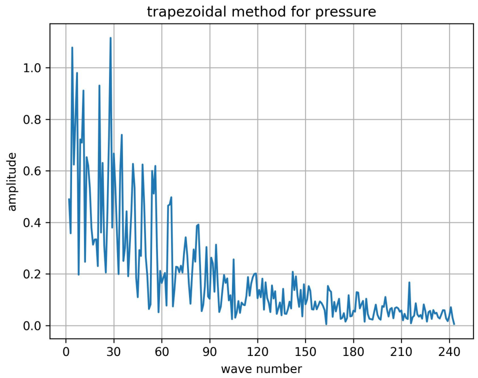
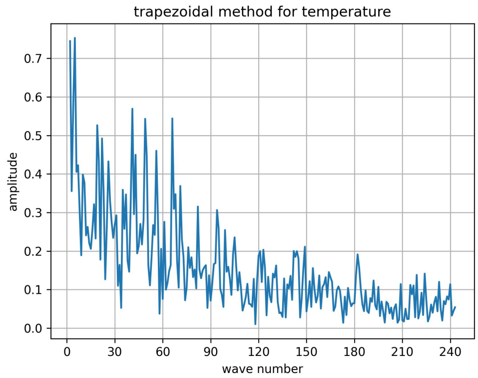
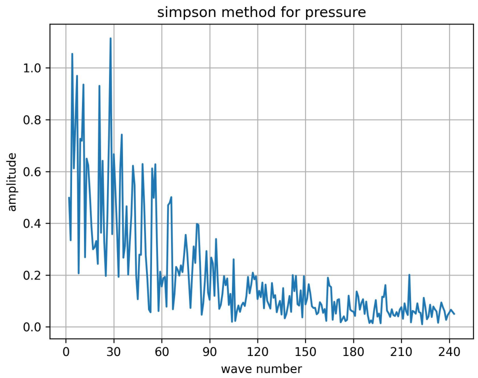
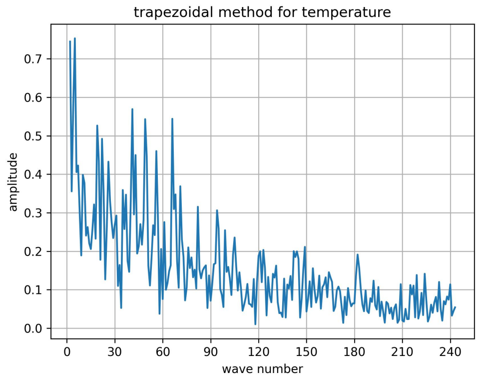
1121 天氣學與天氣分析（下） --- 作業四

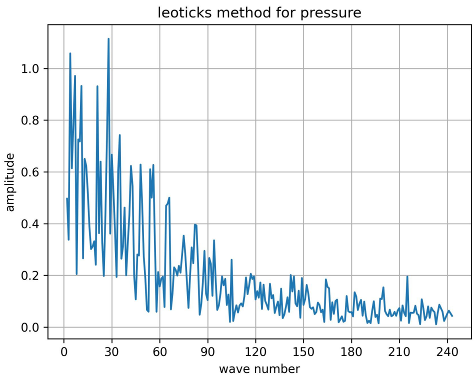
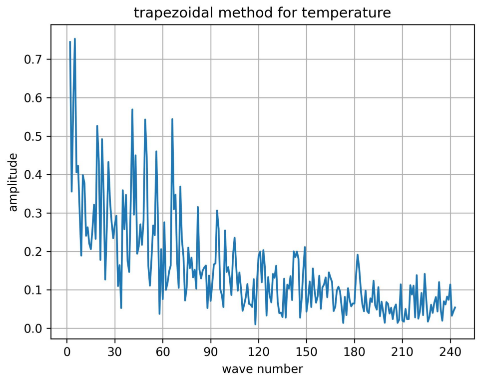
> - 姓名：林群賀
> - 系級：大氣四
> - 學號：109601003

執行程式碼

```
$ python3 main.py
```

(一) 求出每個波的 power spectral 並作圖，比較三種數值 方法之結果差異與使用上的優缺點

method	Pressure	Temperature
Trapezoidal		
Simpson		

method	Pressure	Temperature
Leoticks		

1. 梯形法在計算時沒有資料點數量的限制，而辛普森法和 Leo Ticks 方法需要資料點為 0 到 2N 中奇數個，且區間為偶數個。
2. 梯形法相較於後兩者有較大的誤差，而辛普森法和 Leo Ticks 方法的誤差相近。
3. 在計算相位時，如果結果為正值，會減去 2pi 以避免出現負數的日期。
4. 辛普森法對於曲線變化較大的函數有更好的逼近能力和較高的精確度，特別對於凹凸型的函數，在某些情況下需要更小的子區間才能達到相同的準確度，而梯形法可能需要更多子區間。
5. 梯形法較為簡單易懂、實作容易，適用於一般函數；而辛普森法則更適用於平滑且具有較高次導數的函數。

(二) 計算 power 最大的前五個主波之振幅、相位與日期

溫度

Trapezoida

項目 / 波數	震幅	相位	日期
1	7.27	-0.45	27
5	0.75	-0.3	1
2	0.74	-0.05	1
4	0.58	-5.42	20
41	0.57	-0.63	1

Simpson

項目 / 波數	震幅	相位	日期
1	7.18	-0.45	27
5	0.75	-0.04	1
2	0.75	-0.26	1
4	0.59	-5.39	20

項目 / 波數	震幅	相位	日期
41	0.58	-0.64	1

Leo Ticks

項目 / 波數	震幅	相位	日期
1	7.19	-0.45	27
5	0.75	-0.04	1
2	0.75	-0.27	1
4	0.59	-5.39	20
41	0.58	-0.64	1

壓力

Trapezoida

項目 / 波數	震幅	相位	日期
1	7.51	-0.2	12
28	1.12	-1.12	1
4	1.08	-0.32	2
7	0.98	-0.35	1
21	0.93	-5.56	1

Simpson

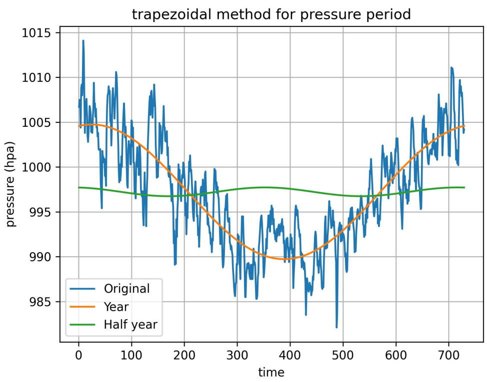
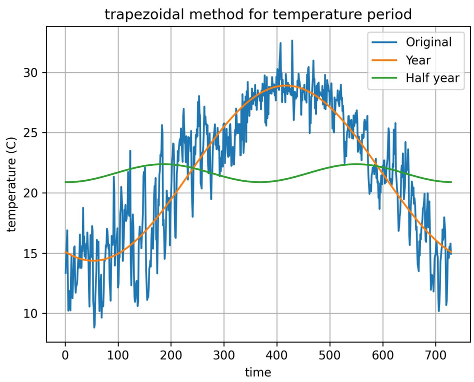
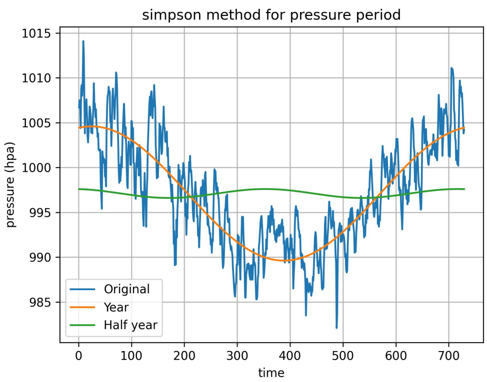
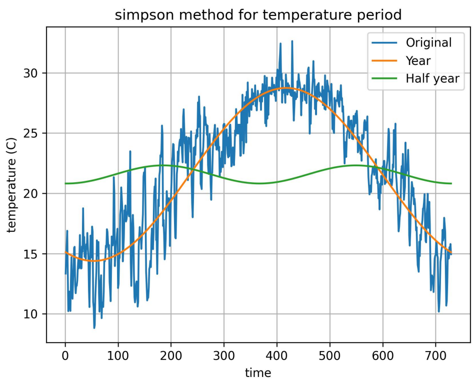
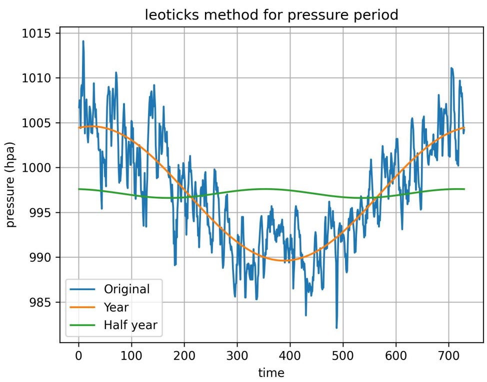
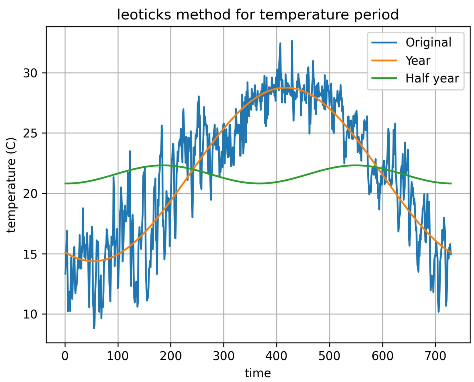
項目 / 波數	震幅	相位	日期
1	7.48	-0.2	12
28	1.11	-1.15	1
4	1.05	-0.32	2
7	0.97	-0.34	1
21	0.94	-1.01	1

Leo Ticks

項目 / 波數	震幅	相位	日期
1	7.49	-0.2	12

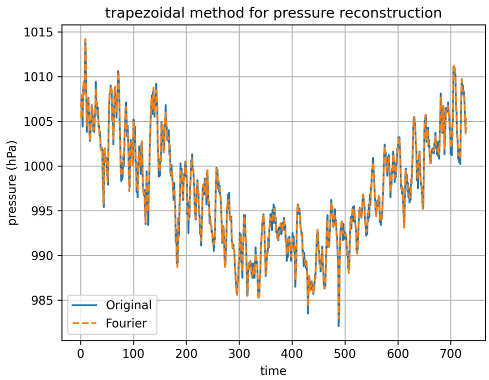
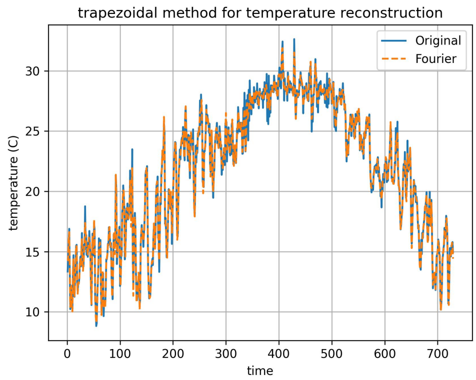
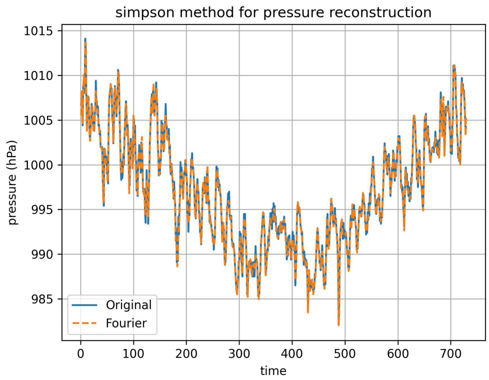
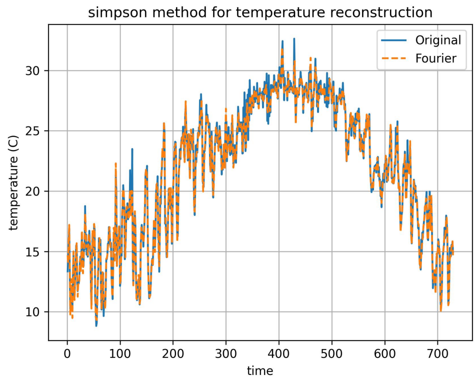
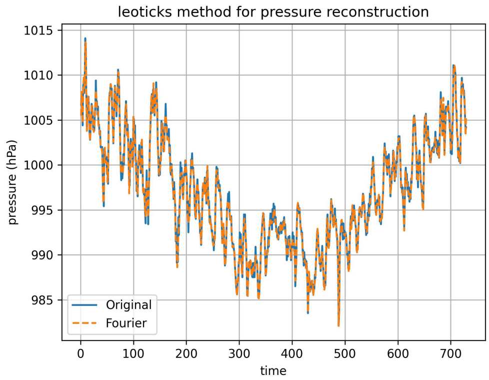
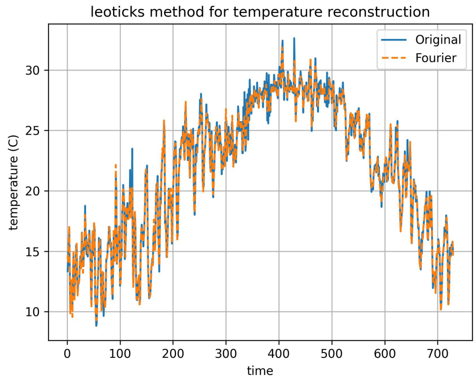
項目 / 波數	震幅	相位	日期
28	1.11	-1.14	1
4	1.06	-0.32	2
7	0.97	-0.35	1
21	0.93	-1.01	1

(三) 繪出年週期與半年周期的圖形

<table>
<tr><th>method</th><th>Pressure</th><th>Temperature</th></tr>
<tr><td>Trapezoidal</td><td>
</td><td>
</td></tr>
<tr><td>Simpson</td><td>
</td><td>
</td></tr>
<tr><td>Leoticks</td><td>
</td><td>
</td></tr>
</table>

透過傅立葉還原：

method	Pressure	Temperature

method	Pressure	Temperature
Trapezoidal		
Simpson		
Leoticks		

程式碼

Import packages

```python
import os
import math
import numpy as np
import matplotlib.pyplot as plt
```

Load Data

```
```

```
def read_file(file_path):
    data = []

    with open(file_path, 'r') as file:
        for line in file:
            number = float(line.strip())
            data.append(number)

    return data
```

Trapzoidal

```
def count_fourier_trapezoidal(data):
    n = 243
    pi = np.pi
    h = 2 * pi / (len(data) - 1)
    a0 = 0
    an = []
    bn = []

    # count a0
    for i in range(len(data) - 1):
        a0 += ((data[i] + data[i + 1]) / 2) * h / (2 * pi)

    # count an
    for i in range(n):
        value = 0
        x = np.linspace(-pi, pi, len(data))
        for j in range(len(data) - 1):
            value += (
                ((data[j] * np.cos((i+1) * x[j]) + data[j+1] *
np.cos((i+1) * x[j+1])) / 2) * h / pi
            )
        an.append(value)

    # 計算bn
    for i in range(n):
        value = 0
        x = np.linspace(-pi, pi, len(data))
        for j in range(len(data) - 1):
            value += (
                ((data[j] * np.sin((i+1) * x[j]) + data[j+1] *
np.sin((i+1) * x[j+1])) / 2) * h / pi
            )
        bn.append(value)

    return (
        a0,
        an,
```

```
        bn,
    )
```

Simpson

```
def count_fourier_simpson(data):
    n = 243
    pi = np.pi
    h = 2 * pi / (len(data)-1)
    a0 = 0
    an = []
    bn = []

    # count a0
    for i in range(0,len(data)-2,2):
        a0 += ((data[i] + 4*data[i+1]+data[i+2])/3) * h / (2 * pi)

    # count an
    for i in range(n):
        value = 0
        x = np.linspace(-pi, pi, len(data))
        for j in range(0,len(data)-2,2):
            value += (data[j] * np.cos((i+1)*x[j]) + 4*data[j+1] *
np.cos((i+1)*x[j+1])+data[j+2] * np.cos((i+1)*x[j+2]))/3*h/pi
        an.append(value)

    # count bn
    for i in range(n):
        value = 0
        x = np.linspace(-pi, pi, len(data))
        for j in range(0,len(data)-2,2):
            value += (data[j] * np.sin((i+1)*x[j]) + 4*data[j+1] *
np.sin((i+1)*x[j+1])+data[j+2] * np.sin((i+1)*x[j+2]))/3*h/pi
        bn.append(value)


    return (
        a0,
        an,
        bn,
    )
```

Leo Ticks

```
def count_fourier_leoticks(data):
    n = 243
    pi = np.pi
    h = 2 * pi / (len(data)-1)
    a0 = 0
```

```
    an = []
    bn = []

    # count a0
    for i in range(0,len(data)-2,2):
        a0 += ((data[i] + 4*data[i+1]+data[i+2])/3) * h / (2 * pi)

    # count an
    for i in range(n):
        value = 0
        x = np.linspace(-pi, pi, len(data))
        for j in range(0,len(data)-2,2):
            value += (1.0752*data[j] * np.cos((i+1)*x[j]) +
3.8496*data[j+1] * np.cos((i+1)*x[j+1])+1.0752*data[j+2] *
np.cos((i+1)*x[j+2]))/3*h/pi
        an.append(value)

    # count bn
    for i in range(n):
        value = 0
        x = np.linspace(-pi, pi, len(data))
        for j in range(0,len(data)-2,2):
            value += (1.0752*data[j] * np.sin((i+1)*x[j]) +
3.8496*data[j+1] * np.sin((i+1)*x[j+1])+1.0752*data[j+2] *
np.sin((i+1)*x[j+2]))/3*h/pi
        bn.append(value)

    return (
        a0,
        an,
        bn,
    )
```

Plot

```
def calculate_power_spectrum_and_plot(
        an,
        bn,
        title,
        type
    ):
    os.makedirs(
        f"./imgs/fourier_{type}",
        exist_ok=True
    )
    title_name = f"{title} method for {type}"
    file_name = f"./imgs/fourier_{type}/{type}_spectrum_{title}.jpg"

    x = np.linspace(1, 243, 243)
    fx = np.zeros(len(an))
```

```
    for n in range(243):
        fx[n] = (an[n] ** 2 + bn[n] ** 2) ** 0.5

    plt.figure()
    plt.title(title_name)
    plt.xticks(np.arange(min(x)-1, max(x), 30))
    plt.xlabel("wave number")
    plt.ylabel("amplitude")
    plt.plot(x[1:], fx[1:])
    plt.grid('--')
    plt.savefig(file_name, dpi=300)
    # plt.show()

    return fx


def plot_fourier_analysis_with_half_year(
        a0,
        an,
        bn,
        data,
        title,
        type
    ):
    os.makedirs(
        f"./imgs/fourier_{type}",
        exist_ok=True
    )
    title_name = f"{title} method for {type} period"
    file_name = f"./imgs/fourier_{type}/{type}_period_{title}.jpg"

    time = np.linspace(1, 729, 729)
    pi = np.pi
    x = np.linspace(-pi, pi, 729)

    plt.figure(dpi=300)
    plt.plot(
        time,
        data,
        label='Original'
    )
    fx =  np.zeros((2, 729))

    for i in range(2):
        for j in range(729):
            value = a0 + an[i] * np.cos((i + 1) * x[j]) + bn[i] *
np.sin((i + 1) * x[j])
            fx[i, j] = value

    plt.plot(
        time,
        fx[0,:],
        label="Year"
```

```
    )
    plt.plot(
        time,
        fx[1,:],
        label="Half year"
    )

    if type == "temperature":
        plt.ylabel("temperature (C)")
    elif type == "pressure":
        plt.ylabel("pressure (hpa)")
    plt.legend()
    plt.xlabel("time")
    plt.title(title_name)
    plt.grid('--')
    plt.savefig(file_name, dpi=300)
    # plt.show()


def calculate_extreme_values_and_properties(
        fx,
        an,
        bn
    ):
    indexed_lst = list(enumerate(fx))

    # 使用sorted將元組列表按值進行降序排序
    sorted_lst = sorted(
        indexed_lst,
        key=lambda x: x[1],
        reverse=True
    )

    top_five_indices = [index for index, value in sorted_lst[:5]]
    incremented_indices = [index + 1 for index in top_five_indices]

    amplitude = []
    phase = []
    date = []


    for i in top_five_indices:
        amplitude.append(
            round(
                np.sqrt(an[i]**2+bn[i]**2),
                2
            )
        )
        if np.arctan(-bn[i]/an[i])>0:
            phase.append(
                round(
                    np.arctan(-bn[i]/an[i])-2*np.pi,
                    2
                )
```

```
            )
        else:
            phase.append(
                round(
                    np.arctan(-bn[i]/an[i]),
                    2
                )
            )
        if np.arctan(-bn[i]/an[i])>0:
            value = -(np.arctan(-
bn[i]/an[i])-2*np.pi)/(i+1)/(2*np.pi)*365/(i+1)
        else:
            value = -(np.arctan(-bn[i]/an[i]))/(i+1)/(2*np.pi)*365/(i+1)
        date.append(math.ceil(value))

    print("Incremented indices")
    print(incremented_indices)

    print("Amplitude")
    print(amplitude)

    print("Phase")
    print(phase)

    print("Date")
    print(date)


def reconstruct_from_fourier_analysis(
        a0,
        an,
        bn,
        data,
        title,
        type
    ):
    os.makedirs(
        f"./imgs/fourier_{type}",
        exist_ok=True
    )

    title_name = f"{title} method for {type} reconstruction"
    file_name = f"./imgs/fourier_{type}/{type}_reconstruction_{title}.jpg"

    time = np.linspace(1,729,729)
    pi = np.pi
    x = np.linspace(-pi,pi,729)
    fx =  np.zeros(729) + a0

    plt.figure(dpi=400)
    plt.plot(
        time,
        data,
        label='Original'
```

```
    )

    for i in range(243):
        for j in range(729):
            value = an[i]*np.cos((i+1)*x[j])+bn[i]*np.sin((i+1)*x[j])
            fx[j] += value

    plt.plot(
        time,
        fx,
        label='Fourier',
        linestyle="--"
    )
    plt.legend()

    if type == "temperature":
        plt.ylabel("temperature (C)")
    elif type == "pressure":
        plt.ylabel("pressure (hPa)")

    plt.xlabel("time")
    plt.title(title_name)
    plt.grid('--')
    plt.savefig(file_name, dpi=300)
    # plt.show()
```

Deal with Pressure

```
def pressure_analysis() -> None:

    file_path = "./data/Ps.dat.txt"
    pressure = read_file(file_path)

    (   # 使用梯形法進行傅立葉分析
        a0_pressure_trapezoidal,
        an_pressure_trapezoidal,
        bn_pressure_trapezoidal
    ) = count_fourier_trapezoidal(pressure)
    (   # 使用辛普森法進行傅立葉分析
        a0_pressure_simpson,
        an_pressure_simpson,
        bn_pressure_simpson
    ) = count_fourier_simpson(pressure)
    (   # 使用Leoticks方法進行傅立葉分析
        a0_pressure_leoticks,
        an_pressure_leoticks,
        bn_pressure_leoticks
    ) = count_fourier_leoticks(pressure)

    spectral_trapezoidal = calculate_power_spectrum_and_plot(
        an_pressure_trapezoidal,
```

```
        bn_pressure_trapezoidal,
        "trapezoidal",
        "pressure"
    )  # 計算梯形法的功率頻譜
    spectral_simpson = calculate_power_spectrum_and_plot(
        an_pressure_simpson,
        bn_pressure_simpson,
        "simpson",
        "pressure"
    )  # 計算辛普森法的功率頻譜
    spectral_leoticks = calculate_power_spectrum_and_plot(
        an_pressure_leoticks,
        bn_pressure_leoticks,
        "leoticks",
        "pressure"
    )  # 計算Leoticks方法的功率頻譜

    plot_fourier_analysis_with_half_year(
        a0_pressure_trapezoidal,
        an_pressure_trapezoidal,
        bn_pressure_trapezoidal,
        pressure,
        "trapezoidal",
        "pressure"
    )  # 繪製梯形法的波形圖
    plot_fourier_analysis_with_half_year(
        a0_pressure_simpson,
        an_pressure_simpson,
        bn_pressure_simpson,
        pressure,
        "simpson",
        "pressure"
    )  # 繪製辛普森法的波形圖
    plot_fourier_analysis_with_half_year(
        a0_pressure_leoticks,
        an_pressure_leoticks,
        bn_pressure_leoticks,
        pressure,
        "leoticks",
        "pressure"
    )  # 繪製Leoticks方法的波形圖


    calculate_extreme_values_and_properties(
        spectral_trapezoidal,
        an_pressure_trapezoidal,
        bn_pressure_trapezoidal
    )
    calculate_extreme_values_and_properties(
        spectral_simpson,
        an_pressure_simpson,
        bn_pressure_simpson
    )
    calculate_extreme_values_and_properties(
```

```
        spectral_leoticks,
        an_pressure_leoticks,
        bn_pressure_leoticks
    )


    reconstruct_from_fourier_analysis(
        a0_pressure_trapezoidal,
        an_pressure_trapezoidal,
        bn_pressure_trapezoidal,
        pressure,
        "trapezoidal",
        "pressure"
    )  # 繪製梯形法的波形圖
    reconstruct_from_fourier_analysis(
        a0_pressure_simpson,
        an_pressure_simpson,
        bn_pressure_simpson,
        pressure,
        "simpson",
        "pressure"
    )  # 繪製辛普森法的波形圖
    reconstruct_from_fourier_analysis(
        a0_pressure_leoticks,
        an_pressure_leoticks,
        bn_pressure_leoticks,
        pressure,
        "leoticks",
        "pressure"
    )  # 繪製Leoticks方法的波形圖
```

Deal with Temperature

```
def temperature_analysis() -> None:
    file_path = "./data/T.dat.txt"
    temparature = read_file(file_path)


    (   # 使用梯形法進行傅立葉分析
        a0_temperature_trapezoidal,
        an_temperature_trapezoidal,
        bn_temperature_trapezoidal
    ) = count_fourier_trapezoidal(temparature)
    (   # 使用辛普森法進行傅立葉分析
        a0_temperature_simpson,
        an_temperature_simpson,
        bn_temperature_simpson
    ) = count_fourier_simpson(temparature)
    (   # 使用Leoticks方法進行傅立葉分析
        a0_temperature_leoticks,
        an_temperature_leoticks,
```

```
        bn_temperature_leoticks
    ) = count_fourier_leoticks(temparature)


    spectral_trapezoidal = calculate_power_spectrum_and_plot(
        an_temperature_trapezoidal,
        bn_temperature_trapezoidal,
        "trapezoidal",
        "temperature"
    )  # 計算梯形法的功率頻譜
    spectral_simpson = calculate_power_spectrum_and_plot(
        an_temperature_simpson,
        bn_temperature_simpson,
        "simpson",
        "temperature"
    )  # 計算辛普森法的功率頻譜
    spectral_leoticks = calculate_power_spectrum_and_plot(
        an_temperature_leoticks,
        bn_temperature_leoticks,
        "leoticks",
        "temperature"
    )  # 計算Leoticks方法的功率頻譜

    plot_fourier_analysis_with_half_year(
        a0_temperature_trapezoidal,
        an_temperature_trapezoidal,
        bn_temperature_trapezoidal,
        temparature,
        "trapezoidal",
        "temperature"
    )  # 繪製梯形法的波形圖
    plot_fourier_analysis_with_half_year(
        a0_temperature_simpson,
        an_temperature_simpson,
        bn_temperature_simpson,
        temparature,
        "simpson",
        "temperature"
    )  # 繪製辛普森法的波形圖
    plot_fourier_analysis_with_half_year(
        a0_temperature_leoticks,
        an_temperature_leoticks,
        bn_temperature_leoticks,
        temparature,
        "leoticks",
        "temperature"
    )  # 繪製Leoticks方法的波形圖


calculate_extreme_values_and_properties(spectral_trapezoidal,an_temperature_trapezoidal,bn_temperature_trapezoidal)

calculate_extreme_values_and_properties(spectral_simpson,an_temperature_simpson,bn_temperature_simpson)
```

```
calculate_extreme_values_and_properties(spectral_leoticks,an_temperature_l
eoticks,bn_temperature_leoticks)

    reconstruct_from_fourier_analysis(
        a0_temperature_trapezoidal,
        an_temperature_trapezoidal,
        bn_temperature_trapezoidal,
        temparature,
        "trapezoidal",
        "temperature"
    )  # 繪製梯形法的波形圖
    reconstruct_from_fourier_analysis(
        a0_temperature_simpson,
        an_temperature_simpson,
        bn_temperature_simpson,
        temparature,
        "simpson",
        "temperature"
    )  # 繪製辛普森法的波形圖
    reconstruct_from_fourier_analysis(
        a0_temperature_leoticks,
        an_temperature_leoticks,
        bn_temperature_leoticks,
        temparature,
        "leoticks",
        "temperature"
    )  # 繪製Leoticks方法的波形圖
```

Main

```
def main() -> None:
    pressure_analysis()
    temperature_analysis()


if __name__ == "__main__":
    main()
```