天氣學下 hw4　　　大氣 4A　黃展皇　　106601015

工作環境：x64 windows10，conda 4.8.3，python3.6.10

Requestment：os、numpy、matplotlib

1. Power spectral 圖(without wave 1)

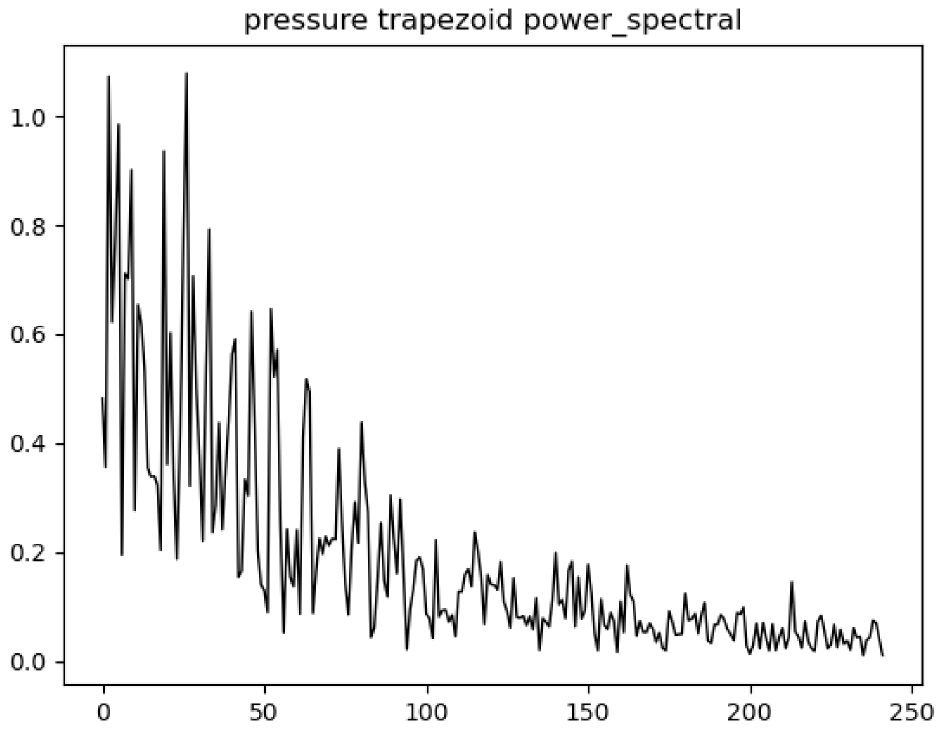

圖 1-1　氣壓梯形法

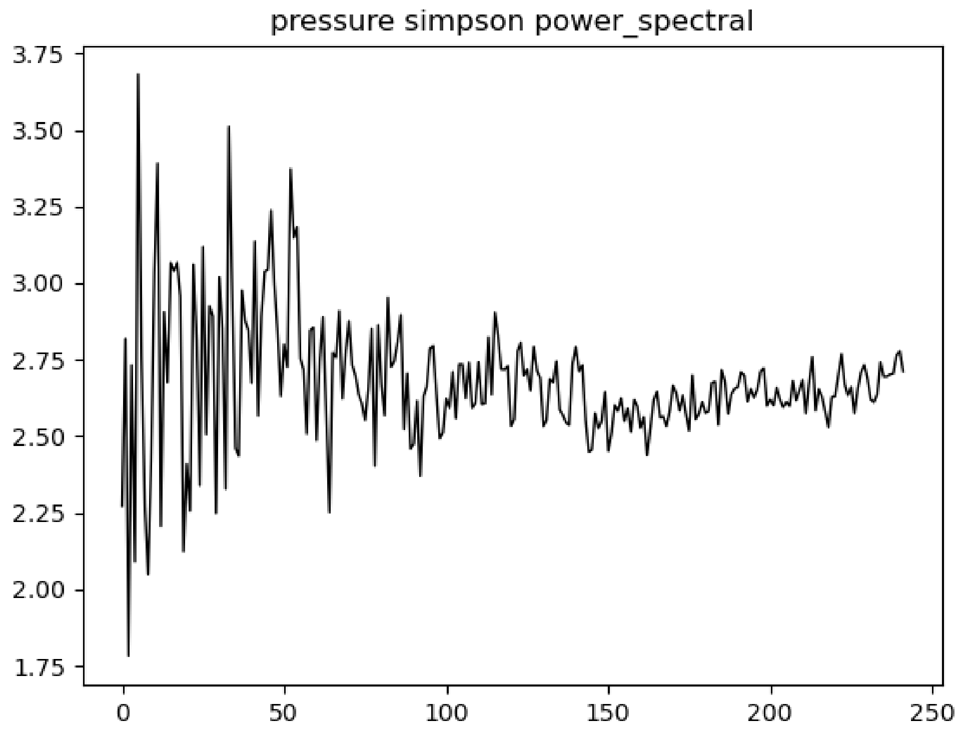

圖 1-2　氣壓 Simpson’s rule 法

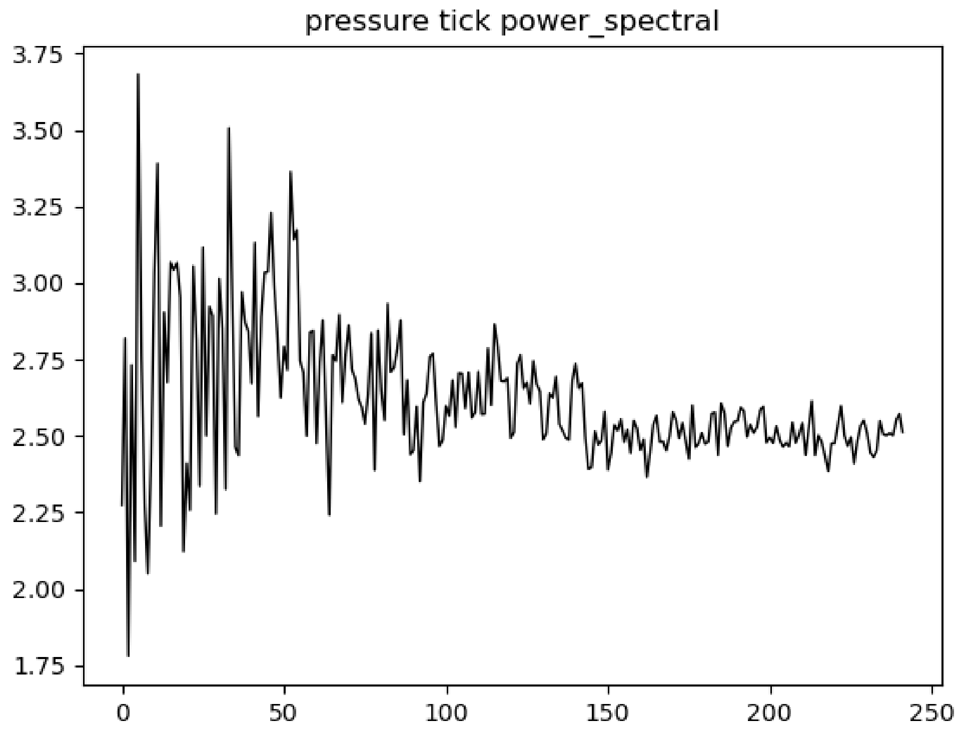

圖 1-3　氣壓 The Leo Tick’s formula 法

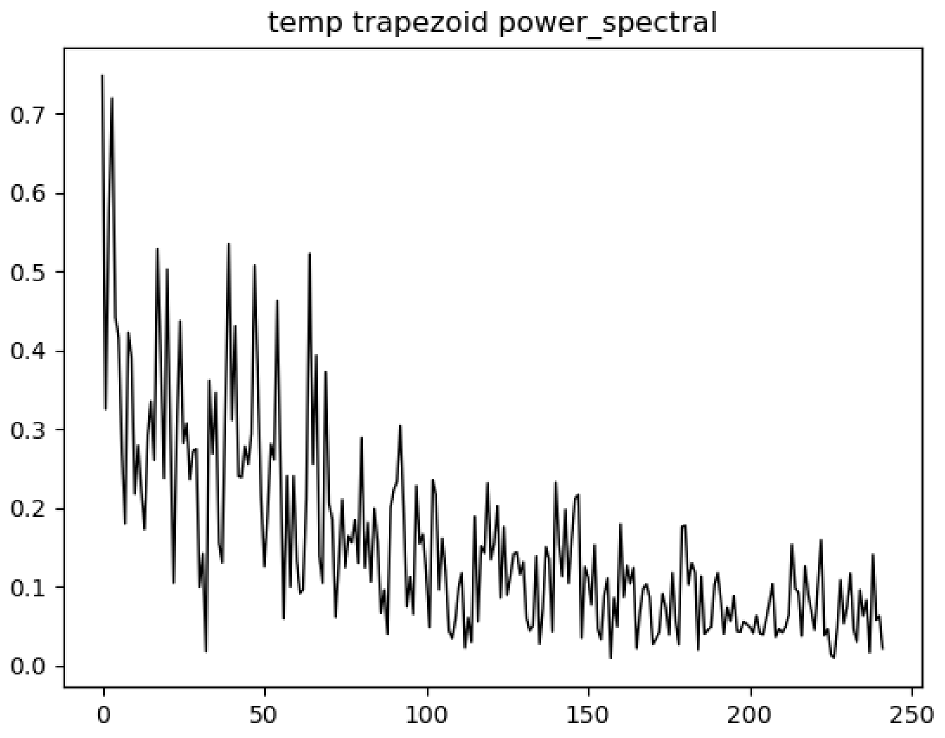

圖 1-4　溫度梯形法

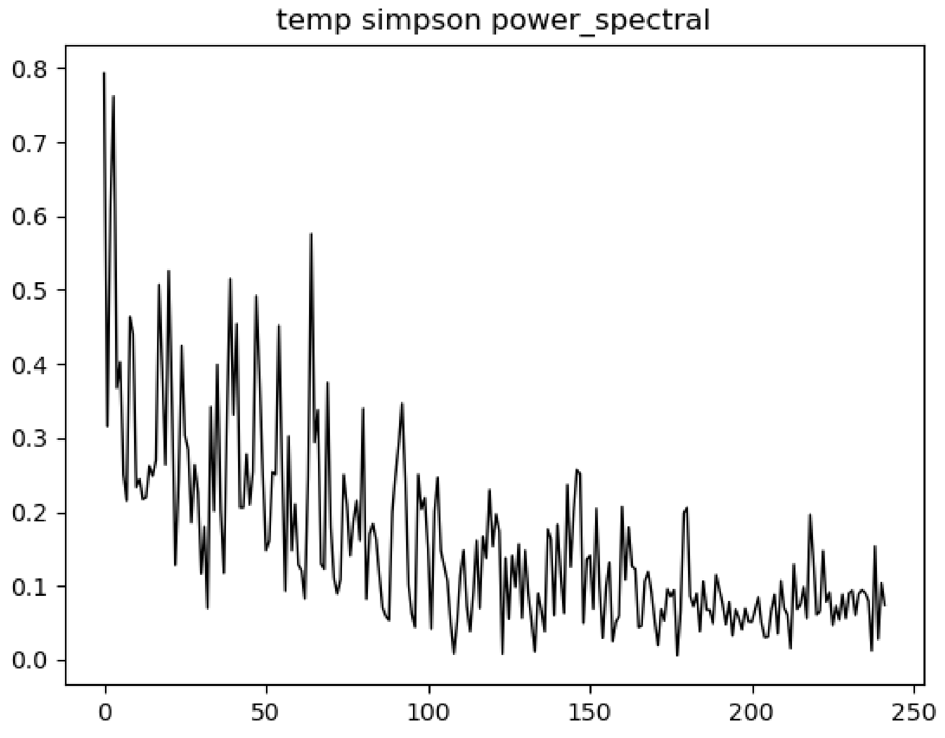

圖 1-5　溫度 Simpson’s rule 法

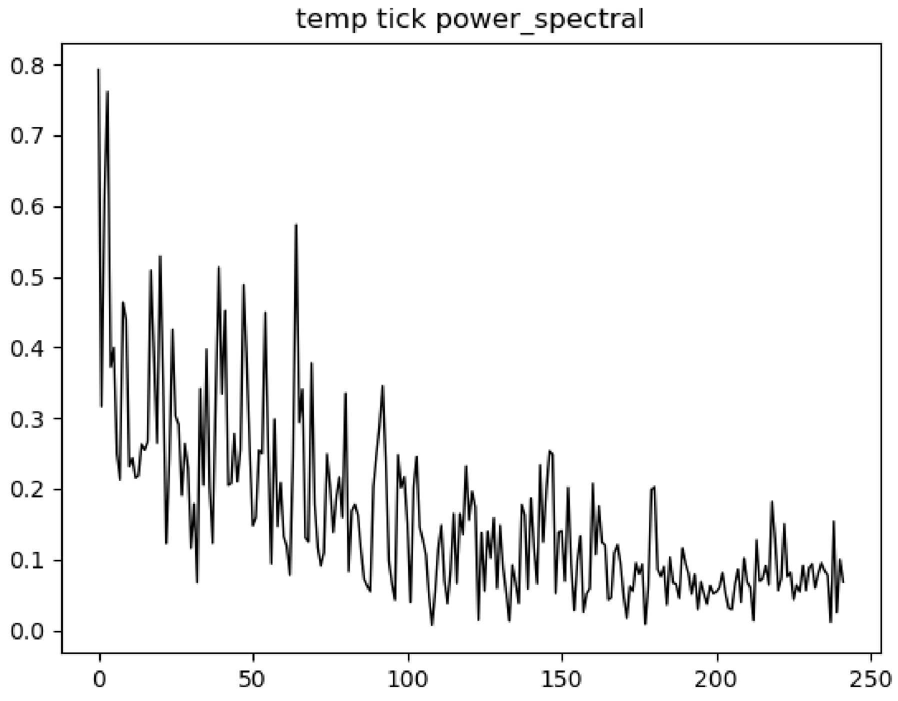

圖 1-6　溫度 The Leo Tick’s formula 法

2.　週期圖

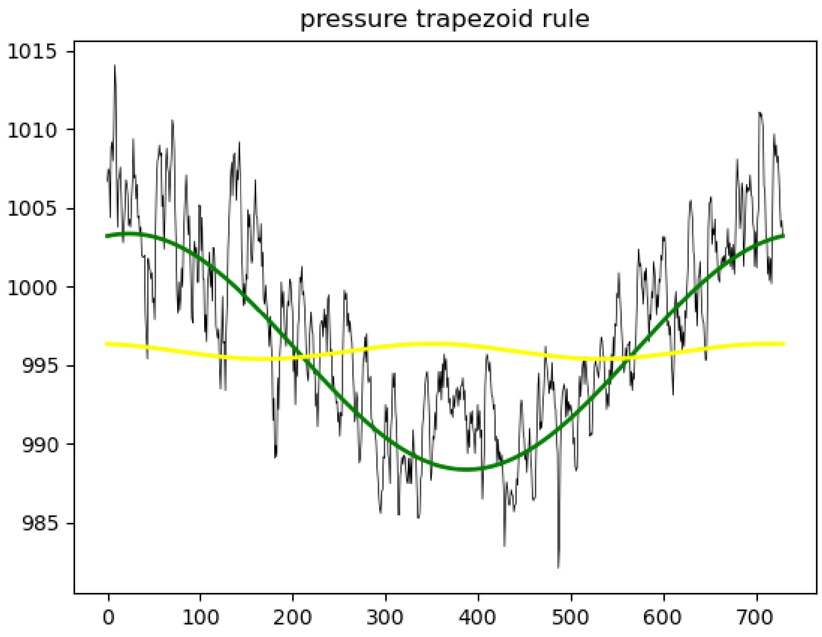

圖 2-1　氣壓梯形法

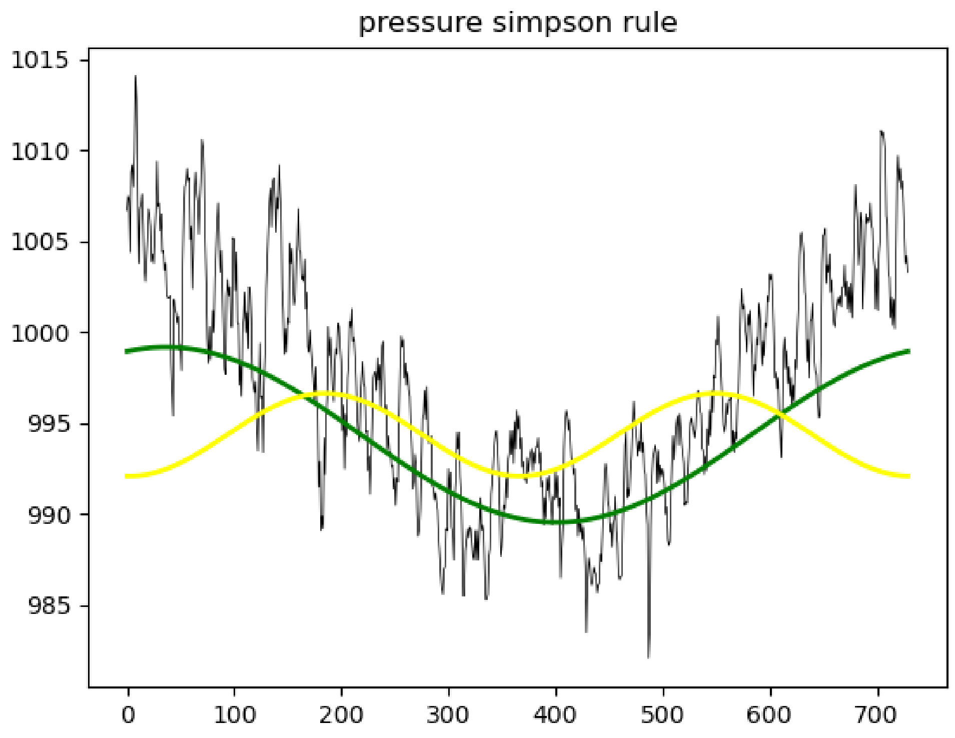

圖 2-2　氣壓 Simpson’s rule 法

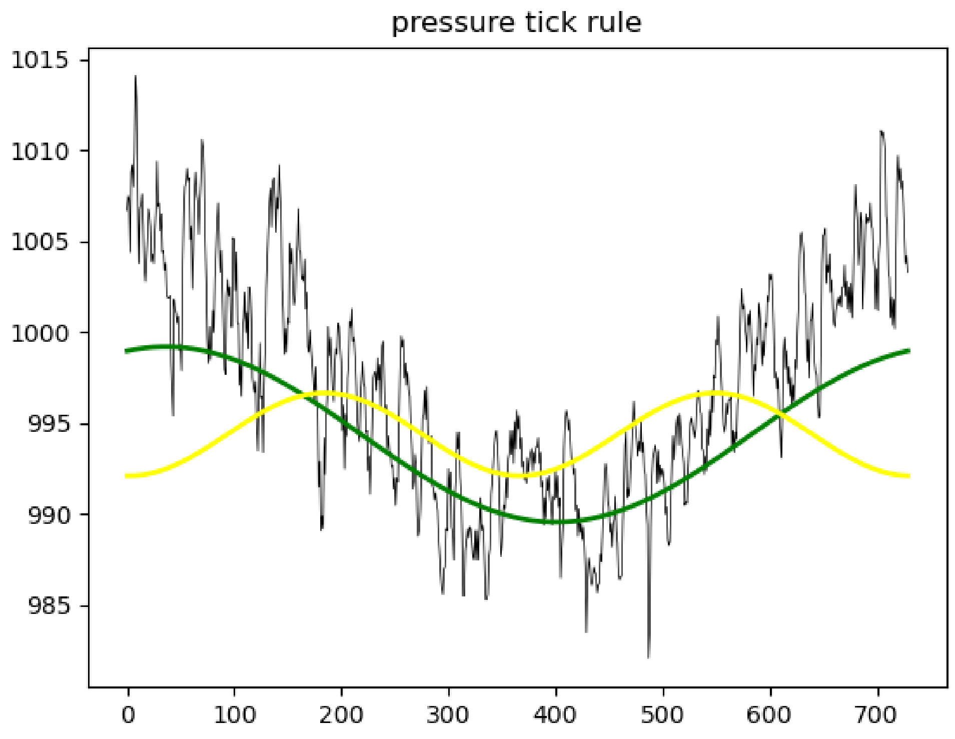

圖 2-3　氣壓 The Leo Tick’s formula 法

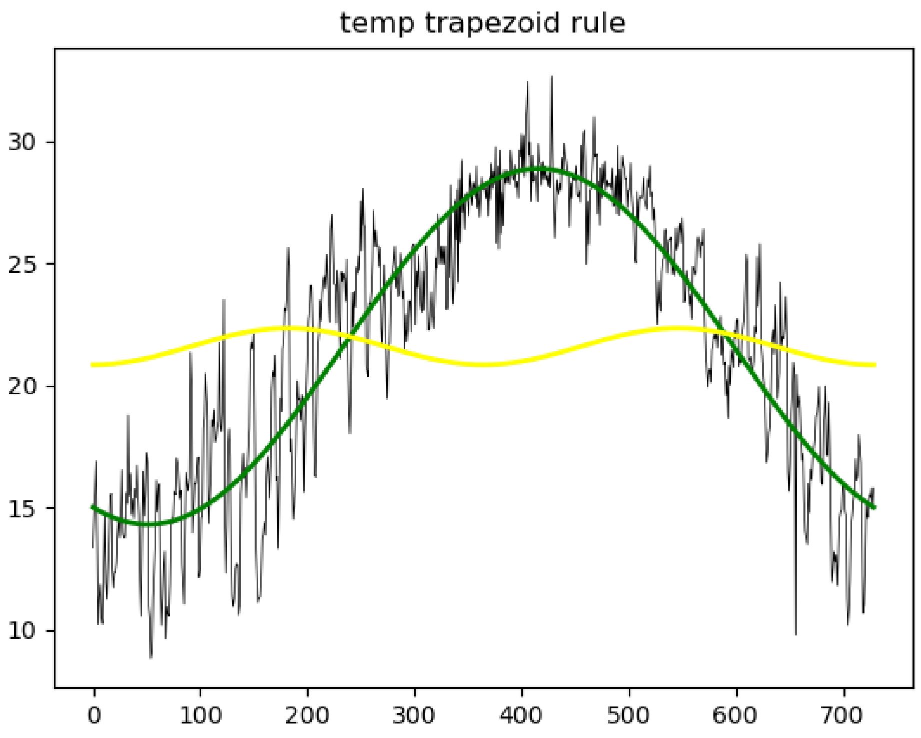

圖 2-4　溫度梯形法

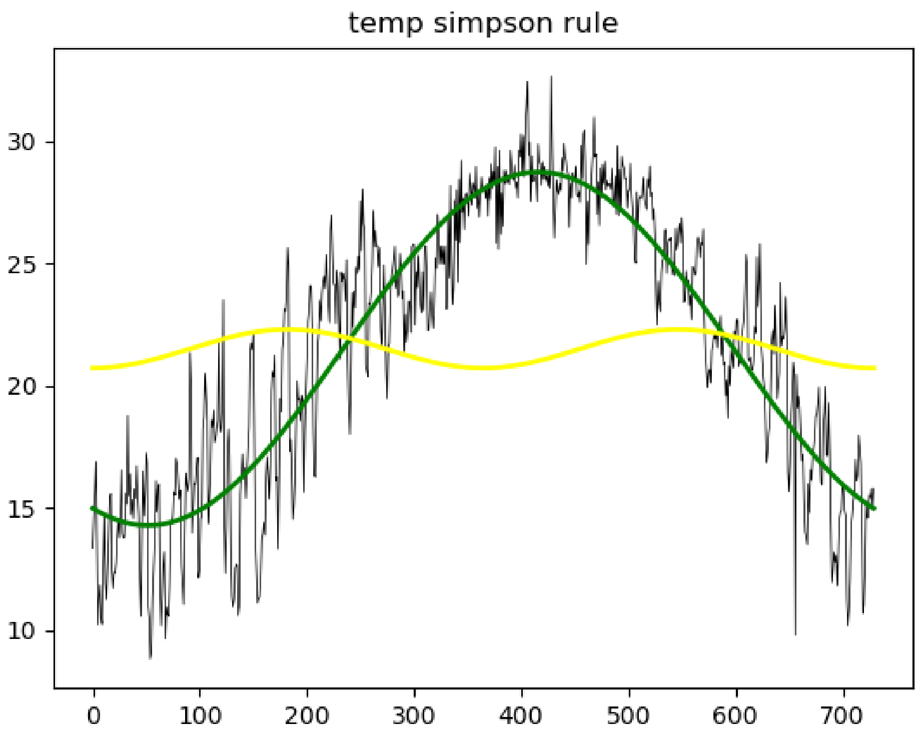

圖 2-5　溫度 Simpson’s rule 法

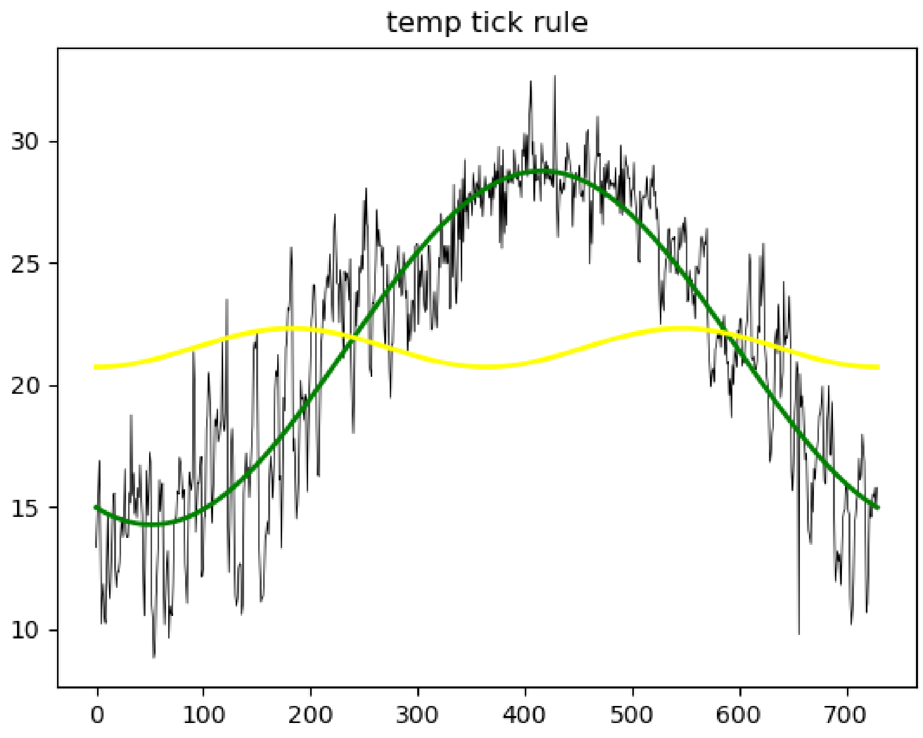

圖 2-6　溫度 The Leo Tick’s formula 法

3. 文字報告(包含第一、第二小題)

先來看第一、第二小題：

第一小題 Power spectral 氣壓的部分，可以看到梯形法隨著 n 的增加，power_spectral 呈現穩定的震盪下降至 0，然而 Simpson’s rule 以及 The Leo Tick’s formula 法皆呈現穩定的震盪下降至 2.6 左右而下不去，在檢查數值積分方法確認沒問題後猜想應該是數值積分本身的問題，誤差計算導致了波譜的不正常反應；但是溫度的部分三種方法都呈現穩定的震盪下降

至 0，應該是氣壓有部分資料的問題不適合後兩者積分方式。

第二小題週期圖氣壓的部分，梯形法擬合出來的一年週期函數相當符合觀測數據，MSE 應該為最小，而半年週期的振幅相對低且位於平均線上，然而 Simpson’s rule 以及 The Leo Tick’s formula 法擬合出來的一年週期函數似乎在氣壓高值有低估的情況，半年週期的振幅也有相對大的震幅；溫度的部分，梯形法擬合出來的一年週期函數同樣的相當符合觀測數據，MSE 應該為最小，而半年週期的振幅低且位於平均線上；Simpson’s rule 以及 The Leo Tick’s formula 法擬合出來的無論一年或半年函數都與梯形法類似。

接下來是 n, power_spectral, amplitude, phase, date 的部分：

pressure trapezoid

n: [0, 27, 3, 6, 20]

power_spectral: [7.5, 1.08, 1.07, 0.98, 0.94]

amplitude: [1991.76, 0.77, 0.36, 0.78, 0.21]

phase: [-0.0, 1.30, -1.40, -0.46, -0.21]

date: [nan, -2.80, 27.09, 4.46, 0.60]

pressure simpson

n: [0, 6, 34, 12, 53]

power_spectral: [4.82, 3.68, 3.51, 3.39, 3.37]

amplitude: [1988.74, 2.09, 2.33, 3.04, 2.73]

phase: [-0.0, 0.20, 0.29, 0.08, 0.25]

date: [nan, -1.90, -0.49, -0.39, -0.28]

pressure tick

n: [0, 6, 34, 12, 53]

power_spectral: [4.82, 3.68, 3.51, 3.39, 3.36]

amplitude: [1988.78, 2.09, 2.33, 3.03, 2.72]

phase: [-0.0, 0.19, 0.29, 0.08, 0.25]

date: [nan, -1.90, -0.49, -0.39, -0.28]

temp trapezoid

n: [0, 1, 4, 3, 40]

power_spectral: [7.28, 0.75, 0.72, 0.57, 0.53]

amplitude: [43.16, 7.28, 0.57, 0.32, 0.34] phase: [-0.0, -0.44, 0.93, 1.38, -0.45]

date: [nan, 25.73, -13.55, -26.77, 0.66]

temp simpson

n: [0, 1, 4, 3, 65]

power_spectral: [7.22, 0.79, 0.76, 0.61, 0.58]

amplitude: [43.00, 7.22, 0.61, 0.32, 0.25]

phase: [-0.0, -0.44, 0.89, 1.48, 1.172]

date: [nan, 25.81, -12.95, -28.66, -1.05]

temp tick

n: [0, 1, 4, 3, 65]

power_spectral: [7.23, 0.79, 0.76, 0.61, 0.57]

amplitude: [43.01, 7.23, 0.61, 0.32, 0.25]

phase: [-0.0, -0.44, 0.89, 1.48, 1.17]

date: [nan, 25.77, -12.92, -28.75, -1.05]

似乎忘記要濾除掉 n=0 的情況，不過大致上還是符合所需要的資料情況。

計算與繪圖程式碼 ＋ 註解

如下附程式碼與註解：

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

# config: sample number and integration's h and power_spectral_endpoint
item_num = 730
gap = (2*np.pi)/item_num
power_spectral_endpoint = 244

# get pressure data and fix error data
def get_p():
    p = np.loadtxt('Ps.dat.txt')
    for i in range(item_num):
        if p[i] < 900.0:
            p[i] = (p[i+1]+p[i-1])/2
    return p
# get temp data and fix error data
def get_t():
    t = np.loadtxt('T.dat.txt')
    for i in range(item_num):
        if t[i] < 7.0:
            t[i] = (t[i+1]+t[i-1])/2
    return t

# trapezoid integral: delta A = (ui + ui+1)*gap/2
def trapezoid(inside_integral):
    integral_area = 0
    for i in range(item_num-1):
        integral_area += (inside_integral[i] + inside_integral[i+1])*ga
p/2
    #print('trapezoid area:', integral_area)
    return integral_area
#  simpson integral: delta A = (1/3)*gap*(ui + 4*ui+1 + ui+2)  (but i=1
, 3, 5...)
```

```python
def simpson(inside_integral):
    integral_area = 0
    for i in range(0, item_num-2, 2):
        integral_area += (1/3)*gap*(inside_integral[i] + 4*inside_integ
ral[i+1] + inside_integral[i+2])
    #print('simpson area:', integral_area)
    return integral_area
# tick integral: delta A = (1/3)*gap*(1.0752*ui + 3.8496*ui+1 + 1.0752*
ui+2)  (but i=1, 3, 5...)
def tick(inside_integral):
    integral_area = 0
    for i in range(0, item_num-2, 2):
        integral_area += (1/3)*gap*(1.0752*inside_integral[i] + 3.8496*
inside_integral[i+1] + 1.0752*inside_integral[i+2])
    #print('tick area:', integral_area)
    return integral_area

# a routing for numerical integration
def numerical_integration_routing(inside_integral, way):
    if way == 'trapezoid':
        return trapezoid(inside_integral)
    elif way == 'simpson':
        return simpson(inside_integral)
    elif way == 'tick':
        return tick(inside_integral)
    else:print('numerical_integration way error!!!')

# calculate a0, an and bn
def calculate_a0(p_or_t, way):
    return numerical_integration_routing(p_or_t, way) / (2*np.pi)
def calculate_an(p_or_t, n, way):
    x = np.linspace(0, 2*np.pi, item_num)
    inside_integral = p_or_t*np.cos(n*x)
    return numerical_integration_routing(inside_integral, way) / np.pi
def calculate_bn(p_or_t, n, way):
    x = np.linspace(0, 2*np.pi, item_num)
    inside_integral = p_or_t*np.sin(n*x)
    return numerical_integration_routing(inside_integral, way) / np.pi
```

```python
# calculate amplitude and phase and date
def calculate_amplitude(a, b):
    return (a**2 + b**2)**0.5
def calculate_phase(a, b):
    return np.arctan(-1*b/a)
def calculate_date(phase, n):
    x = -1*(phase/n)
    return x*365/(2*np.pi)

# plot f(x)(n=1=one year, 2=half year) and source data
def plot_f_of_x(p_or_t, way, text):
    print('plot_f_of_x', text, way)
    x = np.linspace(0, 2*np.pi, item_num)

    # calculate f_of_x_1 and f_of_x_2
    a1 = calculate_an(p_or_t=p_or_t, n=1, way=way)
    b1 = calculate_bn(p_or_t=p_or_t, n=1, way=way)
    f_of_x_1 = calculate_a0(p_or_t, way=way) + a1*np.cos(1*x) + b1*np.s
in(1*x)
    a2 = calculate_an(p_or_t=p_or_t, n=2, way=way)
    b2 = calculate_bn(p_or_t=p_or_t, n=2, way=way)
    f_of_x_2 = calculate_a0(p_or_t, way=way) + a2*np.cos(2*x) + b2*np.s
in(2*x)

    # plot and save
    plt.plot(p_or_t, color='black', linewidth=0.5)
    plt.plot(f_of_x_1, color='green', linewidth=2)
    plt.plot(f_of_x_2, color='yellow', linewidth=2)
    plt.title('{} {} rule'.format(text, way))

    if not os.path.exists('f_of_x'):
        os.mkdir('f_of_x')
    plt.savefig(os.path.join('f_of_x', 'f_of_x_{}_{}.png'.format(text,
way)))
    plt.close()

# calculate and plot power_spectral([1:])
```

```python
def plot_power_spectral(p_or_t, way, text):
    print('plot_power_spectral', text, way)
    # create and calculate power_spectral
    power_spectral = []
    for n in range(1, power_spectral_endpoint):
        a = calculate_an(p_or_t=p_or_t, n=n, way=way)
        b = calculate_bn(p_or_t=p_or_t, n=n, way=way)
        power_spectral.append(calculate_amplitude(a, b))
    power_spectral = power_spectral[1:]

    # plot and save
    plt.plot(power_spectral, color='black', linewidth=1)
    plt.title('{} {} power_spectral'.format(text, way))

    if not os.path.exists('power_spectral'):
        os.mkdir('power_spectral')
    plt.savefig(os.path.join('power_spectral', 'power_spectral_{}_{}.pn
g'.format(text, way)))
    plt.close()

# return n, power_spectral, amplitude, phase, date for range search_dee
p=5
def get_max_power_spectral_info(p_or_t, way, text):
    print('get_max_power_spectral_info', text, way)
    search_deep = 5
    complete_power_spectral = []
    n, power_spectral, amplitude, phase, date = [], [], [], [], []

    # create and calculate complete_power_spectral
    for nn in range(1, power_spectral_endpoint):
        a = calculate_an(p_or_t=p_or_t, n=nn, way=way)
        b = calculate_bn(p_or_t=p_or_t, n=nn, way=way)
        complete_power_spectral.append(calculate_amplitude(a, b))
    n = sorted(range(len(complete_power_spectral)), key = lambda k : co
mplete_power_spectral[k], reverse = True)[:search_deep]
    print('n: ', n)

    # calculate and append every list
    count = 0
    for i in n:
        power_spectral.append(complete_power_spectral[i])
        a = calculate_an(p_or_t=p_or_t, n=i, way=way)
        b = calculate_bn(p_or_t=p_or_t, n=i, way=way)
        amplitude.append(calculate_amplitude(a, b))
        phase.append(calculate_phase(a, b))
        date.append(calculate_date(phase[count], i))
        count += 1
    print('power_spectral: ', power_spectral)
    print('amplitude: ', amplitude)
    print('phase: ', phase)
    print('date: ', date)
    print()


if __name__ == "__main__":
    #integration_list = ['trapezoid', 'simpson', 'tick']
    print('main init')

    t = get_t()
    p = get_p()

    # plot_f_of_x
    plot_f_of_x(p_or_t=p, way='trapezoid', text='pressure')
    plot_f_of_x(p_or_t=t, way='trapezoid', text='temp')
    plot_f_of_x(p_or_t=p, way='simpson', text='pressure')
    plot_f_of_x(p_or_t=t, way='simpson', text='temp')
    plot_f_of_x(p_or_t=p, way='tick', text='pressure')
    plot_f_of_x(p_or_t=t, way='tick', text='temp')

    # plot_power_spectral
    plot_power_spectral(p_or_t=p, way='trapezoid', text='pressure')
    plot_power_spectral(p_or_t=t, way='trapezoid', text='temp')
    plot_power_spectral(p_or_t=p, way='simpson', text='pressure')
    plot_power_spectral(p_or_t=t, way='simpson', text='temp')
    plot_power_spectral(p_or_t=p, way='tick', text='pressure')
    plot_power_spectral(p_or_t=t, way='tick', text='temp')
    # get_max_power_spectral_info(n, power_spectral, amplitude, phase, date)
    get_max_power_spectral_info(p_or_t=p, way='trapezoid', text='pressure')
    get_max_power_spectral_info(p_or_t=t, way='trapezoid', text='temp')
    get_max_power_spectral_info(p_or_t=p, way='simpson', text='pressure')
    get_max_power_spectral_info(p_or_t=t, way='simpson', text='temp')
    get_max_power_spectral_info(p_or_t=p, way='tick', text='pressure')
    get_max_power_spectral_info(p_or_t=t, way='tick', text='temp')
```