天氣學下 hw5　　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. 圖片部分，由於原始參數 t=0.01 疊代 5000 次 leap_frog 會爆掉，因此將參數設為依照 t=0.005 疊代 10000 次，並且圖片順序為 forward->leap_frog->backward、x-y -> x-z -> z-y -> x-t -> y-t -> z-t 的順序排列)

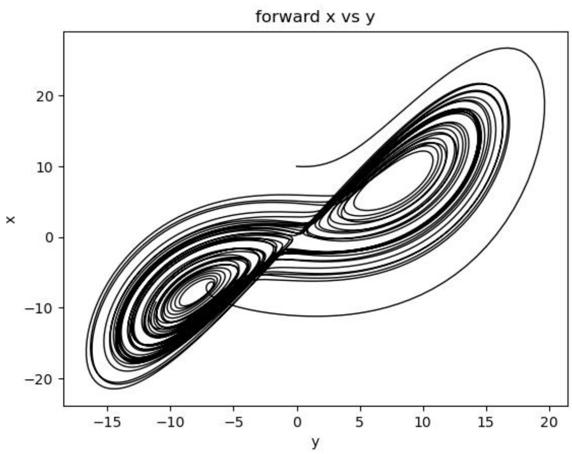


圖 1 forward x-y

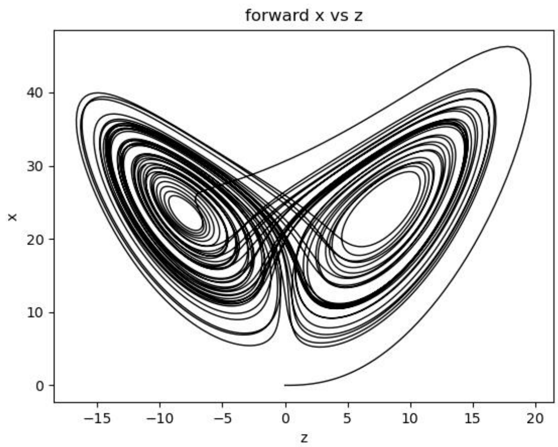


圖 2 forward x-z

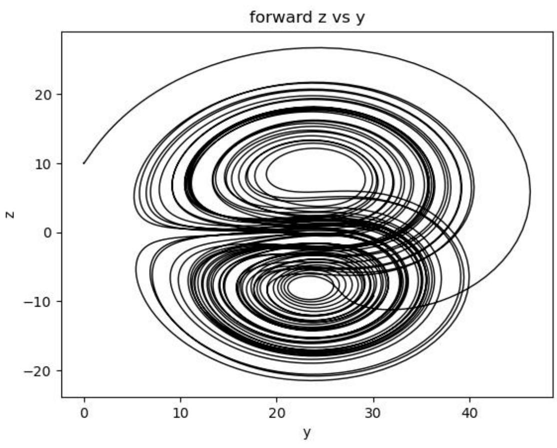


圖 3 forward z-y

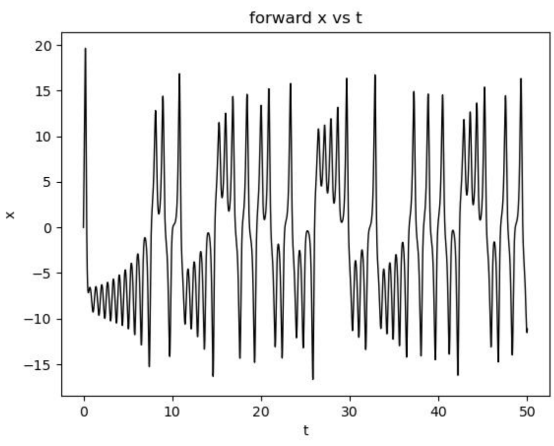
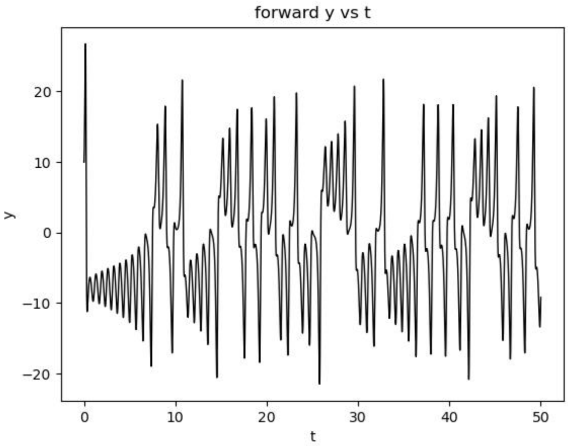


圖 4 forward x-t



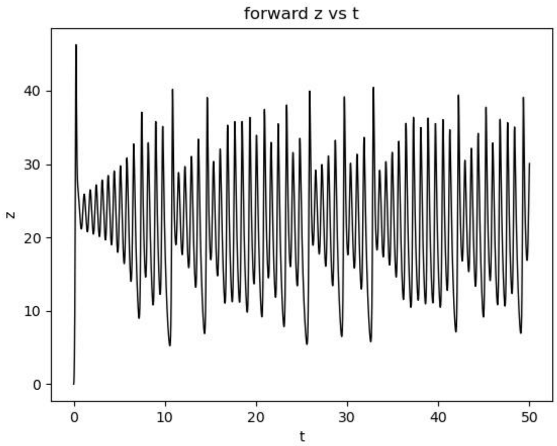
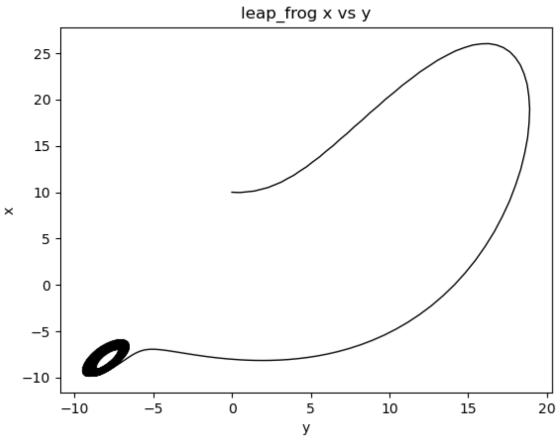


圖 5 forward y-t

圖 6 forward z-t



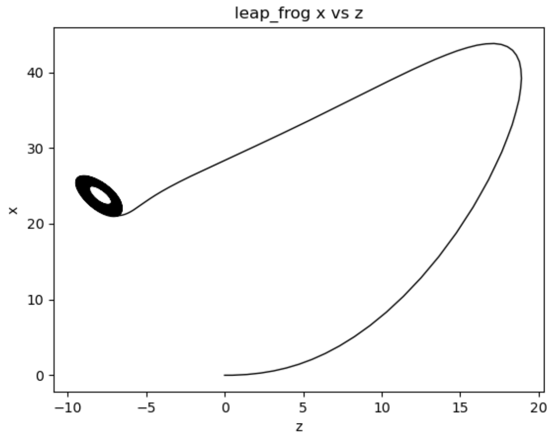
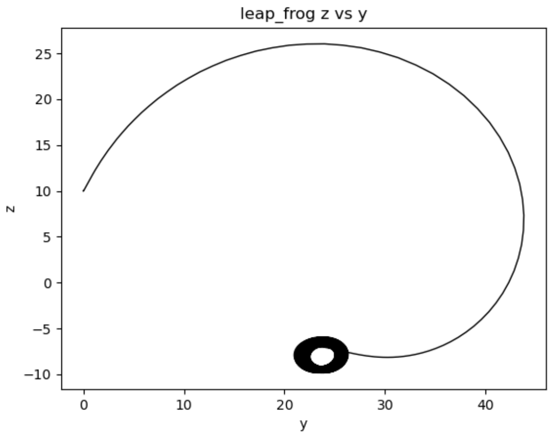


圖 7 leap_frog x-y

圖 8 leap_frog x-z



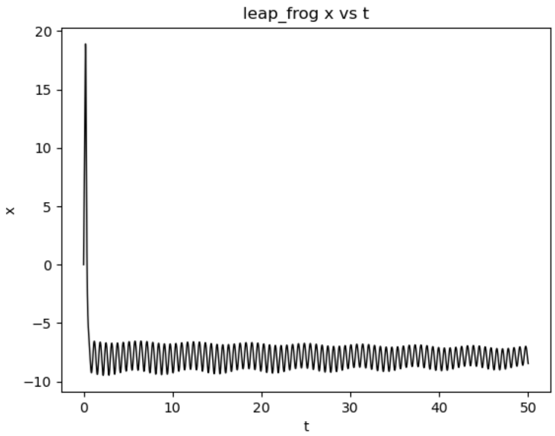


圖 9 leap_frog z-y

圖 10 leap_frog x-t

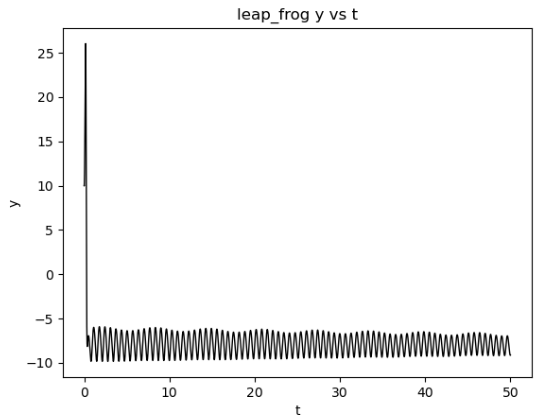


圖 11 leap_frog y-t

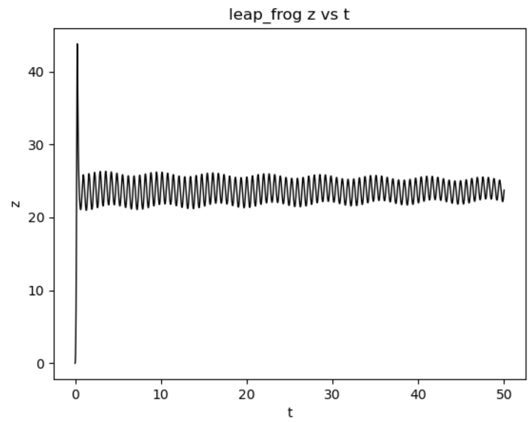


圖 12 leap_frog z-t

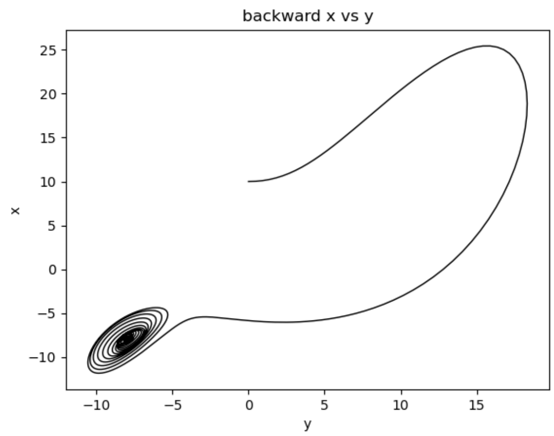


圖 13 backward x-y

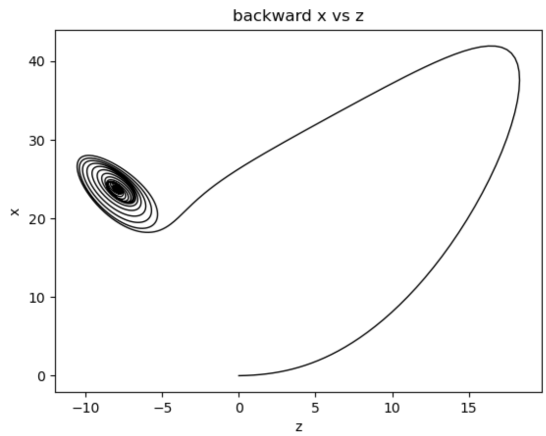


圖 14 backward x-z

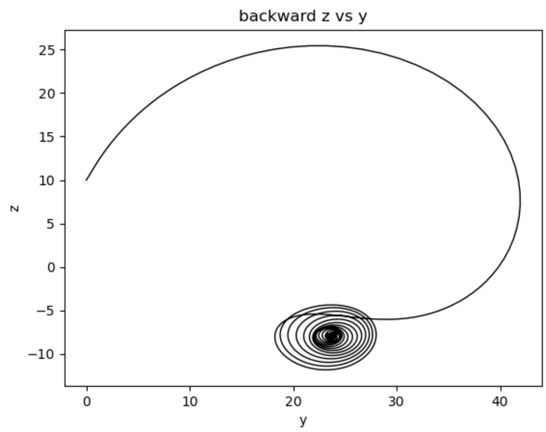


圖 15 backward z-y

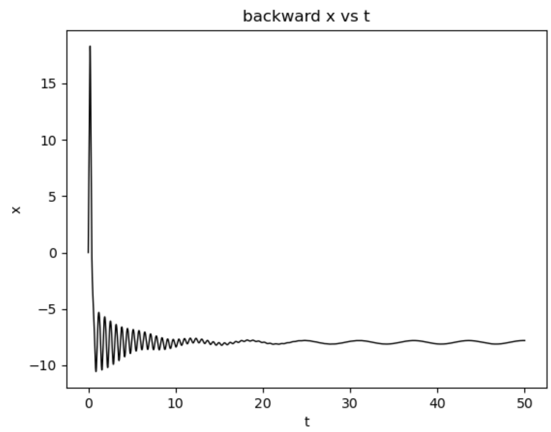


圖 16 backward x-t

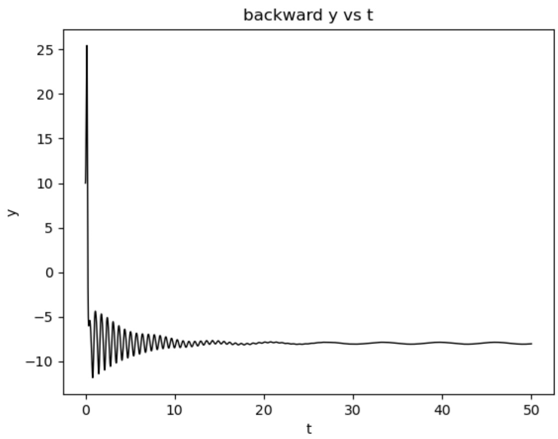


圖 17 backward y-t

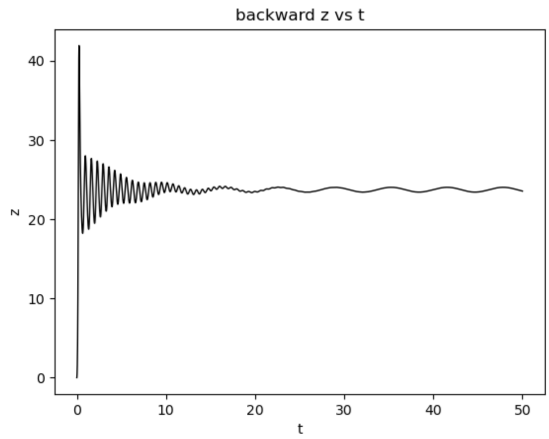


圖 18 backward z-t

2. 討論 2 小題：藉由 Lorenz 方程討論數值積分非線性方程時出現的問題，對數值天氣預報會有甚麼問題?

這個作業採用了三個非線性微分方程的系統如下：

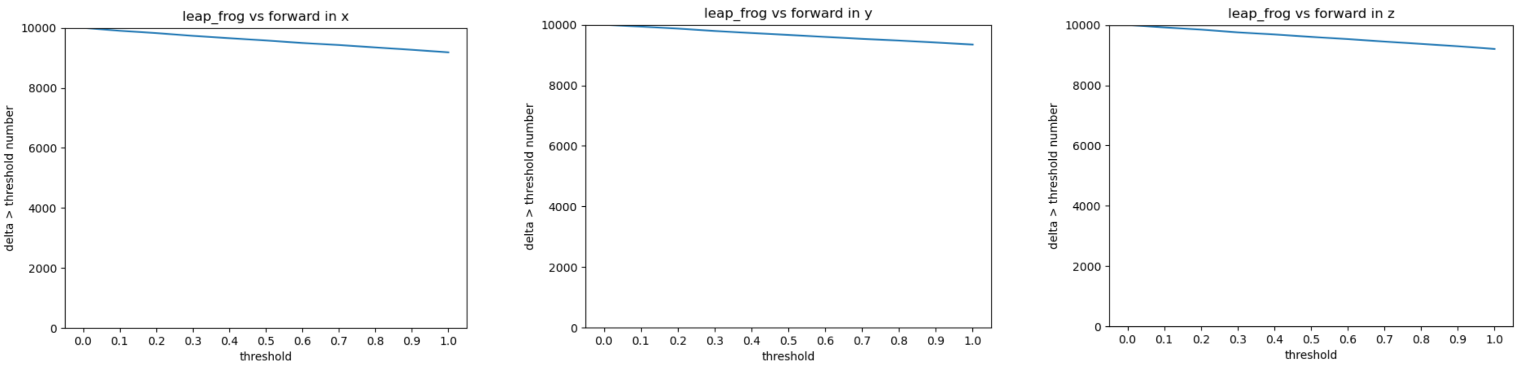
$$\frac{dX}{dt} = -sX + sY + f\cos\theta$$
$$\frac{dY}{dt} = -XZ + rX - Y + f\sin\theta$$
$$\frac{dZ}{dt} = XY - bZ$$

每段的 d(x/y/z) / dt 都可以用給定的(x/y/z)以及其他固定數計算得到，因此可以配合不同的數值積分方式一路推展(x/y/z)以及 d(x/y/z) / dt。然而有趣的地方在於 x/y/z 之間有著一定的關係，繪圖也可發現規律，而且不同的積分方法會有很不同的規律樣式，例如 forward 的 x-y 斜 45 度無限型漸進線圖樣、x-z 的雙

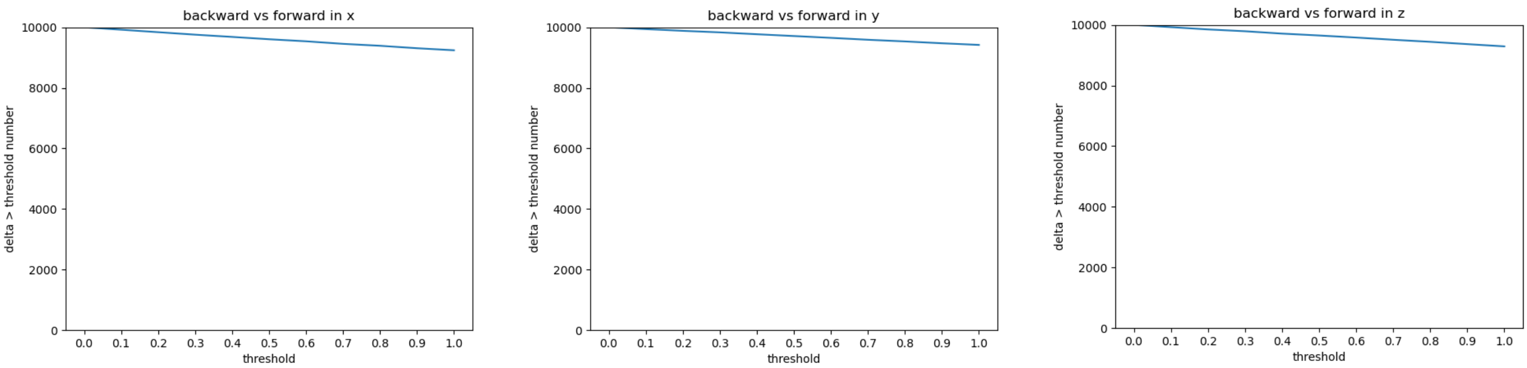
翼蝴蝶狀圖樣、z-y 的雙核環繞圖樣，或是 backward 以及 leap_frog 的 x-y、x-z、z-y 的螺旋漸進圖樣(兩者在收斂的速度不同)，然而其值卻非常地不規律，forward (x/y/z)對 t 的作圖可以發現在大週期波上有很不規律的震盪，但是在小週期上是穩定呈現震幅逐漸加大的震盪，直至某個不固定的閾值在收斂到另外一個不固定的值，類似於數學領域上的碎形理論；而 leap_frog 則是呈現收斂後短周期震盪的情況；backward 則是呈現較長周期的震盪。

因此在處理類似或更複雜的問題，例如大氣非線性方程時也可能出現類似的情形，也就是非線性帶來的初始值不穩定，會導致後續大小週期震盪的不穩定，並進而引響後續所有參數的值可能有相當大的差距，造成長期預報的不可信，而且不同的預報方法可能都有其特性，對於初始給定值的敏感度與輸出呈現也都不同，甚至於還需要增加時間解析度以防止運算超過一假定的閾值造成如 t=0.01 時 leap_frog 的數值爆炸。

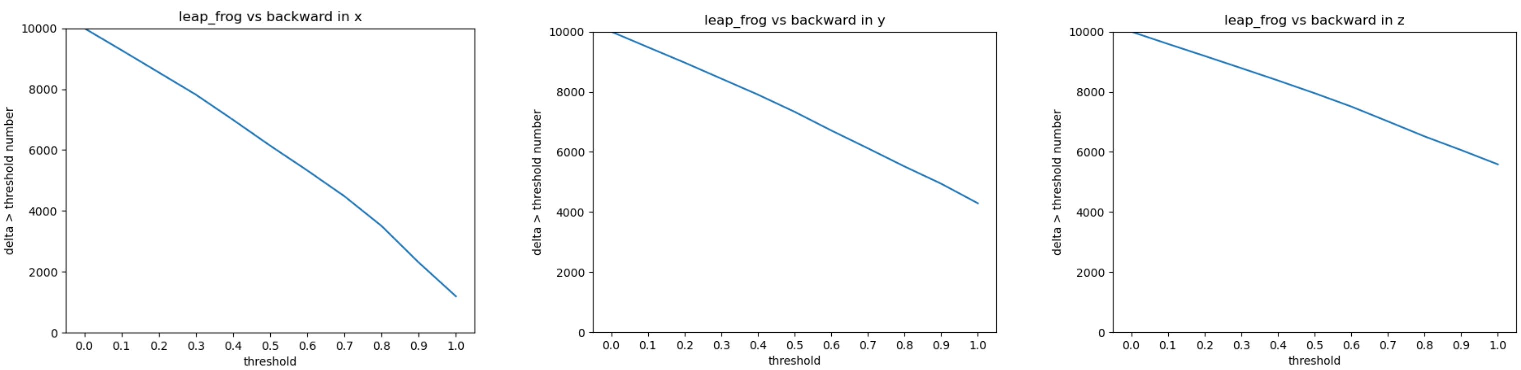
另外值得一提的是各數值積分方法的比較，這部分是學生多做的，目的是想討論三種 scheme 的 x/y/z 值差距：比較三種方法得出的 10000 個值的差異，y 軸為大於 x 軸閾值的數量：



Forward vs leap_frog



Forward vs backward



leap_frog vs backward

可以看到三種數值方式的差距都相當大，尤其是 Forward 與另外兩種積分方式

有九成以上的數值與另外兩種積分方式得出的數值差異>1；相較之下

leap_frog vs backward 差異較小，尤其在 x 的部分如此。

在跟同學以及助教討論過之後，發覺即使演算邏輯完全相同，不同的 os 還記也有可能產生所謂的捨入誤差或是其他非人為誤差，導致作非線性方程式的求解時會有加成的現象，使結果也相當不同，可謂是蝴蝶效應的最佳實證😊，我頭好痛。

最後想說的是這份作業特別感謝助教跟同學們的幫忙跟討論，尤其是洪琳助教跟龍孟偉同學，不然我光是思考系統影響應該就會直接掛掉。

3. 計算與繪圖程式碼如下，註解應該寫得還行：

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

# basic initialization
dt = 0.01/2
times = 5000*2

s = 10
r = 24.74
b = 2.6666667
f = 2.5

# return d(x/y/z)/dt
def get_dxdt(x, y, sita):
    return -1*s*x + s*y + f*np.cos(sita)
```

```python
def get_dydt(x, y, z, sita):
    return -1*x*z + r*x - y + f*np.sin(sita)
def get_dzdt(x, y, z):
    return x*y - b*z

# return x_n+1
def forward_scheme(x_n, f_n, dt=0.01):
    return (x_n + f_n*dt)
def leap_frog_scheme(x_n_sub1, f_n, dt=0.01):
    return (x_n_sub1 + f_n*2*dt)
def backward_scheme(x_n, f_n_add1, dt=0.01):
    return (x_n + f_n_add1*dt)

# leap_frog_time_filter uses x_n+1 and xn and y_n-1 to derive y_n, then y_n replace x_n
def leap_frog_time_filter(x_n, x_n_add1, y_n_sub1):
    alpha = 0.06
    return x_n + alpha*(y_n_sub1 - 2*x_n + x_n_add1)

# plot
def plot_xyz_t(xyz, t, title):
    plt.plot(t, xyz, color='black', linewidth=1)
    plt.xlabel(title.split(' ')[3])
    plt.ylabel(title.split(' ')[1])
    plt.title(title)
    plt.savefig(os.path.join(os.getcwd(), 'output_image2', title))
    #plt.show()
    plt.close()
def plot_x_y_z(xyz1, xyz2, title):
    plt.plot(xyz1, xyz2, color='black', linewidth=1)
    plt.xlabel(title.split(' ')[3])
    plt.ylabel(title.split(' ')[1])
    plt.title(title)
    plt.savefig(os.path.join(os.getcwd(), 'output_image2', title))
    #plt.show()
    plt.close()

# valid scheme function
def plot_scheme_delta(array1, array2, text):
```

```
        over_num_list = []
        threshold_list = np.linspace(1.0, 0.0, 10+1)
        for threshold in threshold_list:
            num = 0
            for delta in array1-array2:
                if delta>threshold:num+=1
                if delta<-1*threshold:num+=1
            over_num_list.append(num)
        plt.xticks(threshold_list)
        plt.ylim(top=times)
        plt.xlabel('threshold')
        plt.ylabel('delta > threshold number')
        plt.plot(threshold_list, over_num_list)
        plt.title(text)
        plt.savefig(os.path.join(os.getcwd(), 'output_image2', text))
        #plt.show()
        plt.close()


    if __name__ == '__main__':
        # check output_image2 exists
        if not os.path.exists(os.path.join(os.getcwd(), 'output_image2')):
            os.mkdir(os.path.join(os.getcwd(), 'output_image2'))

        # parameter initialization
        x_array, y_array, z_array, sita_array, t_array = np.full((times+1), -
1.0), np.full((times+1), -1.0), np.full((times+1), -1.0), np.full((times+1), -
1.0), np.full((times+1), -1.0)
        x_array[0], y_array[0], z_array[0], sita_array[0], t_array[0] = 0, 10, 0, 45*2*np.pi/360, 0
        for i in range(1, times+1):
            sita_array[i] = sita_array[i-1] + dt
            t_array[i] = t_array[i-1] + dt

        # copy x/y/z array for next three schemes, used to record and valid
        x_array_forward, x_array_leap_frog, x_array_backward = np.copy(x_array), np.copy(x_array), n
p.copy(x_array)
        y_array_forward, y_array_leap_frog, y_array_backward = np.copy(y_array), np.copy(y_array), n
p.copy(y_array)
```

```
        z_array_forward, z_array_leap_frog, z_array_backward = np.copy(z_array), np.copy(z_array), n
p.copy(z_array)

        # forward part(know x_n -> know f'(x) -> derive x_n+1, EZ)
        scheme = 'forward'
        for i in range(times):
            if i % 1000 == 0:print('{} epoch i='.format(scheme), i)
            # calculate f_n
            dxdt = get_dxdt(x_array_forward[i], y_array_forward[i], sita_array[i])
            dydt = get_dydt(x_array_forward[i], y_array_forward[i], z_array_forward[i], sita_array[i
])
            dzdt = get_dzdt(x_array_forward[i], y_array_forward[i], z_array_forward[i])
            # calculate x_n+1 with x_n and f_n
            x_array_forward[i+1] = forward_scheme(x_array_forward[i], dxdt, dt=dt)
            y_array_forward[i+1] = forward_scheme(y_array_forward[i], dydt, dt=dt)
            z_array_forward[i+1] = forward_scheme(z_array_forward[i], dzdt, dt=dt)
        plot_xyz_t(x_array_forward, t_array, title='{} x vs t'.format(scheme))
        plot_xyz_t(y_array_forward, t_array, title='{} y vs t'.format(scheme))
        plot_xyz_t(z_array_forward, t_array, title='{} z vs t'.format(scheme))
        plot_x_y_z(x_array_forward, y_array_forward, title='{} x vs y'.format(scheme))
        plot_x_y_z(x_array_forward, z_array_forward, title='{} x vs z'.format(scheme))
        plot_x_y_z(z_array_forward, y_array_forward, title='{} z vs y'.format(scheme))
        print()

        # leap_frog part
        # if i=0: forward to know x_1
        # else:  1.using f'(x) and x_n-1 to derive x_n+1  2.using x_n+1 and xn and y_n-
1 to derive y_n, then y_n replace x_n
        scheme = 'leap_frog'
        for i in range(times):
            if i % 1000 == 0:
                print('{} epoch i='.format(scheme), i)
            # forward to know x_1
            if i == 0:
                # calculate f_0
                dxdt = get_dxdt(x_array_leap_frog[i], y_array_leap_frog[i], sita_array[i])
                dydt = get_dydt(x_array_leap_frog[i], y_array_leap_frog[i], z_array_leap_frog[i], si
ta_array[i])
```

```
            dzdt = get_dzdt(x_array_leap_frog[i], y_array_leap_frog[i], z_array_leap_frog[i])
            # forward calculate x_1 with x0 and f0
            x_array_leap_frog[i+1] = forward_scheme(x_array_leap_frog[i], dxdt, dt=dt)
            y_array_leap_frog[i+1] = forward_scheme(y_array_leap_frog[i], dydt, dt=dt)
            z_array_leap_frog[i+1] = forward_scheme(z_array_leap_frog[i], dzdt, dt=dt)
            # calculate y_0
            yx, yy, yz = x_array_leap_frog[i], y_array_leap_frog[i], z_array_leap_frog[i]
            continue
        # 1.using f'(x) and x_n-1 to derive x_n+1
        # calculate f_n
        dxdt = get_dxdt(x_array_leap_frog[i], y_array_leap_frog[i], sita_array[i])
        dydt = get_dydt(x_array_leap_frog[i], y_array_leap_frog[i], z_array_leap_frog[i], sita_a
rray[i])
        dzdt = get_dzdt(x_array_leap_frog[i], y_array_leap_frog[i], z_array_leap_frog[i])
        # leap_frog calculate x_n+1 with x_n-1 and f_n
        x_array_leap_frog[i+1] = leap_frog_scheme(x_array_leap_frog[i-1], dxdt, dt=dt)
        y_array_leap_frog[i+1] = leap_frog_scheme(y_array_leap_frog[i-1], dydt, dt=dt)
        z_array_leap_frog[i+1] = leap_frog_scheme(z_array_leap_frog[i-1], dzdt, dt=dt)

        # 2.using x_n+1 and xn and y_n-1 to derive y_n, then y_n replace x_n
        yx = leap_frog_time_filter(x_array_leap_frog[i], x_array_leap_frog[i+1], yx)
        x_array_leap_frog[i] = yx
        yy = leap_frog_time_filter(y_array_leap_frog[i], y_array_leap_frog[i+1], yy)
        y_array_leap_frog[i] = yy
        yz = leap_frog_time_filter(z_array_leap_frog[i], z_array_leap_frog[i+1], yz)
        z_array_leap_frog[i] = yz
        #if i % 100 == 0:
        #    print(i, 'yx', yx, yy, yz)

    plot_xyz_t(x_array_leap_frog, t_array, title='{} x vs t'.format(scheme))
    plot_xyz_t(y_array_leap_frog, t_array, title='{} y vs t'.format(scheme))
    plot_xyz_t(z_array_leap_frog, t_array, title='{} z vs t'.format(scheme))
    plot_x_y_z(x_array_leap_frog, y_array_leap_frog, title='{} x vs y'.format(scheme))
    plot_x_y_z(x_array_leap_frog, z_array_leap_frog, title='{} x vs z'.format(scheme))
    plot_x_y_z(z_array_leap_frog, y_array_leap_frog, title='{} z vs y'.format(scheme))
    print()
```

```
        # backward part(1.forward to know x_n+1(fake) and f'(n+1) 2.backward_scheme uses x_n and f'(
n+1) to derive x_n+1)
        scheme = 'backward'
        for i in range(times):
            if i % 1000 == 0:print('{} epoch i='.format(scheme), i)
            # 1.forward to know x_n+1(fake) and f'(n+1)
            # calculate f_n
            dxdt = get_dxdt(x_array_backward[i], y_array_backward[i], sita_array[i])
            dydt = get_dydt(x_array_backward[i], y_array_backward[i], z_array_backward[i], sita_arra
y[i])
            dzdt = get_dzdt(x_array_backward[i], y_array_backward[i], z_array_backward[i])
            # calculate x_n+1(fake) with x_n and f_n
            x_array_backward[i+1] = forward_scheme(x_array_backward[i], dxdt, dt=dt)
            y_array_backward[i+1] = forward_scheme(y_array_backward[i], dydt, dt=dt)
            z_array_backward[i+1] = forward_scheme(z_array_backward[i], dzdt, dt=dt)
            # calculate f_n+1
            dxdt = get_dxdt(x_array_backward[i+1], y_array_backward[i+1], sita_array[i+1])
            dydt = get_dydt(x_array_backward[i+1], y_array_backward[i+1], z_array_backward[i+1], sit
a_array[i+1])
            dzdt = get_dzdt(x_array_backward[i+1], y_array_backward[i+1], z_array_backward[i+1])
            # 2.backward_scheme uses x_n and f'(n+1) to derive x_n+1)
            x_array_backward[i+1] = backward_scheme(x_array_backward[i], dxdt, dt=dt)
            y_array_backward[i+1] = backward_scheme(y_array_backward[i], dydt, dt=dt)
            z_array_backward[i+1] = backward_scheme(z_array_backward[i], dzdt, dt=dt)
        plot_xyz_t(x_array_backward, t_array, title='{} x vs t'.format(scheme))
        plot_xyz_t(y_array_backward, t_array, title='{} y vs t'.format(scheme))
        plot_xyz_t(z_array_backward, t_array, title='{} z vs t'.format(scheme))
        plot_x_y_z(x_array_backward, y_array_backward, title='{} x vs y'.format(scheme))
        plot_x_y_z(x_array_backward, z_array_backward, title='{} x vs z'.format(scheme))
        plot_x_y_z(z_array_backward, y_array_backward, title='{} z vs y'.format(scheme))
        print()


        # scheme_delta check
        plot_scheme_delta(x_array_leap_frog, x_array_forward, text='leap_frog vs forward in x')
        plot_scheme_delta(y_array_leap_frog, y_array_forward, text='leap_frog vs forward in y')
        plot_scheme_delta(z_array_leap_frog, z_array_forward, text='leap_frog vs forward in z')

        plot_scheme_delta(x_array_backward, x_array_forward, text='backward vs forward in x')
```

```
    plot_scheme_delta(y_array_backward, y_array_forward, text='backward vs forward in y')
    plot_scheme_delta(z_array_backward, z_array_forward, text='backward vs forward in z')

    plot_scheme_delta(x_array_leap_frog, x_array_backward, text='leap_frog vs backward in x')
    plot_scheme_delta(y_array_leap_frog, y_array_backward, text='leap_frog vs backward in y')
    plot_scheme_delta(z_array_leap_frog, z_array_backward, text='leap_frog vs backward in z')
```