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(依照 forward->leap_frog->backward、

x-y -> x-z -> z-y -> x-t -> y-t -> z-t 的順序排列)

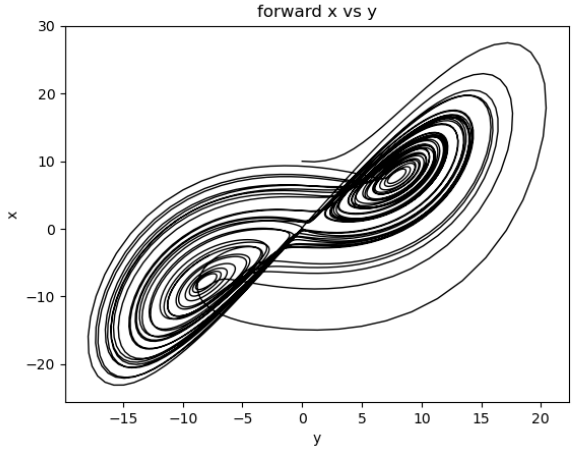


圖 1 forward x-y

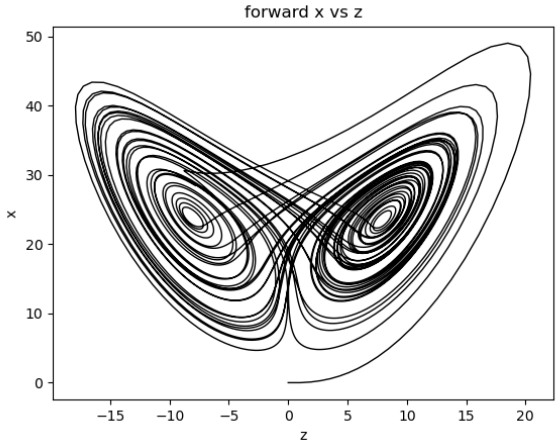


圖 2 forward x-z

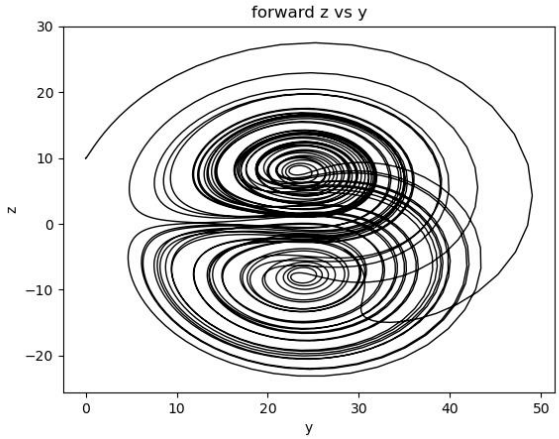


圖 3 forward z-y

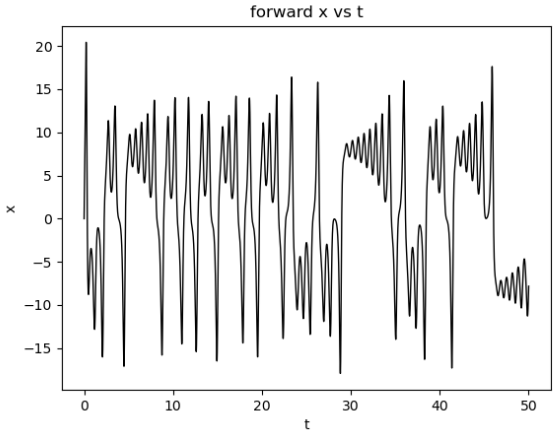


圖 4 forward x-t

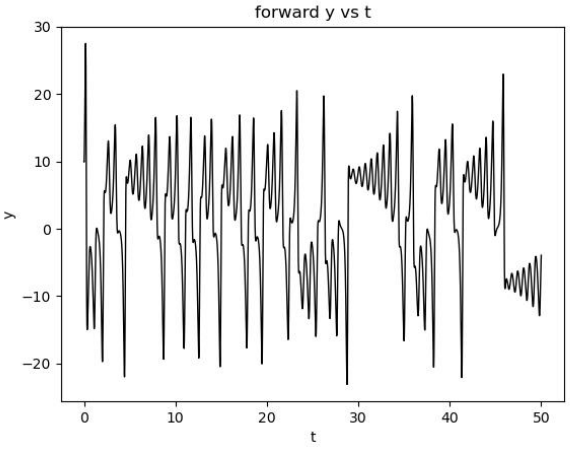


圖 5 forward y-t

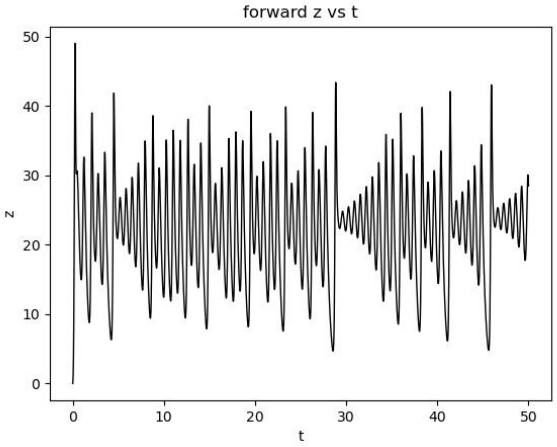


圖 6 forward z-t

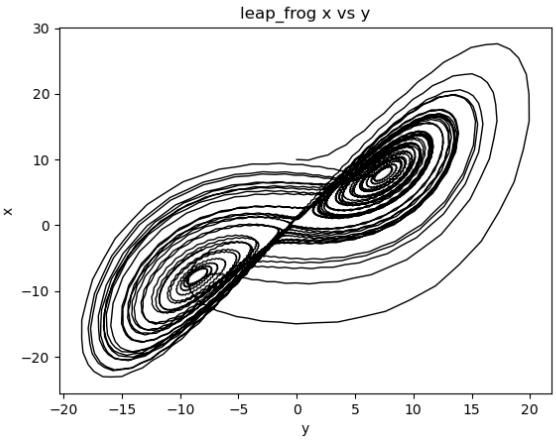


圖 7 leap_frog x-y

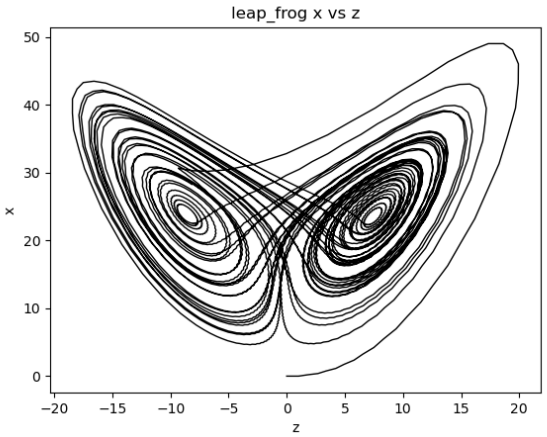


圖 8 leap_frog x-z

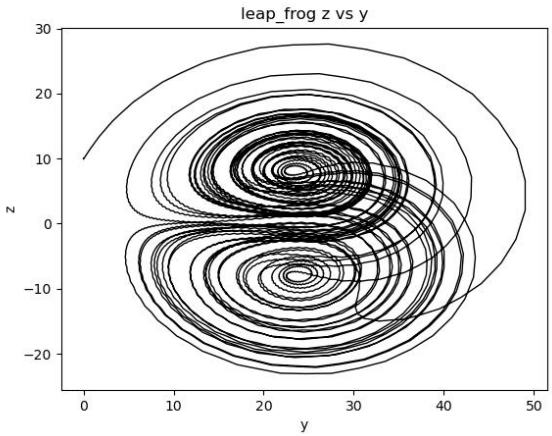


圖 9 leap_frog z-y

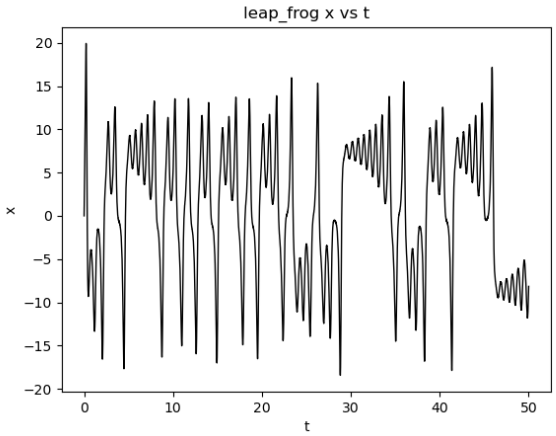


圖 10 leap_frog x-t

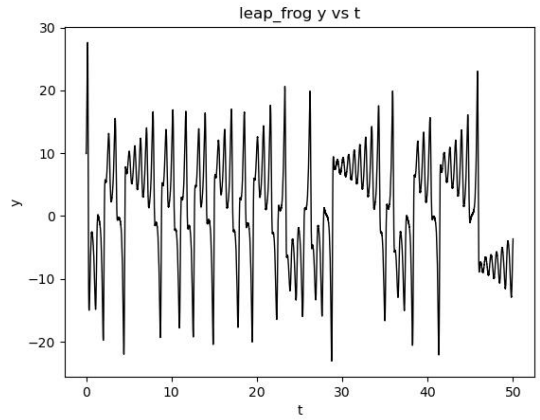


圖 11 leap_frog y-t

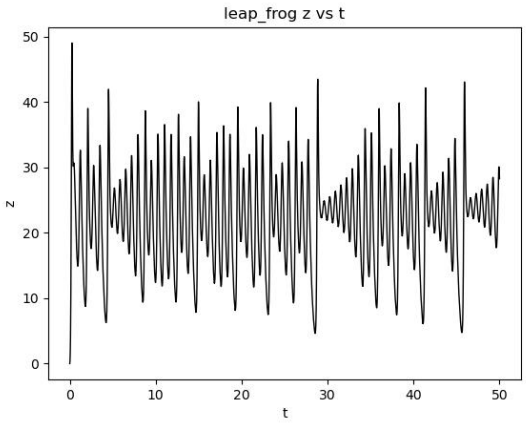


圖 12 leap_frog z-t

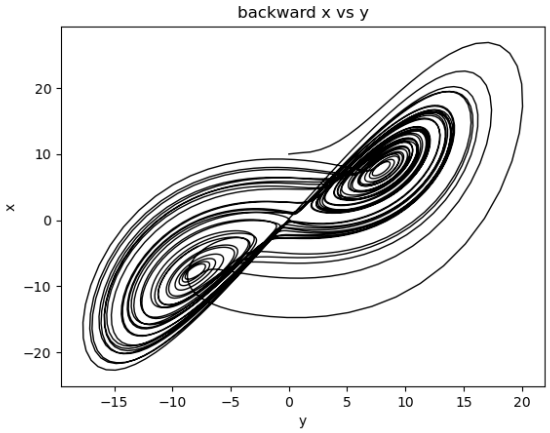


圖 13 backward x-y

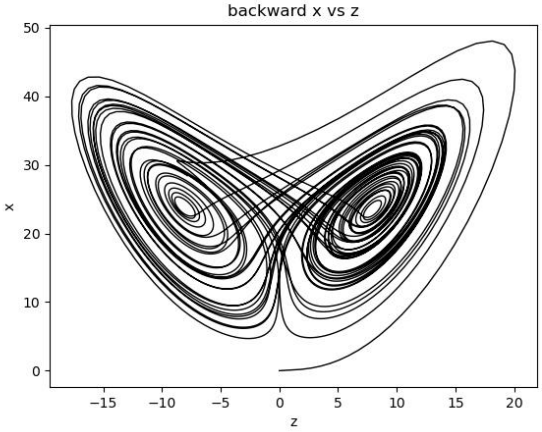


圖 14 backward x-z

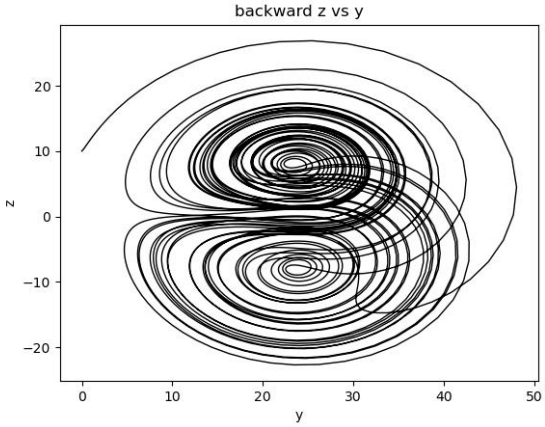


圖 15 backward z-y

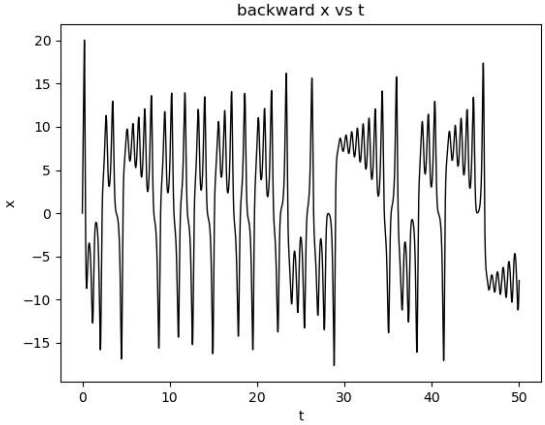


圖 16 backward x-t

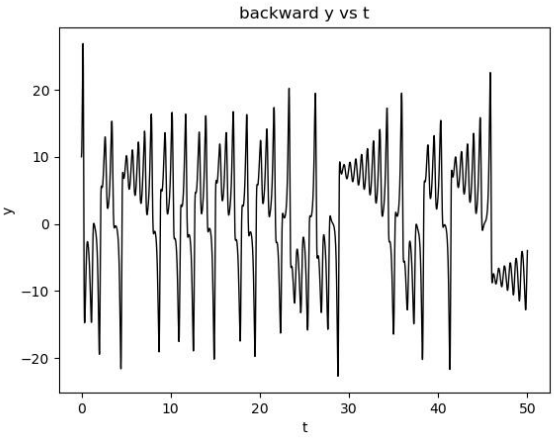


圖 17 backward y-t

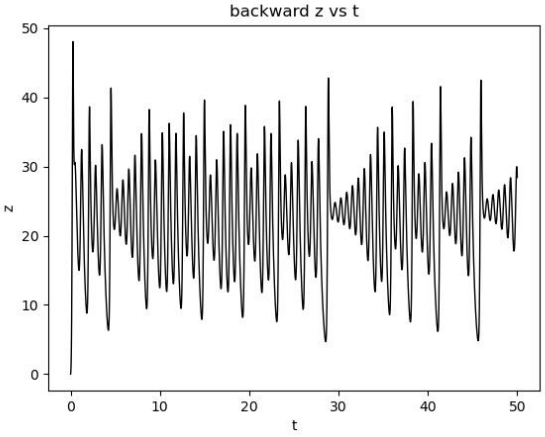


圖 18 backward z-t

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：

$$\frac{dX}{dt} = -s\,X + s\,Y + f\cos\theta$$

$$\frac{dY}{dt} = -X\,Z + r\,X - Y + f\sin\theta$$

$$\frac{dZ}{dt} = X\,Y - b\,Z$$

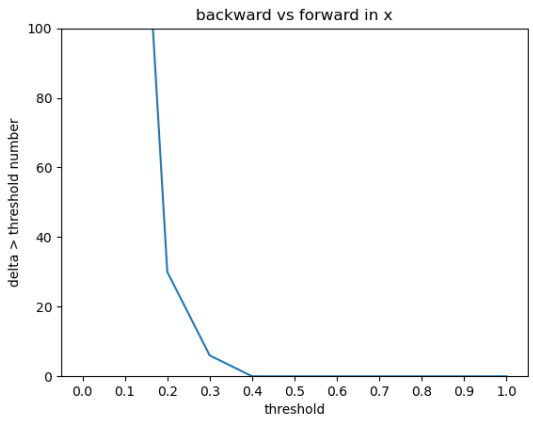
每段的 d(x/y/z) / dt 都可以用給定的(x/y/z)以及其他固定數計算得到，因此可以配合不同的數值積分方式一路推展(x/y/z)以及 d(x/y/z) / dt。然而有趣的地方在於 x/y/z 之間有著一定的關係，繪圖也可發現規律，例如 x-y 的斜 45 度無限型漸進線圖樣、x-z 的雙翼蝴蝶狀圖樣、z-y 的雙核環繞圖樣，然而其值卻非常地不

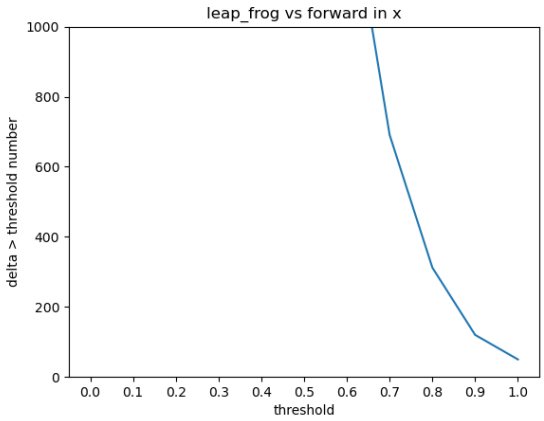
規律，觀察(x/y/z)對 t 的作圖可以發現在大週期波上有很不規律的震盪，但是在小週期上是穩定呈現震幅逐漸加大的震盪，直至某個不固定的閾值在收斂到另外一個不固定的值，類似於數學領域上的碎形理論。

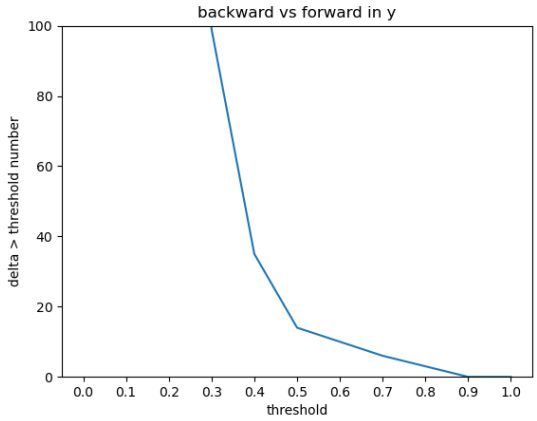
因此在處理類似或更複雜的問題，例如大氣非線性方程時也可能出現類似的情形，也就是非線性帶來的初始值不穩定，會導致後續大小週期震盪的不穩定，並進而引響後續所有參數的值可能有相當大的差距，造成長期預報的不可信。

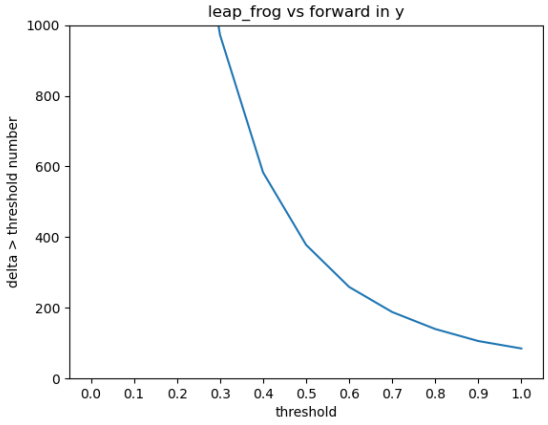
另外值得一提的是各數值積分方法的比較，這部分是多做的，想討論最簡單的 forward 與另外兩種 scheme 的 x/y/z 值差距：

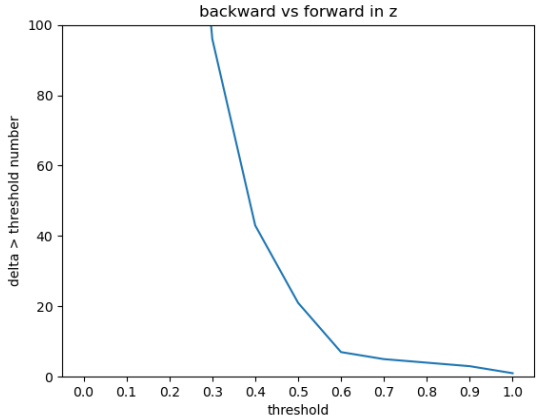
(比較三種方法得出的 5000 個值的差異，y 軸為大於 x 軸閾值的數量)

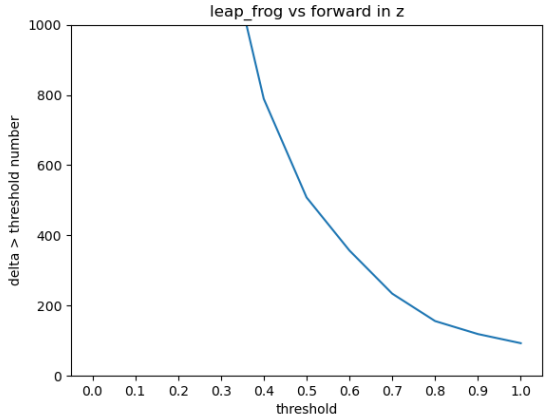












可以看到 forward 跟 backward 之間差距不大，圖樣也差不多，相差的閾值在 0.5 的時候 5000 個值中只有 20 個以下的值相差超過 0.5；然而 leap_frog 卻有相較嚴重的差異，圖樣上在細部也有抖動的情況，估計是時間積分上相較不穩定，即使做了時間濾波仍然沒有更系集集中的表現。

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
times = 5000

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)
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):     #need time filter!!!
    return (x_n_sub1 + f_n*2*dt)
def backward_scheme(x_n, fn_add1, dt=0.01):
    return (x_n + fn_add1*dt)

# 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_image', 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_image', title))
    #plt.show()
    plt.close()

# valid scheme function
def plot_scheme_delta(array1, array2, text, ylim):
    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=ylim)
    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_image', text))
    #plt.show()
    plt.close()


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

    # 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

    # forward part
    scheme = 'forward'
    for i in range(times):
        if i % 1000 == 0:print('{} epoch i='.format(scheme), i)
        # f_n
        dxdt = get_dxdt(x_array[i], y_array[i], sita_array[i])
        dydt = get_dydt(x_array[i], y_array[i], z_array[i], sita_array[
i])
        dzdt = get_dzdt(x_array[i], y_array[i], z_array[i])
        # x_n/f_n
        x_array[i+1] = forward_scheme(x_array[i], dxdt, dt=dt)
        y_array[i+1] = forward_scheme(y_array[i], dydt, dt=dt)
        z_array[i+1] = forward_scheme(z_array[i], dzdt, dt=dt)
    plot_xyz_t(x_array, t_array, title='{} x vs t'.format(scheme))
    plot_xyz_t(y_array, t_array, title='{} y vs t'.format(scheme))
    plot_xyz_t(z_array, t_array, title='{} z vs t'.format(scheme))
    plot_x_y_z(x_array, y_array, title='{} x vs y'.format(scheme))
    plot_x_y_z(x_array, z_array, title='{} x vs z'.format(scheme))
    plot_x_y_z(z_array, y_array, title='{} z vs y'.format(scheme))
    print()

    # copy x/y/z array for next two schemes, and used to check
    x_array_copy = np.copy(x_array)
    y_array_copy = np.copy(y_array)
    z_array_copy = np.copy(z_array)

    # leap_frog part
    scheme = 'leap_frog'
    for i in range(times):
        if i == 0:continue
        if i % 1000 == 0:print('{} epoch i='.format(scheme), i)
        # f_n
        dxdt = get_dxdt(x_array_copy[i], y_array_copy[i], sita_array[i]
)
        dydt = get_dydt(x_array_copy[i], y_array_copy[i], z_array_copy[
i], sita_array[i])
```

```
        dzdt = get_dzdt(x_array_copy[i], y_array_copy[i], z_array_copy[
i])
        # x_n-1/f_n
        x_array[i+1] = leap_frog_scheme(x_array[i-1], dxdt, dt=dt)
        y_array[i+1] = leap_frog_scheme(y_array[i-1], dydt, dt=dt)
        z_array[i+1] = leap_frog_scheme(z_array[i-1], dzdt, dt=dt)
    plot_xyz_t(x_array, t_array, title='{} x vs t'.format(scheme))
    plot_xyz_t(y_array, t_array, title='{} y vs t'.format(scheme))
    plot_xyz_t(z_array, t_array, title='{} z vs t'.format(scheme))
    plot_x_y_z(x_array, y_array, title='{} x vs y'.format(scheme))
    plot_x_y_z(x_array, z_array, title='{} x vs z'.format(scheme))
    plot_x_y_z(z_array, y_array, title='{} z vs y'.format(scheme))

    plot_scheme_delta(x_array, x_array_copy, text='leap_frog vs forward
 in x', ylim=1000)
    plot_scheme_delta(y_array, y_array_copy, text='leap_frog vs forward
 in y', ylim=1000)
    plot_scheme_delta(z_array, z_array_copy, text='leap_frog vs forward
 in z', ylim=1000)
    print()

    # backward part
    scheme = 'backward'
    for i in range(times):
        if i % 1000 == 0:print('{} epoch i='.format(scheme), i)
        # f_n+1
        j = i+1
        dxdt = get_dxdt(x_array_copy[j], y_array_copy[j], sita_array[j]
)
        dydt = get_dydt(x_array_copy[j], y_array_copy[j], z_array_copy[
j], sita_array[j])
        dzdt = get_dzdt(x_array_copy[j], y_array_copy[j], z_array_copy[
j])
        # x_n/f_n+1
        x_array[i+1] = backward_scheme(x_array_copy[i], dxdt, dt=dt)
        y_array[i+1] = backward_scheme(y_array_copy[i], dydt, dt=dt)
        z_array[i+1] = backward_scheme(z_array_copy[i], dzdt, dt=dt)
    plot_xyz_t(x_array, t_array, title='{} x vs t'.format(scheme))
```

```
    plot_xyz_t(y_array, t_array, title='{} y vs t'.format(scheme))
    plot_xyz_t(z_array, t_array, title='{} z vs t'.format(scheme))
    plot_x_y_z(x_array, y_array, title='{} x vs y'.format(scheme))
    plot_x_y_z(x_array, z_array, title='{} x vs z'.format(scheme))
    plot_x_y_z(z_array, y_array, title='{} z vs y'.format(scheme))

    plot_scheme_delta(x_array, x_array_copy, text='backward vs forward
in x', ylim=100)
    plot_scheme_delta(y_array, y_array_copy, text='backward vs forward
in y', ylim=100)
    plot_scheme_delta(z_array, z_array_copy, text='backward vs forward
in z', ylim=100)
    print()
```