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)

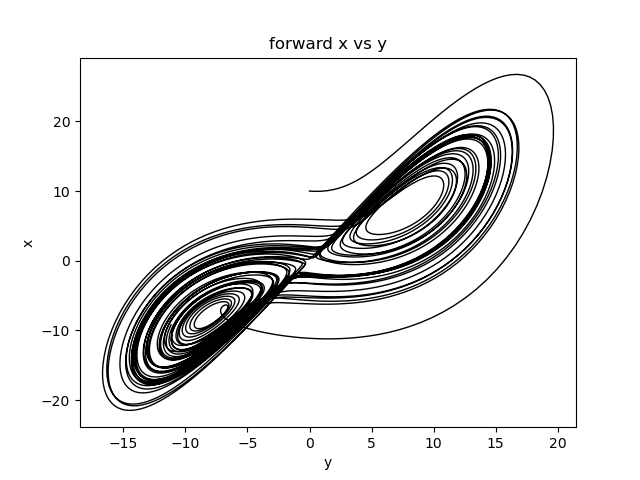
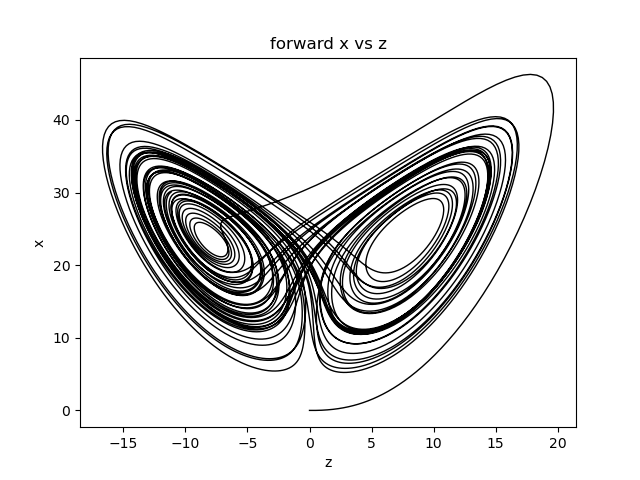
 

圖1 forward x-y 圖2 forward x-z

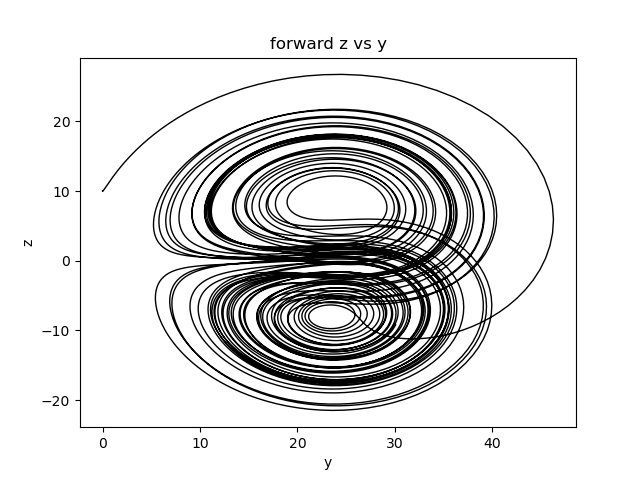
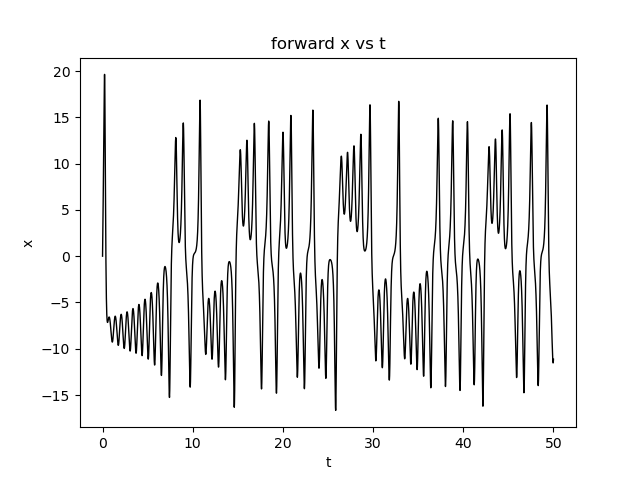
 

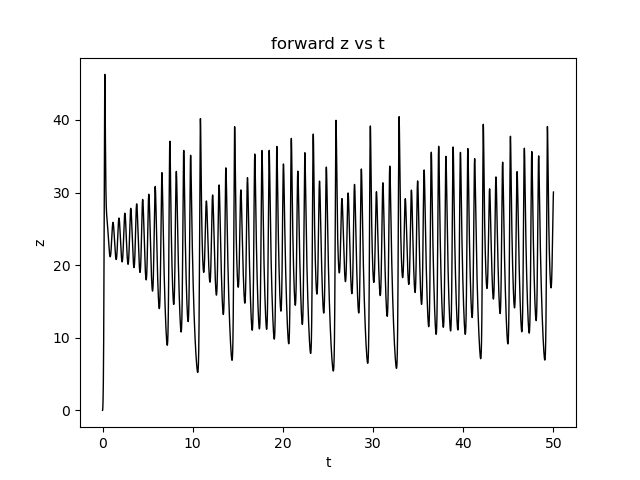
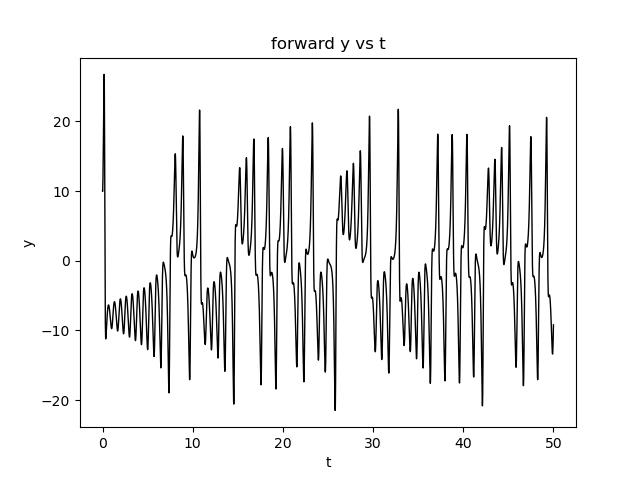
圖3 forward z-y 圖4 forward x-t

圖5 forward y-t 圖6 forward z-t

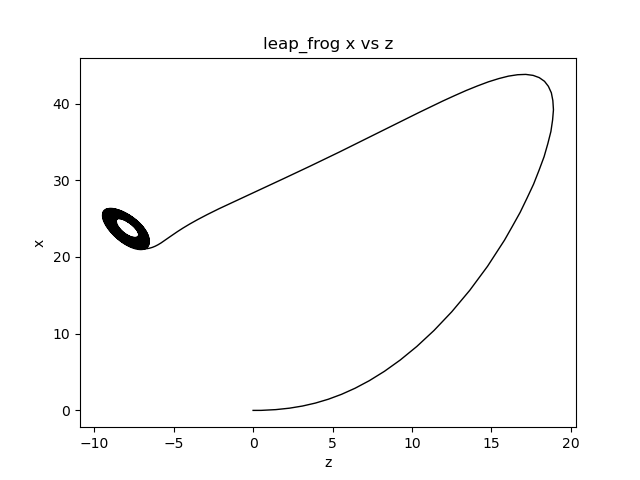
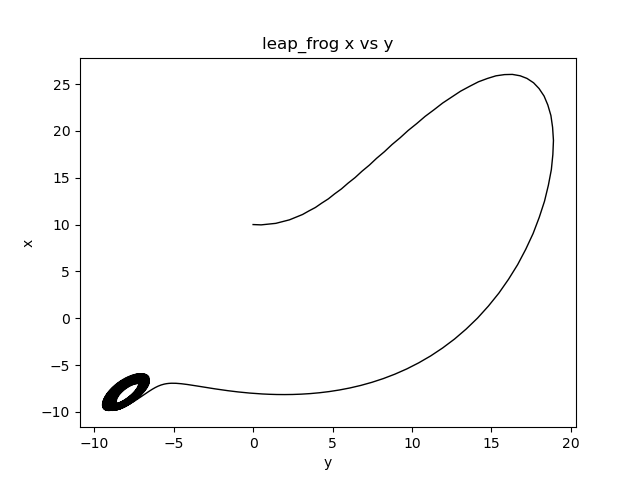


圖7 leap\_frog x-y 圖8 leap\_frog x-z

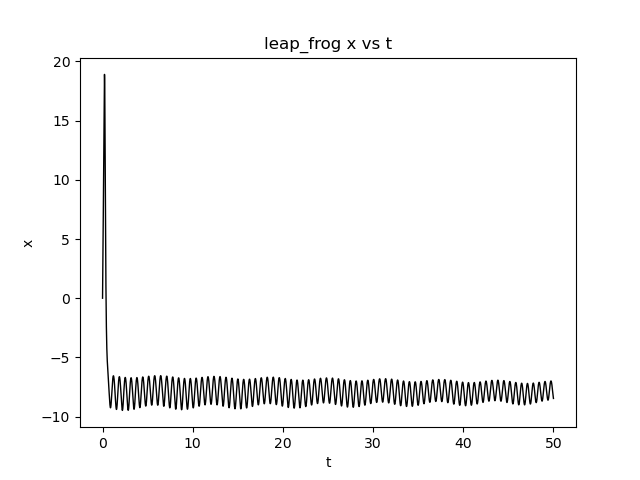
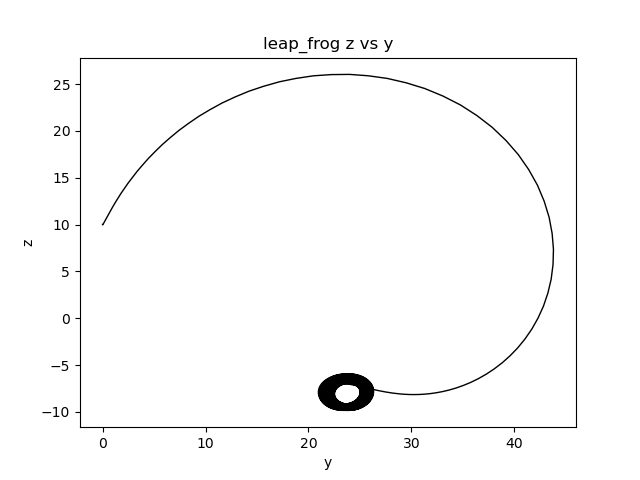


圖9 leap\_frog z-y 圖10 leap\_frog x-t

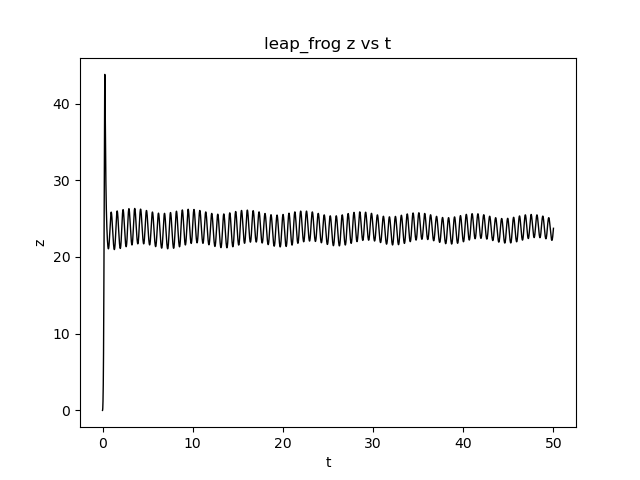
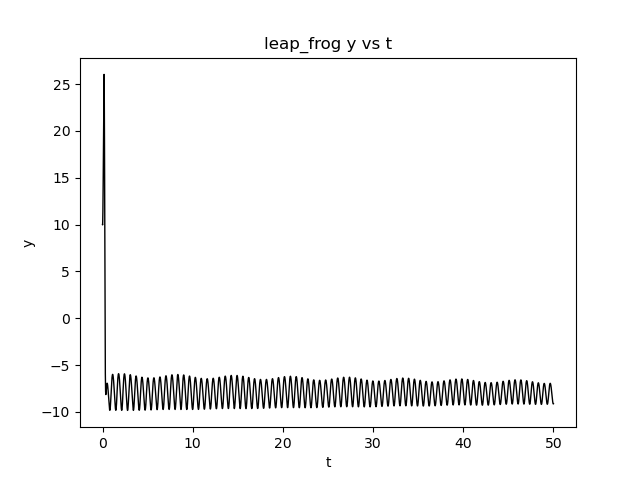


圖11 leap\_frog y-t 圖12 leap\_frog z-t

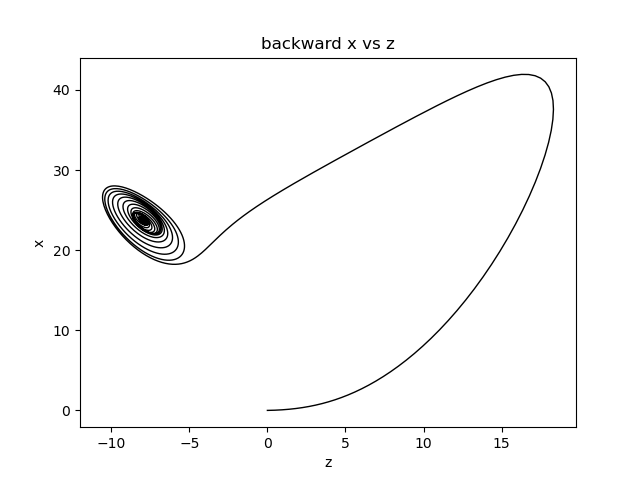
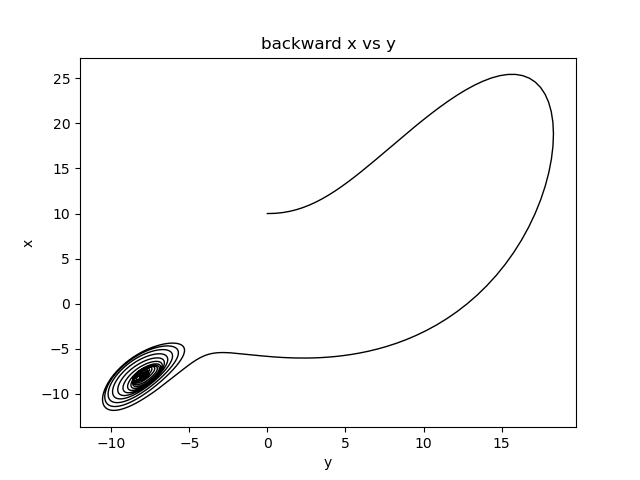


圖13 backward x-y 圖14 backward x-z

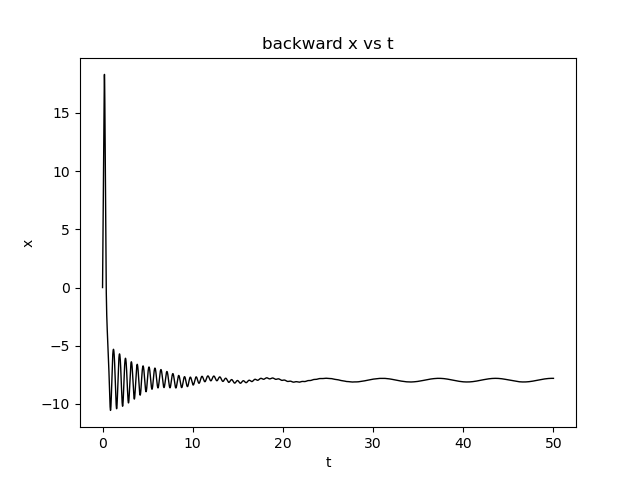
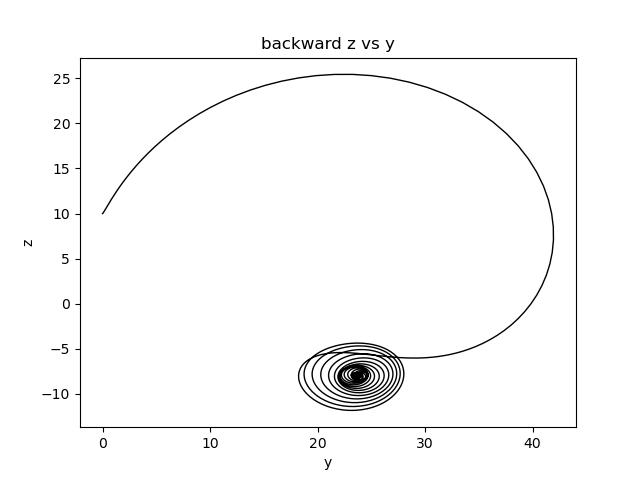


圖15 backward z-y 圖16 backward x-t

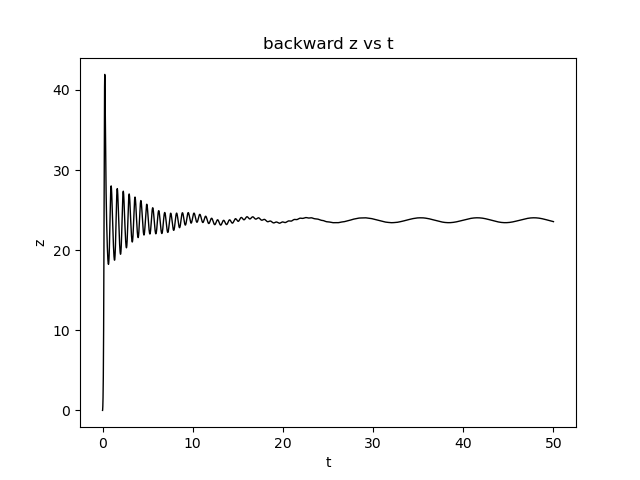
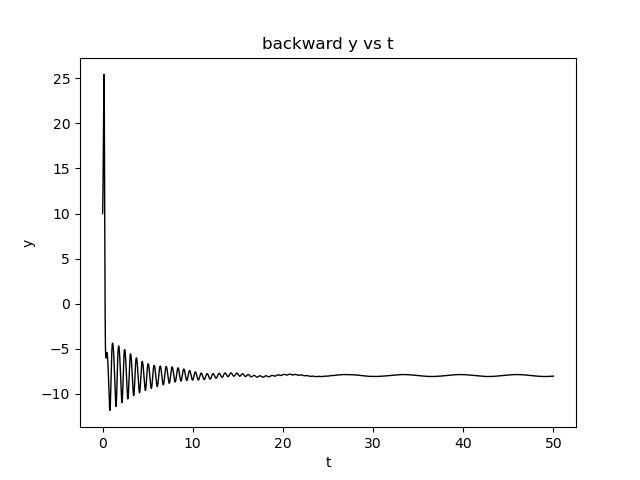
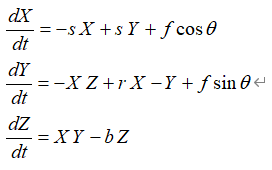


圖17 backward y-t 圖18 backward z-t

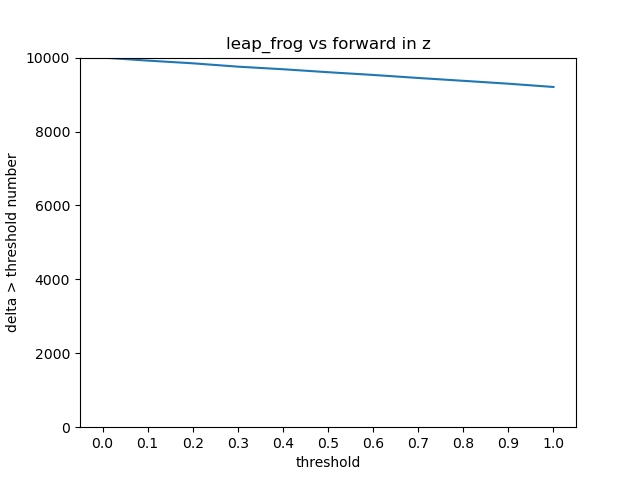
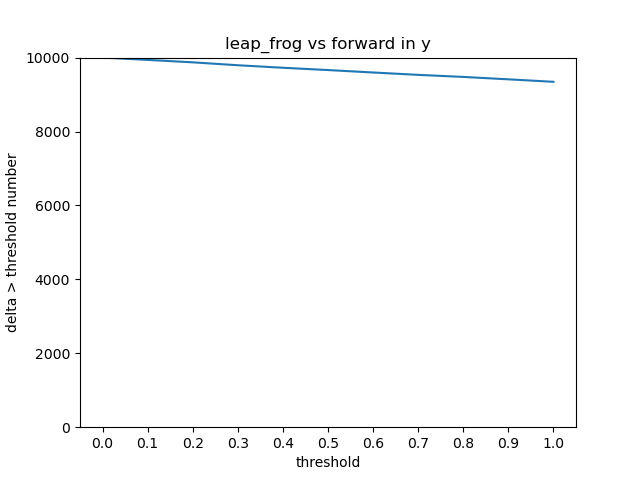
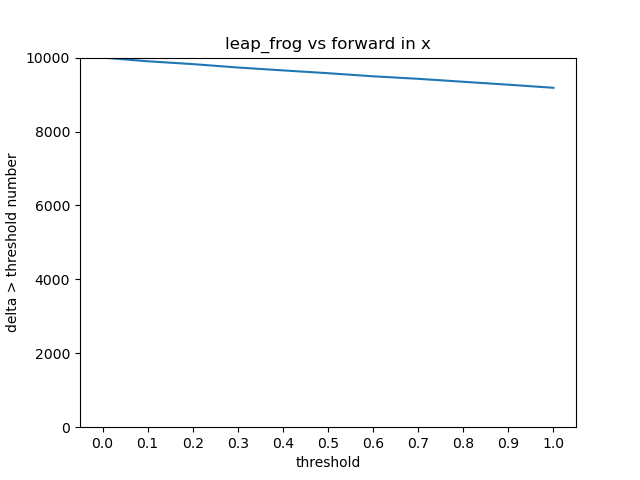
1. 討論2小題：藉由Lorenz 方程討論數值積分非線性方程時出現的問題，對數值天氣預報會有甚麼問題?

這個作業採用了三個非線性微分方程的系統如下：

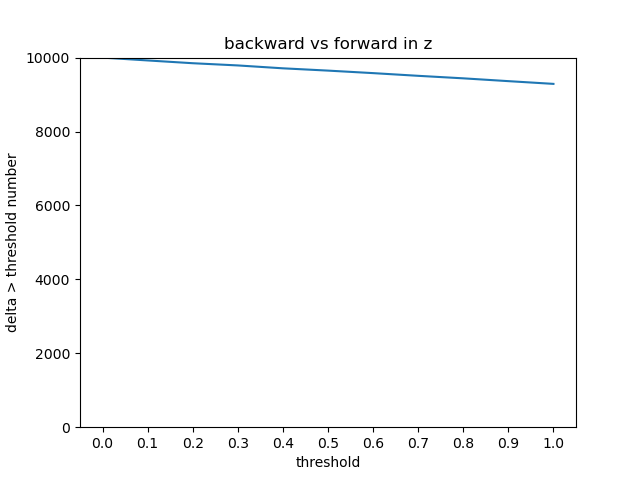
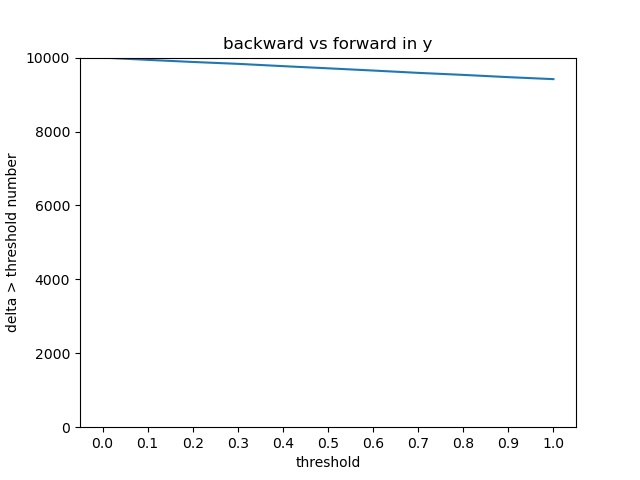
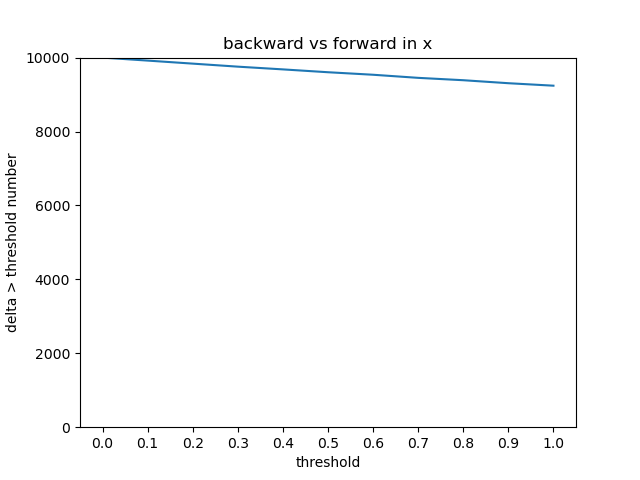
每段的d(x/y/z) / dt都可以用給定的(x/y/z)以及其他固定數計算得到，因此可以配合不同的數值積分方式一路推展(x/y/z)以及d(x/y/z) / dt。然而有趣的地方在於x/y/z之間有著一定的關係，繪圖也可發現規律，而且不同的積分方法會有很不同的規律樣式，例如forward的x-y斜45度無限型漸進線圖樣、x-z的雙翼蝴蝶狀圖樣、z-y的雙核環繞圖樣，或是backward以及leap\_frog的x-y、x-z、z-y的螺旋漸進圖樣(兩者在收斂的速度不同)，然而其值卻非常地不規律，forward (x/y/z)對t的作圖可以發現在大週期波上有很不規律的震盪，但是在小週期上是穩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。

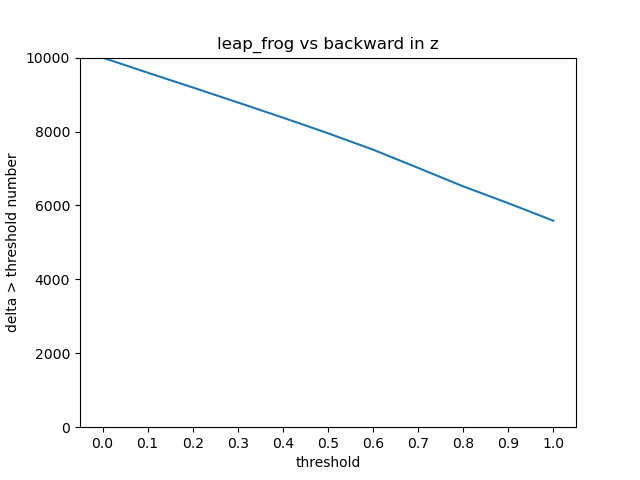
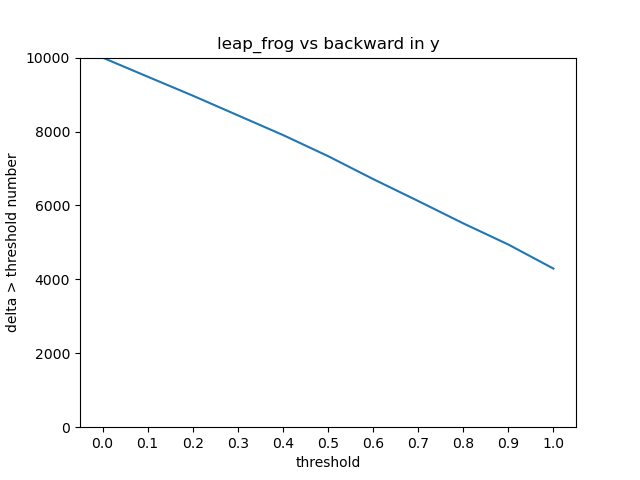
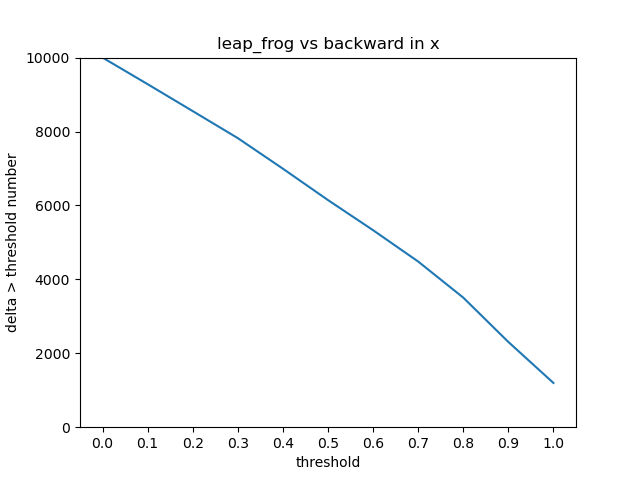
另外值得一提的是各數值積分方法的比較，這部分是學生多做的，目的是想討論三種scheme的x/y/z值差距：比較三種方法得出的10000個值的差異，y軸為大於x軸閾值的數量：



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。

1. 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)

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), np.copy(x\_array)

    y\_array\_forward, y\_array\_leap\_frog, y\_array\_backward = np.copy(y\_array), np.copy(y\_array), np.copy(y\_array)

    z\_array\_forward, z\_array\_leap\_frog, z\_array\_backward = np.copy(z\_array), np.copy(z\_array), np.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], sita\_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\_array[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\_array[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], sita\_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')