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)

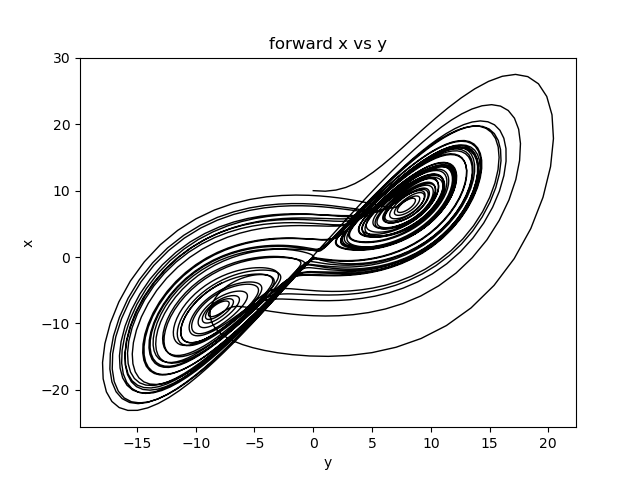
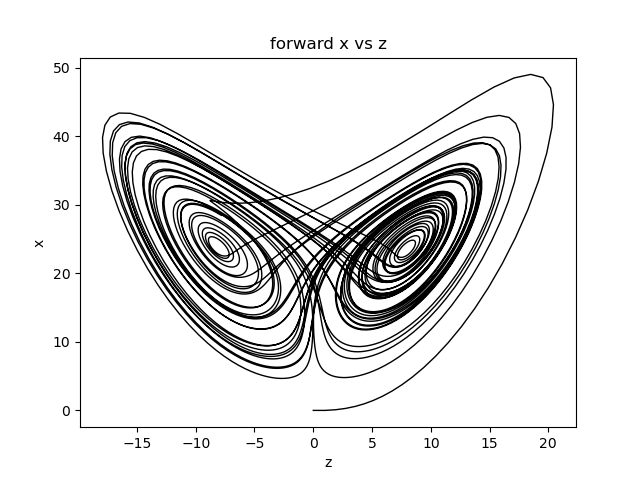
 

圖1 forward x-y 圖2 forward x-z

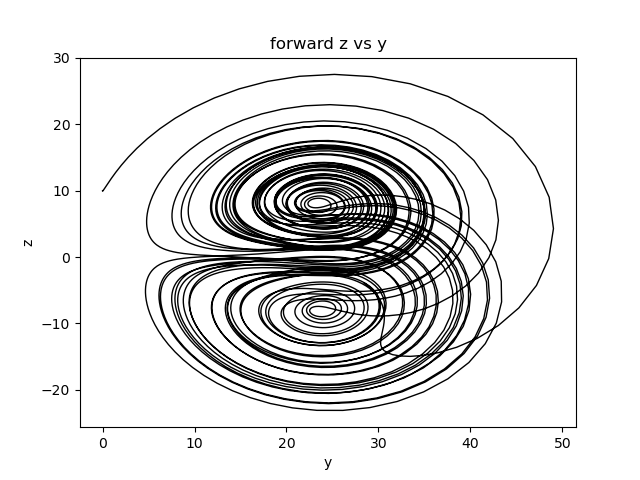
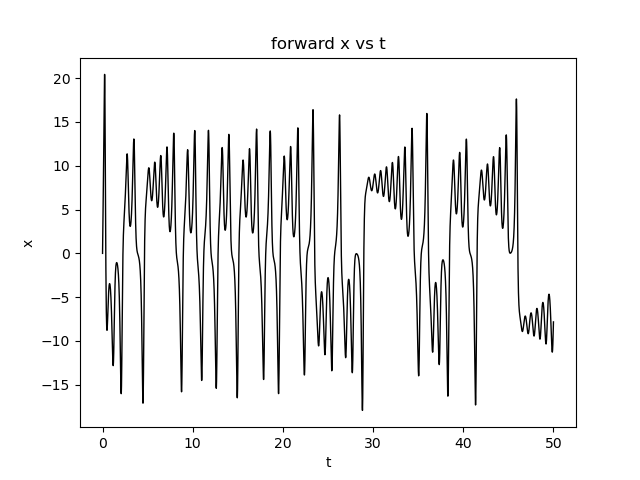
 

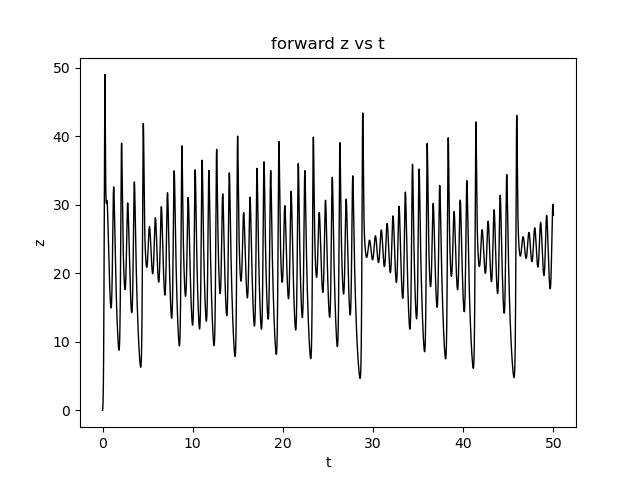
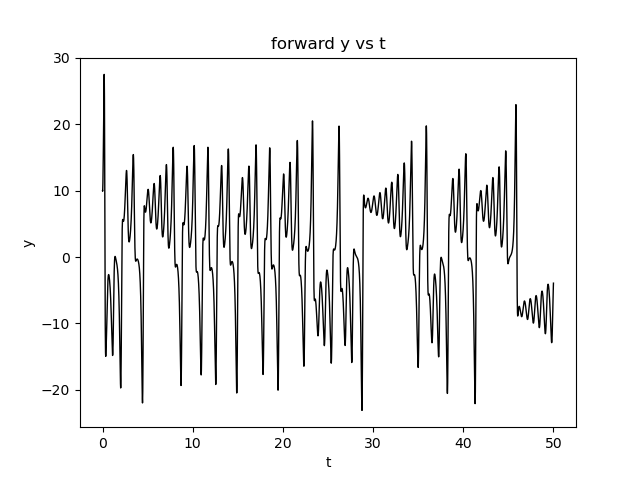
圖3 forward z-y 圖4 forward x-t

圖5 forward y-t 圖6 forward z-t

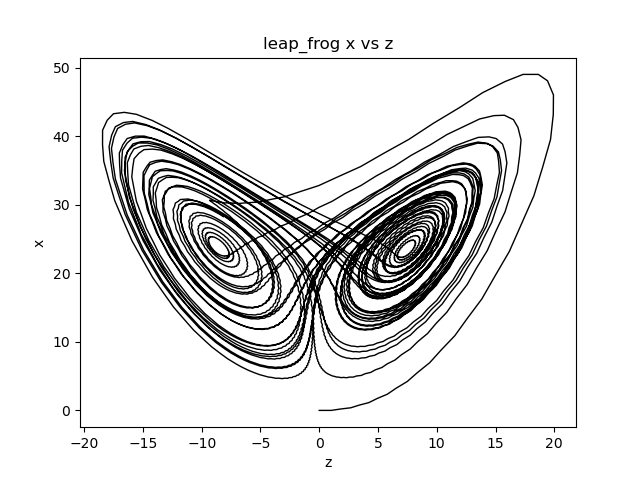
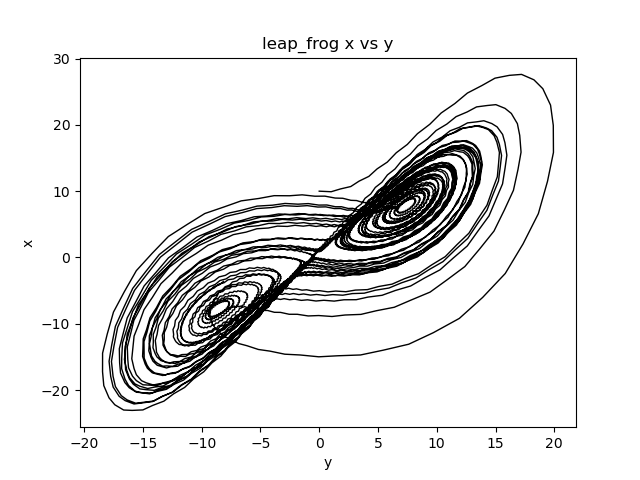


圖7 leap\_frog x-y 圖8 leap\_frog x-z

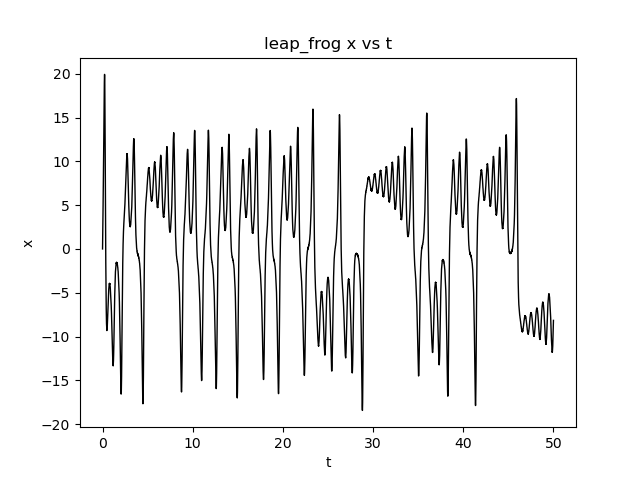
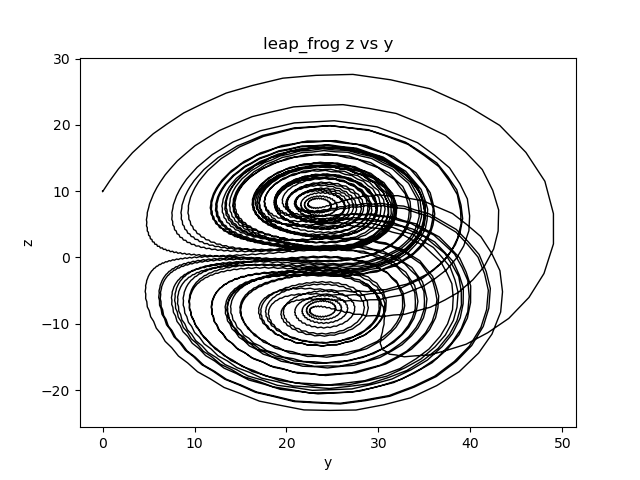


圖9 leap\_frog z-y 圖10 leap\_frog x-t

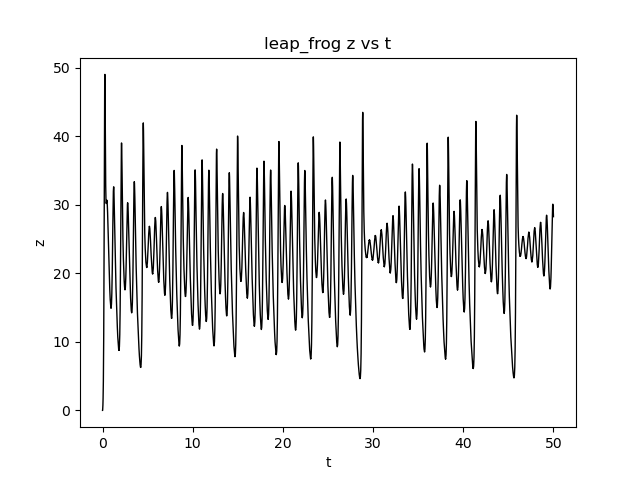
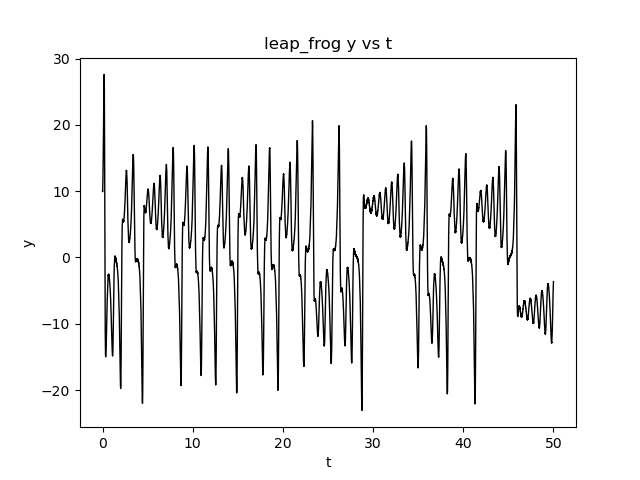


圖11 leap\_frog y-t 圖12 leap\_frog z-t

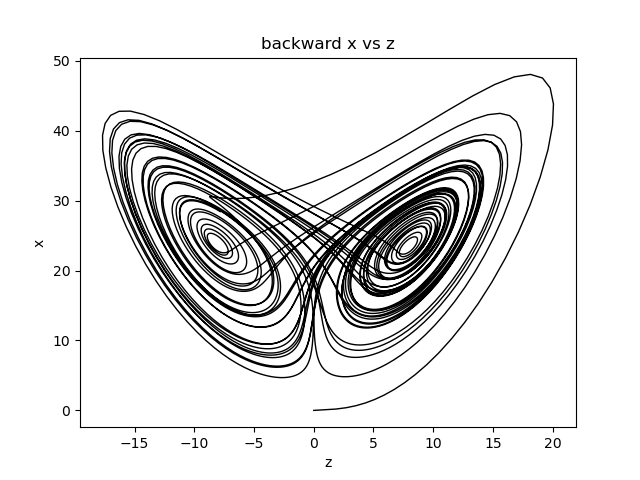
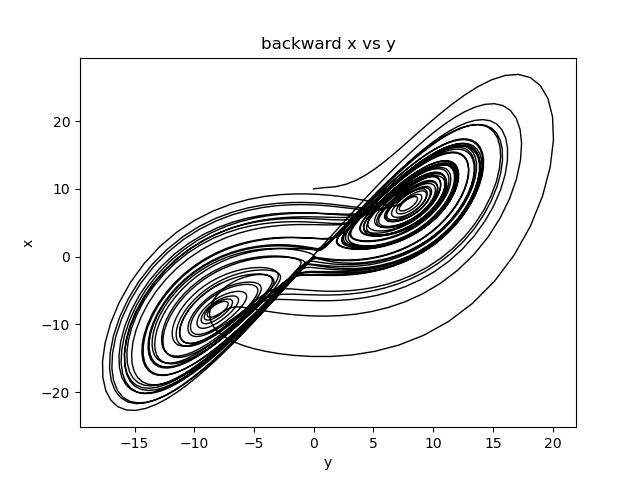


圖13 backward x-y 圖14 backward x-z

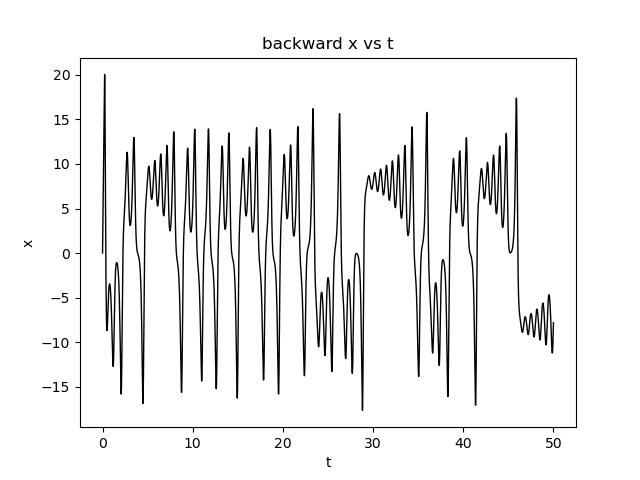
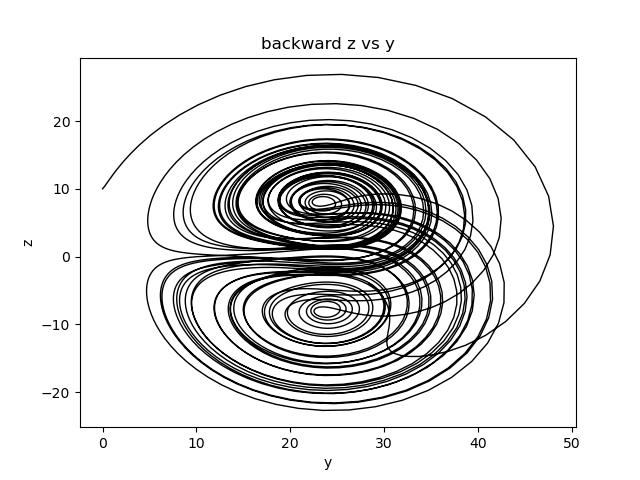


圖15 backward z-y 圖16 backward x-t

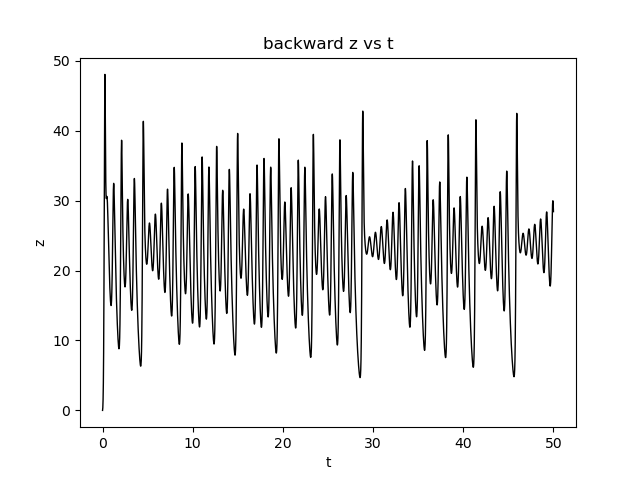
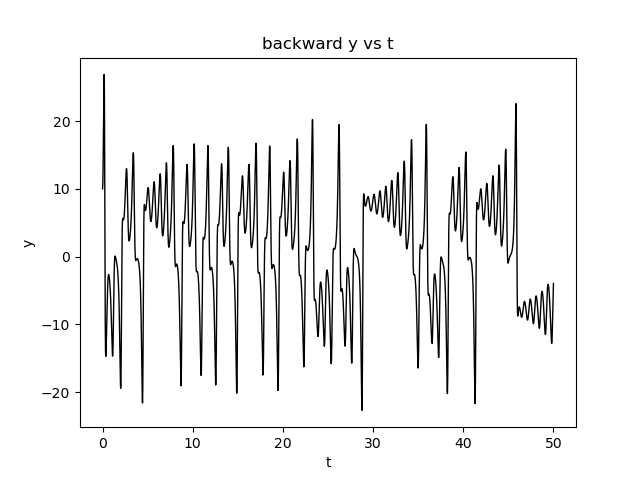
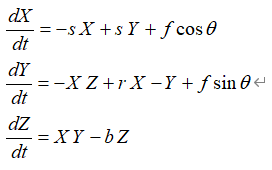


圖17 backward y-t 圖18 backward z-t

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?

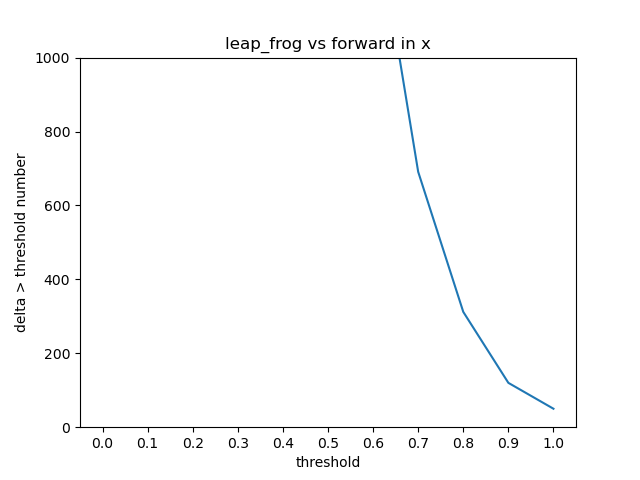
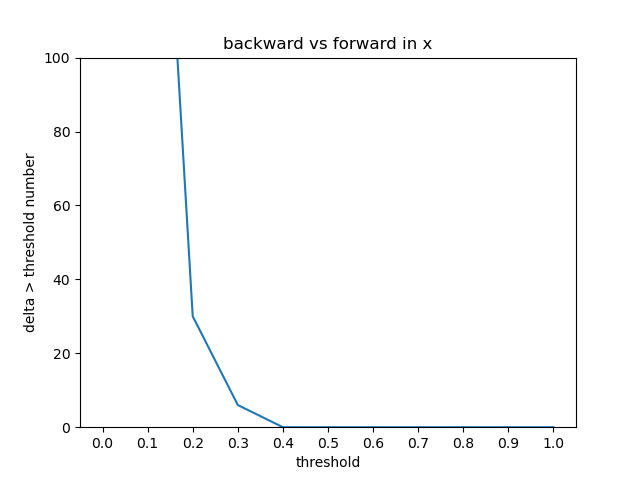
這個作業採用了三個非線性微分方程的系統如下：

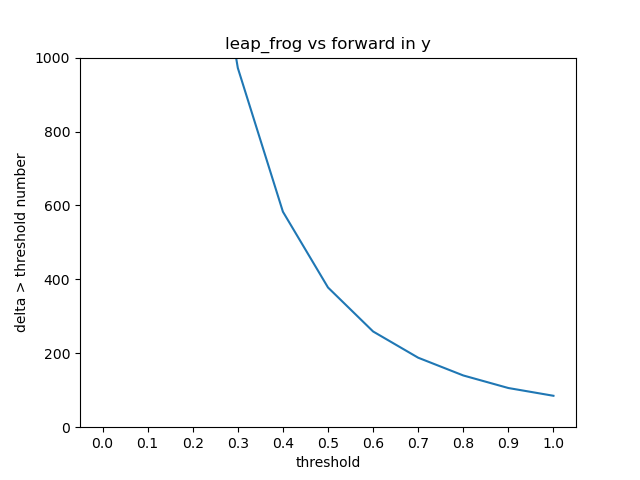
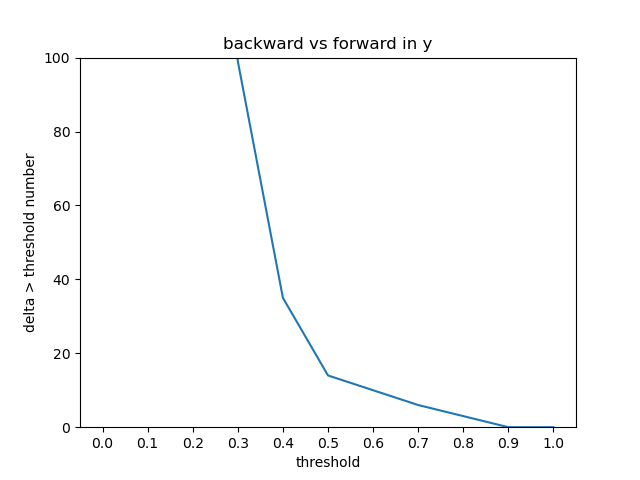
每段的d(x/y/z) / dt都可以用給定的(x/y/z)以及其他固定數計算得到，因此可以配合不同的數值積分方式一路推展(x/y/z)以及d(x/y/z) / dt。然而有趣的地方在於x/y/z之間有著一定的關係，繪圖也可發現規律，例如x-y的斜45度無限型漸進線圖樣、x-z的雙翼蝴蝶狀圖樣、z-y的雙核環繞圖樣，然而其值卻非常地不規律，觀察(x/y/z)對t的作圖可以發現在大週期波上有很不規律的震盪，但是在小週期上是穩定呈現震幅逐漸加大的震盪，直至某個不固定的閾值在收斂到另外一個不固定的值，類似於數學領域上的碎形理論。

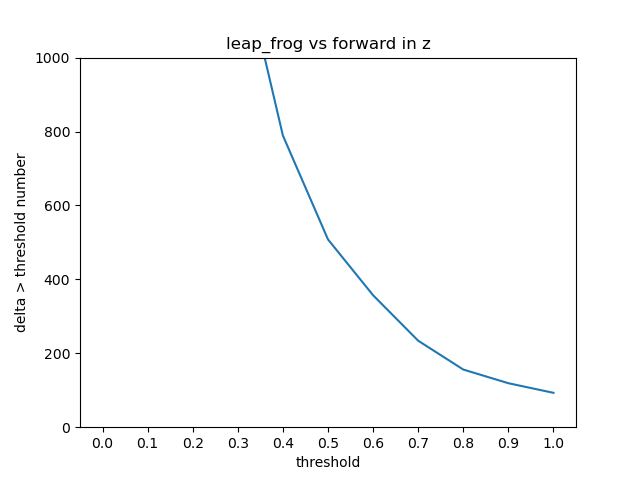
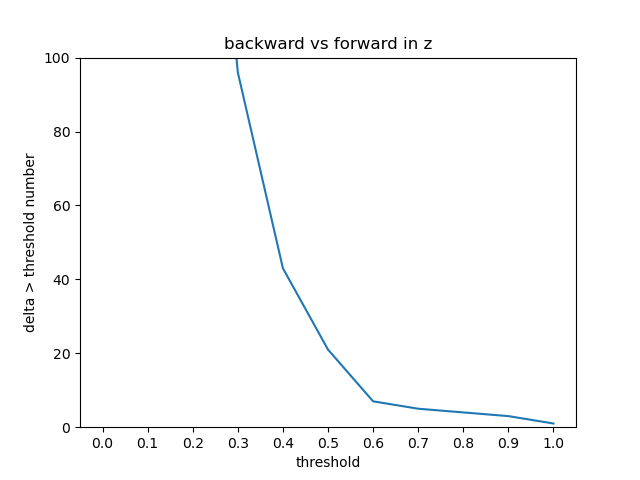
因此在處理類似或更複雜的問題，例如大氣非線性方程時也可能出現類似的情形，也就是非線性帶來的初始值不穩定，會導致後續大小週期震盪的不穩定，並進而引響後續所有參數的值可能有相當大的差距，造成長期預報的不可信。

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。

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

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()