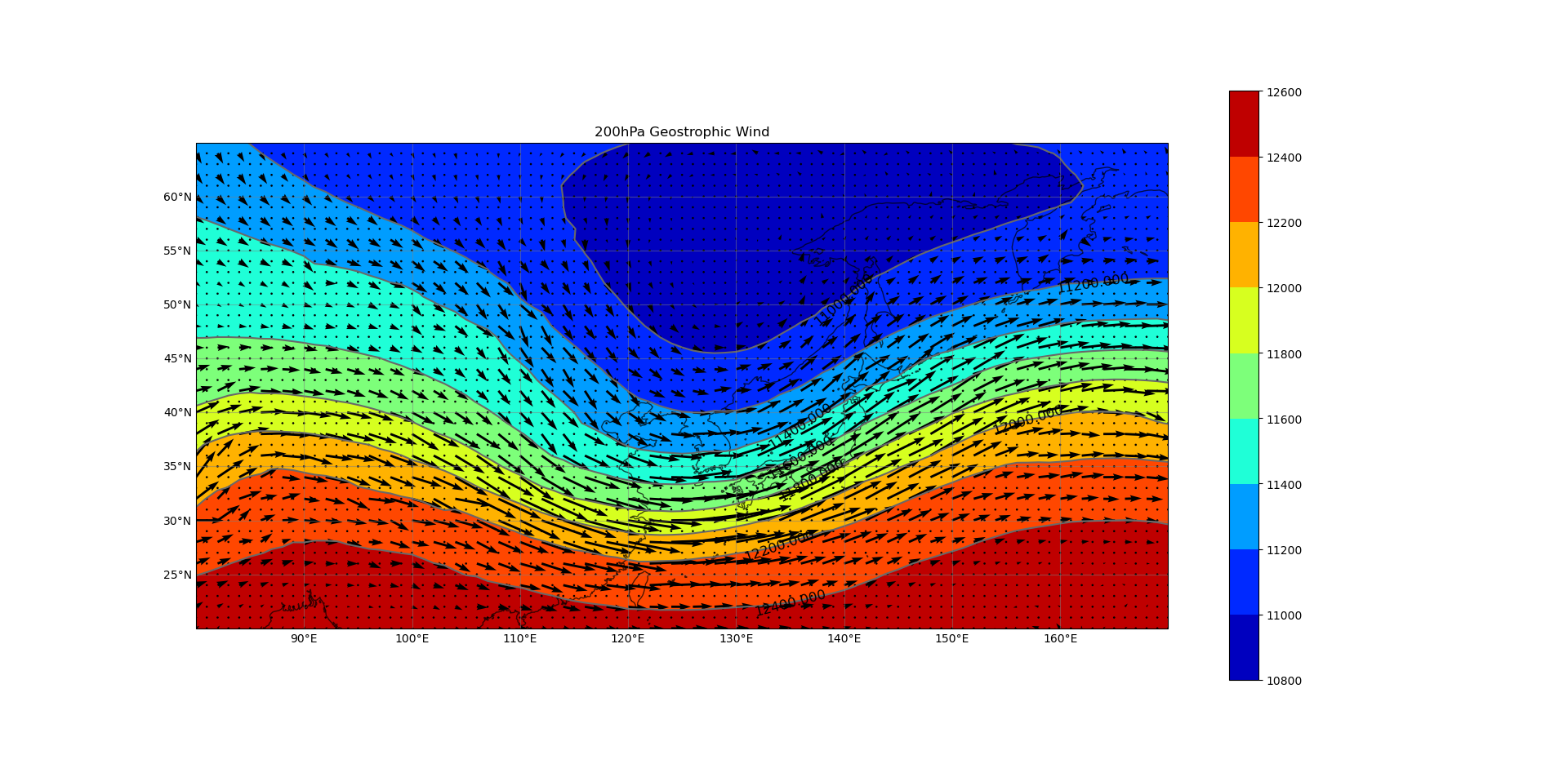
天氣學下hw3 大氣4A 黃展皇 106601015

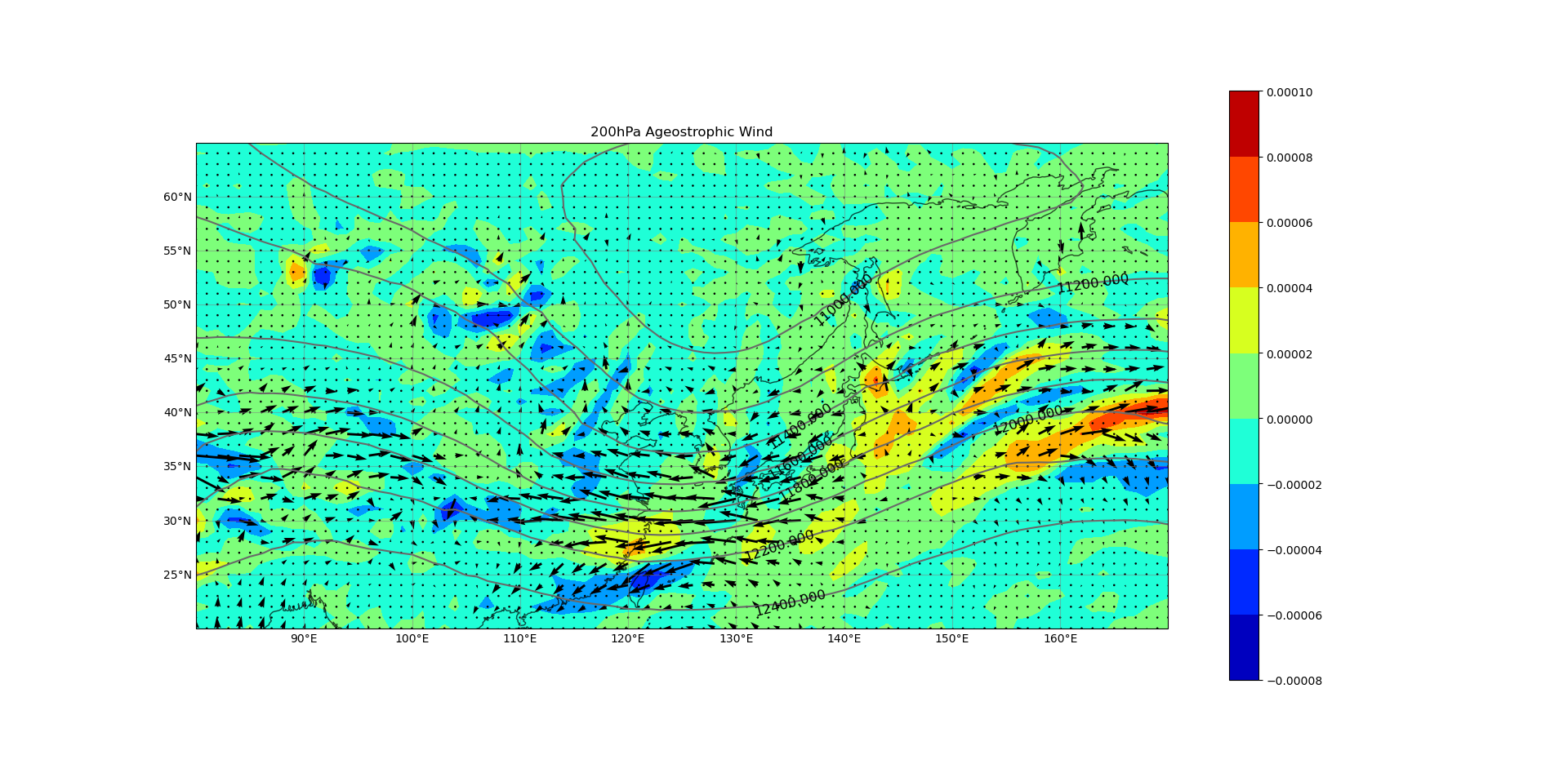
工作環境：x64 windows10，conda 4.8.3，python3.6.10

Requestment：os、numpy、matplotlib、math、sys、cartopy

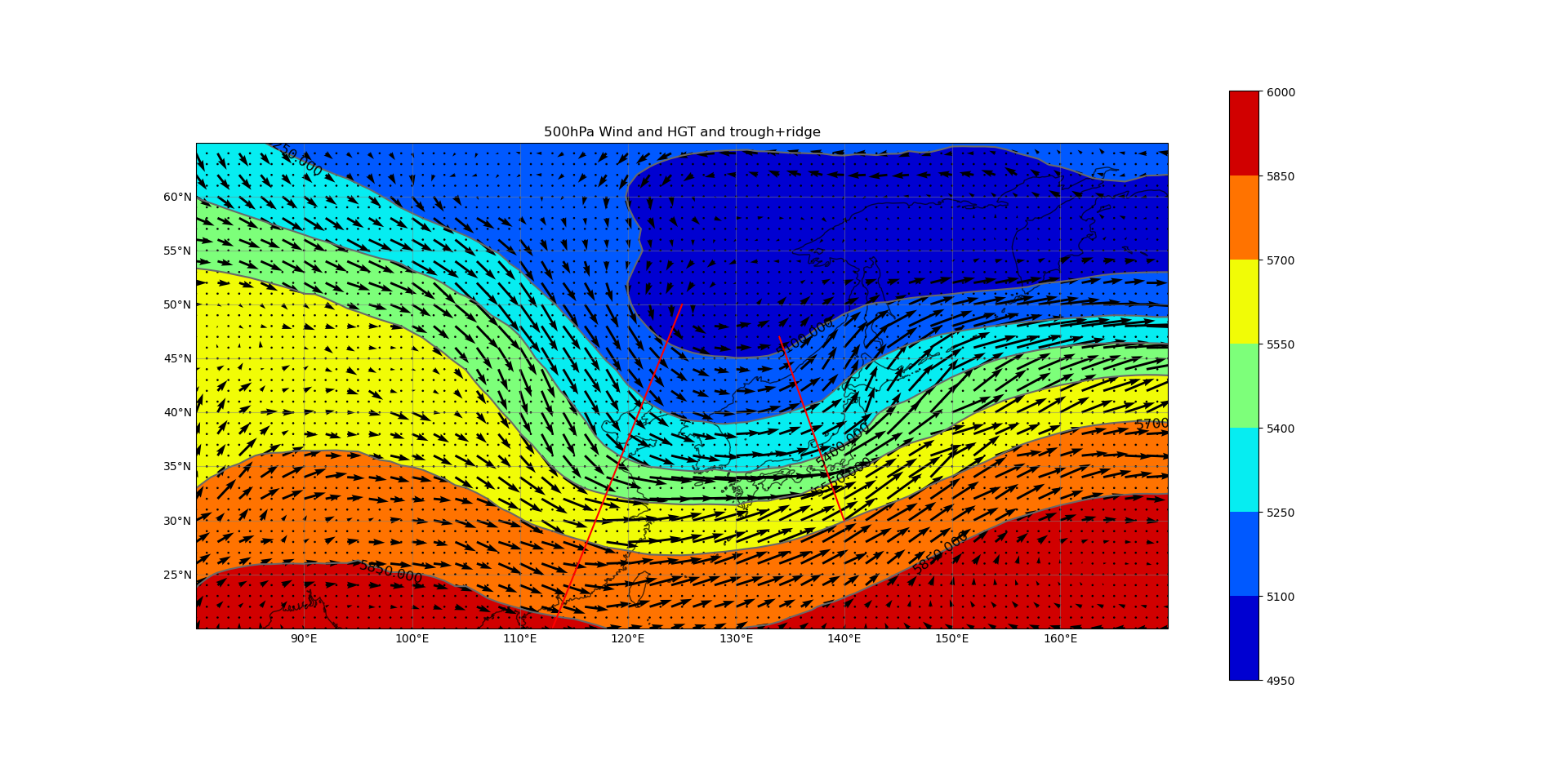
1. 200 hPa地轉風與重力位高度場



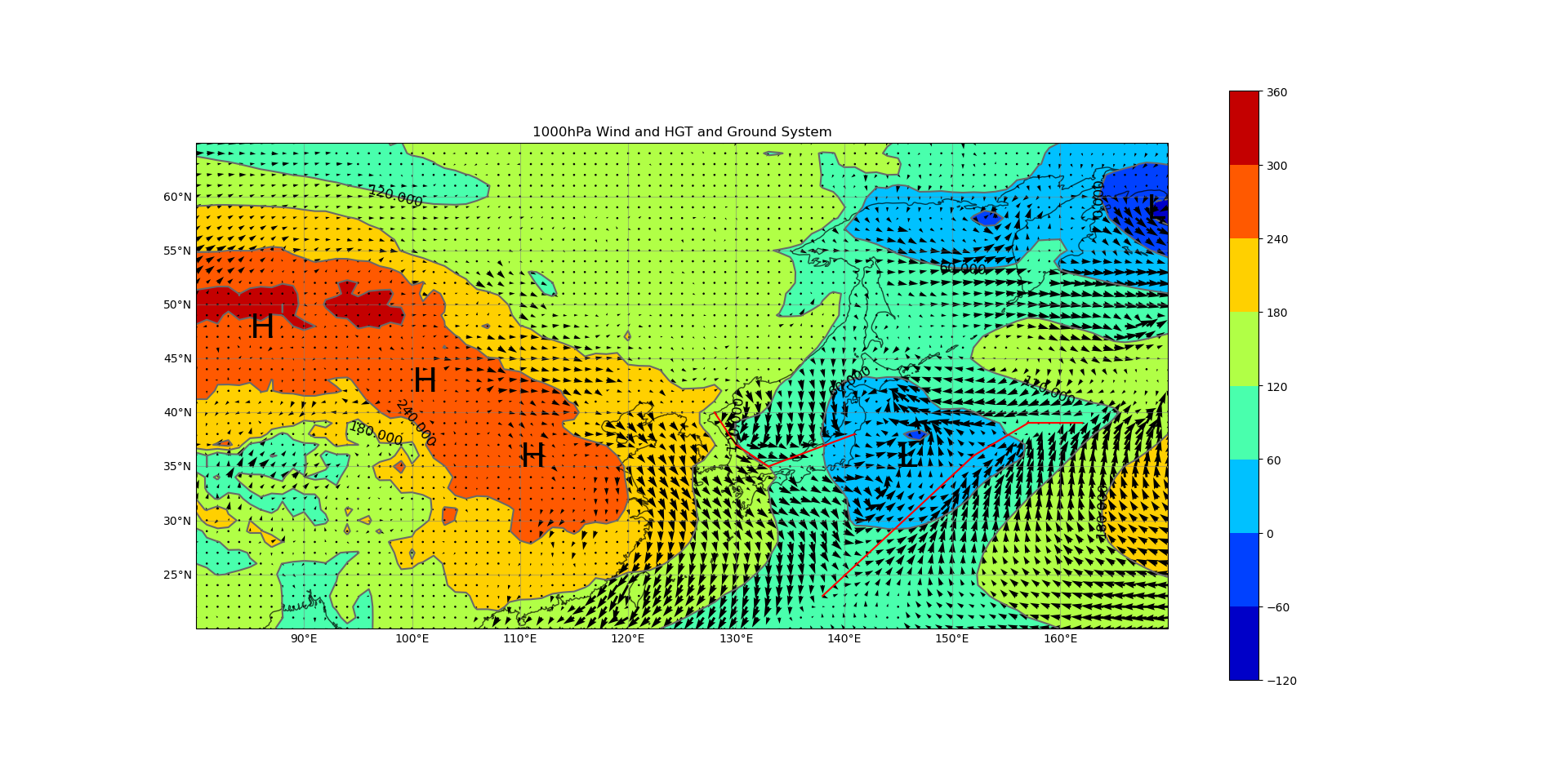
(二)200 hPa非地轉風&輻合輻散與重力位高度場



(三)500 hPa風場與重力位高度場(標示槽脊線)



(四)1000 hPa風場與高度場 (需附加地面系統標示)



(五)問題討論：說明此時的天氣系統高層與低層的關係，如200 hPa噴流軸出區入區，500 hPa槽脊系統的配置如何對地面天氣系統造成影響。

如上圖(一)所示，200hPa的重力位高度在120E~130E之間有個明顯的低壓槽，可發現地轉風Vg因為氣壓梯度增加而增強造成噴流，再由於流體連續性而導致圖(二)中槽前高空(噴流入區)普遍輻合、槽後高空(噴流出區)普遍輻散的情況，引發非地轉風Vag補償作用的東風分量，合成較Vg弱的V實際風場，而160E~170E則是相反的情況。

200hPa的配置對應到500hPa為低壓槽與兩條槽線，對應至地面則分割高壓與低壓帶，槽前(日本東方)為低壓而槽後(中國)為高壓，高低壓之間產生風向風速不連續的鋒面帶，但由於台灣上空高空為輻合，不利於垂直運動，應該會受到大陸冷高壓影響而為乾冷的天氣型態。

(六)計算與繪圖程式碼 + 註解

import os

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.ticker as mticker

import math

import cartopy.crs as ccrs

from cartopy.mpl.gridliner import LONGITUDE\_FORMATTER, LATITUDE\_FORMATTER

import sys

print(sys.prefix) # show what virtual env I am in

# read binary data, analyze to 91x\*46y\*3(200 500 1000)\*3(HGT U V) = 37674 np.array, return wanted plane data

def read\_bindata\_return\_wanted(root\_path, filename, pressure, parameter):

    all\_data = np.fromfile(root\_path+filename, dtype='<f4', count=-1, sep='')

    init\_lndex = len(pressure\_list)\*x\_num\*y\_num\*parameter\_list.index(parameter) + x\_num\*y\_num\*pressure\_list.index(pressure)

    return all\_data[init\_lndex:init\_lndex+x\_num\*y\_num]

# Input y and output the corresponding latitude coordinates

def y\_to\_lat(y):

    return lat\_lower+y\*delta

# Input pre, post, and d and return interpolation differential.

def median\_interpolation(front, behind, d):

    return (behind-front)/(2\*d)

# Input pre, here, and d and return the pre-interpolated differential.

def front\_interpolation(front, here, d):

    return (here-front)/d

# Input here value and post value and return post-interpolation differential.

def behind\_interpolation(here, behind, d):

    return (behind-here)/d

# Sparse quiver dataset(Sparsed points = 0.0)

def sparse(dataset, num\_to\_one=1):

    # call by values

    new\_dataset = np.copy(dataset)

    for y in range(new\_dataset.shape[0]):

        for x in range(new\_dataset.shape[1]):

            if y%num\_to\_one == 0 and x%num\_to\_one == 0:pass

            else:new\_dataset[y, x] = 0.0

    return new\_dataset

# Enter contour/contourf/quiver data, draw on a map and save it.

def plot\_on\_map(title, contour=[], contourf=[], quiver\_u=[], quiver\_v=[], num\_to\_one=1, scale=1000):

    # Create x, y grid points

    x = np.arange(lon\_lower, lon\_upper+delta, delta)

    y = np.arange(lat\_lower, lat\_upper+delta, delta)

    X, Y = np.meshgrid(x, y)

    # Set map parameters: projection, coastline resolution, grid, off right and above longitude and latitude, format latitude and longitude.

    ax = plt.axes(projection=ccrs.PlateCarree())

    ax.coastlines(resolution='50m', alpha=0.7)

    gl = ax.gridlines(crs=ccrs.PlateCarree(), draw\_labels=True, color='gray', alpha=0.5)

    gl.xlocator = mticker.FixedLocator(np.arange(lon\_lower, lon\_upper+10, 10))

    gl.ylocator = mticker.FixedLocator(np.arange(lat\_lower, lat\_upper+5, 5))

    gl.xlabels\_top = False

    gl.ylabels\_right = False

    gl.xformatter = LONGITUDE\_FORMATTER

    gl.yformatter = LATITUDE\_FORMATTER

    # Plot and set contour

    if contour != []:

        C = plt.contour(X, Y, contour, colors='dimgray')

        plt.clabel(C, fontsize=12, colors='black', inline=False)

    # Plot and set contourf

    if contourf != []:

        plt.contourf(X, Y, contourf, cmap='jet')

        plt.colorbar()

    # Plot and set quiver

    if quiver\_u != [] and quiver\_v != []:

        quiver\_u = sparse(dataset=quiver\_u, num\_to\_one=num\_to\_one)

        quiver\_v = sparse(dataset=quiver\_v, num\_to\_one=num\_to\_one)

        plt.quiver(X, Y, quiver\_u, quiver\_v, scale=scale) #, scale=300

    # Others set

    plt.title(title, fontsize='large')

# plot 200hPa Vg and HGT

def plot\_Vg\_HGT(HGT, U, V):

    Vg\_u, Vg\_v = np.zeros\_like(HGT), np.zeros\_like(HGT)

    dy = 6378000\*delta\*math.pi/180

    for y in range(y\_num):

        # renew dx and f per y (f=2\*omega\*sin(lat))

        dx = dy\*math.cos(y\_to\_lat(y)\*(math.pi/180))

        f = 2\*(2\*math.pi/(24\*60\*60))\*math.sin(y\_to\_lat(y)\*(math.pi/180))

        for x in range(x\_num):

            # default try: median\_interpolation

            try:

                dHGTdy = median\_interpolation(HGT[y-1, x], HGT[y+1, x], dy)

                dHGTdx = median\_interpolation(HGT[y, x-1], HGT[y, x+1], dx)

            except IndexError:

                pass

            # edge conditions

            if y == 0:dHGTdy = behind\_interpolation(HGT[y, x], HGT[y+1, x], dy)

            if y == y\_num-1:dHGTdy = front\_interpolation(HGT[y-1, x], HGT[y, x], dy)

            if x == 0:dHGTdx = behind\_interpolation(HGT[y, x], HGT[y, x+1], dx)

            if x == x\_num-1:dHGTdx = front\_interpolation(HGT[y, x-1], HGT[y, x], dx)

            # calculate Vg

            # isopressure formula: Vg=k x gradient(HGT) \*g /f (k x = Counterclockwise 90 degree from gradient)

            # Four quadrant

            if dHGTdx > 0 and dHGTdy > 0:move=[-1, 1]

            elif dHGTdx < 0 and dHGTdy > 0:move=[-1, -1]

            elif dHGTdx < 0 and dHGTdy < 0:move=[1, -1]

            elif dHGTdx > 0 and dHGTdy < 0:move=[1, 1]

            # if dHGTdx/dHGTdy == 0

            elif dHGTdx == 0 and dHGTdy > 0:move=[-1, 0]

            elif dHGTdx == 0 and dHGTdy < 0:move=[1, 0]

            elif dHGTdy == 0 and dHGTdx > 0:move=[0, 1]

            elif dHGTdy == 0 and dHGTdx < 0:move=[0, -1]

            else:print('zero error!!')

            Vg\_u[y, x] = move[0] \* abs(dHGTdy) \*g /f

            Vg\_v[y, x] = move[1] \* abs(dHGTdx) \*g /f

        #print('last dHGTdy, f, Vg\_u', dHGTdy, f, Vg\_u[y, x])

    plot\_on\_map('200hPa Geostrophic Wind', contour=HGT, contourf=HGT, quiver\_u=Vg\_u, quiver\_v=Vg\_v, num\_to\_one=2, scale=2000)

    plt.savefig(hw3\_root\_path+'200hPa Geostrophic Wind'+'.png', dpi=600)

    plt.show()

# plot 200hPa Vag and divergence and HGT

def plot\_Vag\_divergence\_HGT(HGT, U, V):

    Vag\_u, Vag\_v, divergence = np.zeros\_like(HGT), np.zeros\_like(HGT), np.zeros\_like(HGT)

    dy = 6378000\*delta\*math.pi/180

    for y in range(y\_num):

        # renew dx and f per y (f=2\*omega\*sin(lat))

        dx = dy\*math.cos(y\_to\_lat(y)\*(math.pi/180))

        f = 2\*(2\*math.pi/(24\*60\*60))\*math.sin(y\_to\_lat(y)\*(math.pi/180))

        for x in range(x\_num):

            # default try: median\_interpolation

            try:

                dHGTdy = median\_interpolation(HGT[y-1, x], HGT[y+1, x], dy)

                dHGTdx = median\_interpolation(HGT[y, x-1], HGT[y, x+1], dx)

                dvdy = median\_interpolation(V[y-1, x], V[y+1, x], dy)

                dudx = median\_interpolation(U[y, x-1], U[y, x+1], dx)

            except IndexError:

                pass

            # edge conditions

            if y == 0:

                dHGTdy = behind\_interpolation(HGT[y, x], HGT[y+1, x], dy)

                dvdy = behind\_interpolation(V[y, x], V[y+1, x], dy)

            if y == y\_num-1:

                dHGTdy = front\_interpolation(HGT[y-1, x], HGT[y, x], dy)

                dvdy = front\_interpolation(V[y-1, x], V[y, x], dy)

            if x == 0:

                dHGTdx = behind\_interpolation(HGT[y, x], HGT[y, x+1], dx)

                dudx = behind\_interpolation(U[y, x], U[y, x+1], dx)

            if x == x\_num-1:

                dHGTdx = front\_interpolation(HGT[y, x-1], HGT[y, x], dx)

                dudx = front\_interpolation(U[y, x-1], U[y, x], dx)

            # calculate divergence

            divergence[y, x] = dudx + dvdy

            # calculate Vg and Vag(V-Vg)

            # isopressure formula: Vg=k x gradient(HGT) \*g /f (k x = Counterclockwise 90 degree from gradient)

            # Four quadrant

            if dHGTdx > 0 and dHGTdy > 0:move=[-1, 1]

            elif dHGTdx < 0 and dHGTdy > 0:move=[-1, -1]

            elif dHGTdx < 0 and dHGTdy < 0:move=[1, -1]

            elif dHGTdx > 0 and dHGTdy < 0:move=[1, 1]

            # if dHGTdx/dHGTdy == 0

            elif dHGTdx == 0 and dHGTdy > 0:move=[-1, 0]

            elif dHGTdx == 0 and dHGTdy < 0:move=[1, 0]

            elif dHGTdy == 0 and dHGTdx > 0:move=[0, 1]

            elif dHGTdy == 0 and dHGTdx < 0:move=[0, -1]

            else:print('zero error!!')

            Vag\_u[y, x] = U[y, x] - (move[0] \* abs(dHGTdy) \*g /f)

            Vag\_v[y, x] = V[y, x] - (move[1] \* abs(dHGTdx) \*g /f)

        #print('last U Ug Uag:', U[y, x], (move[0] \* abs(dHGTdy) / f), Vag\_u[y, x])

    plot\_on\_map('200hPa Ageostrophic Wind', contour=HGT, contourf=divergence, quiver\_u=Vag\_u, quiver\_v=Vag\_v, num\_to\_one=2, scale=1000)

    plt.savefig(hw3\_root\_path+'200hPa Ageostrophic Wind'+'.png', dpi=600)

    plt.show()

# Plot 500hPa HGT and wind, add trough+ridge line

def plot\_500(HGT500, U500, V500):

    plot\_on\_map('500hPa Wind and HGT and trough+ridge', contour=HGT500, contourf=HGT500, quiver\_u=U500, quiver\_v=V500, num\_to\_one=2, scale=1000)

    plt.plot([125, 113], [50, 20], 'r-')

    plt.plot([140, 147], [50, 30], 'b-')

    plt.savefig(hw3\_root\_path+'500hPa Wind and HGT and trough+ridge'+'.png', dpi=600)

    plt.show()

# Plot 1000hPa HGT and wind, add Ground System(H, L and front)

def plot\_1000(HGT1000, U1000, V1000):

    plot\_on\_map('1000hPa Wind and HGT and Ground System', contour=HGT1000, contourf=HGT1000, quiver\_u=U1000, quiver\_v=V1000, num\_to\_one=1, scale=1000)

    plt.text(85, 47, 'H', fontsize=30)

    plt.text(100, 42, 'H', fontsize=30)

    plt.text(110, 35, 'H', fontsize=30)

    plt.text(145, 35, 'L', fontsize=30)

    plt.text(168, 58, 'L', fontsize=30)

    plt.plot([128, 130], [40, 37], 'r-')

    plt.plot([130, 133], [37, 35], 'r-')

    plt.plot([133, 141], [35, 38], 'r-')

    plt.plot([138, 152], [23, 36], 'r-')

    plt.plot([152, 157], [36, 39], 'r-')

    plt.plot([157, 162], [39, 39], 'r-')

    plt.xlabel('The red line is the front')

    plt.savefig(hw3\_root\_path+'1000hPa Wind and HGT and Ground System'+'.png', dpi=600)

    plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

    # Definition of basic parameters

    print('init!!')

    names = locals()

    hw3\_root\_path = os.path.join(os.path.abspath('.'), 'hw3')+'\\'

    pressure\_list = ['200', '500', '1000']

    parameter\_list = ['HGT', 'U', 'V']

    x\_num, y\_num = 91, 46

    lon\_upper, lon\_lower, lat\_upper, lat\_lower = 170, 80, 65, 20

    delta = 1.0

    g = 9.8

    # Load HGT, u, v data + definate variable, check plane\_data.shape

    for parameter in parameter\_list:

        for pressure in pressure\_list:

            plane\_data\_name = parameter + pressure

            plane\_data = np.reshape(read\_bindata\_return\_wanted(hw3\_root\_path, filename='fnldata.dat', pressure=pressure, parameter=parameter), (y\_num, x\_num))

            names[plane\_data\_name] = plane\_data

            print(parameter+pressure, 'load ok')

            if plane\_data.shape[0] != y\_num or plane\_data.shape[1] != x\_num:print('shape error!!')

    # Call function

    plot\_Vg\_HGT(HGT200, U200, V200)

    plot\_Vag\_divergence\_HGT(HGT200, U200, V200)

    plot\_500(HGT500, U500, V500)

    plot\_1000(HGT1000, U1000, V1000)