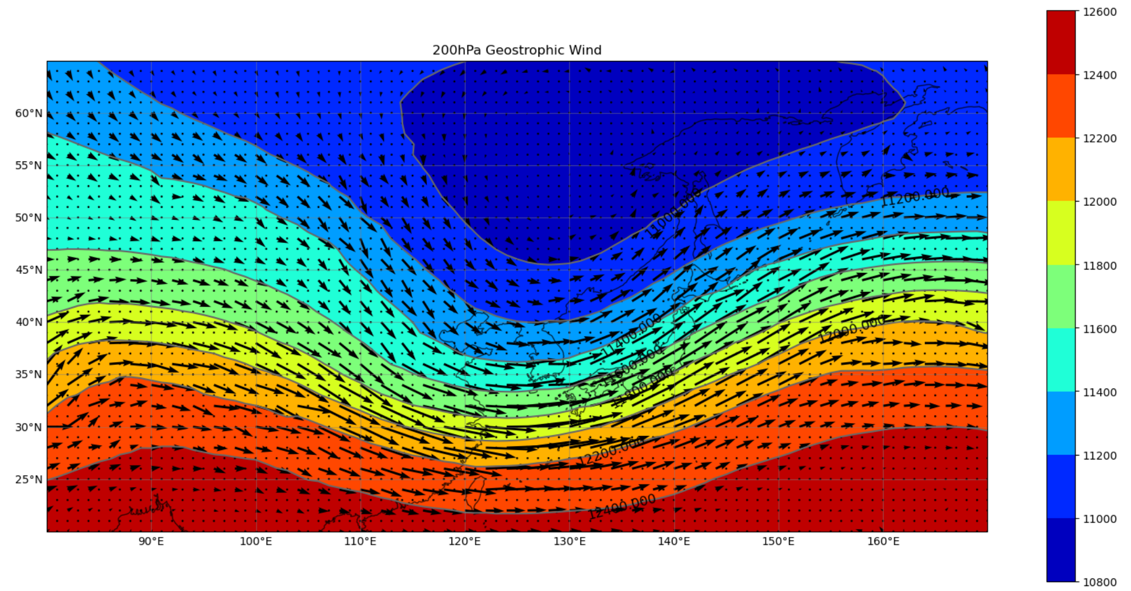
天氣學下 hw3　　大氣 4A　黃展皇　106601015

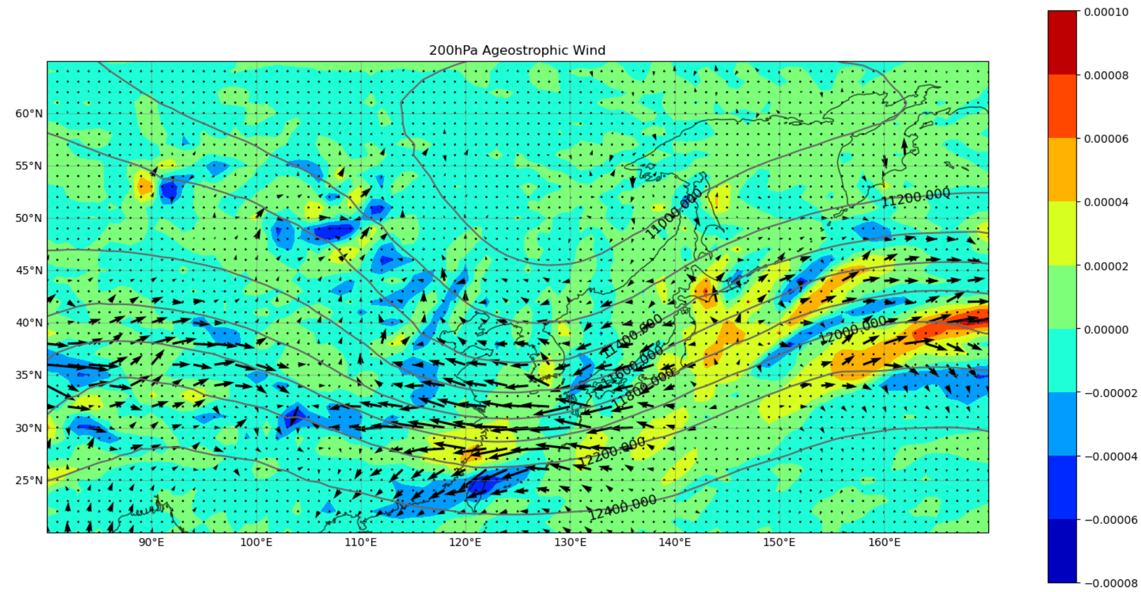
工作環境：x64 windows10，conda 4.8.3，python3.6.10

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

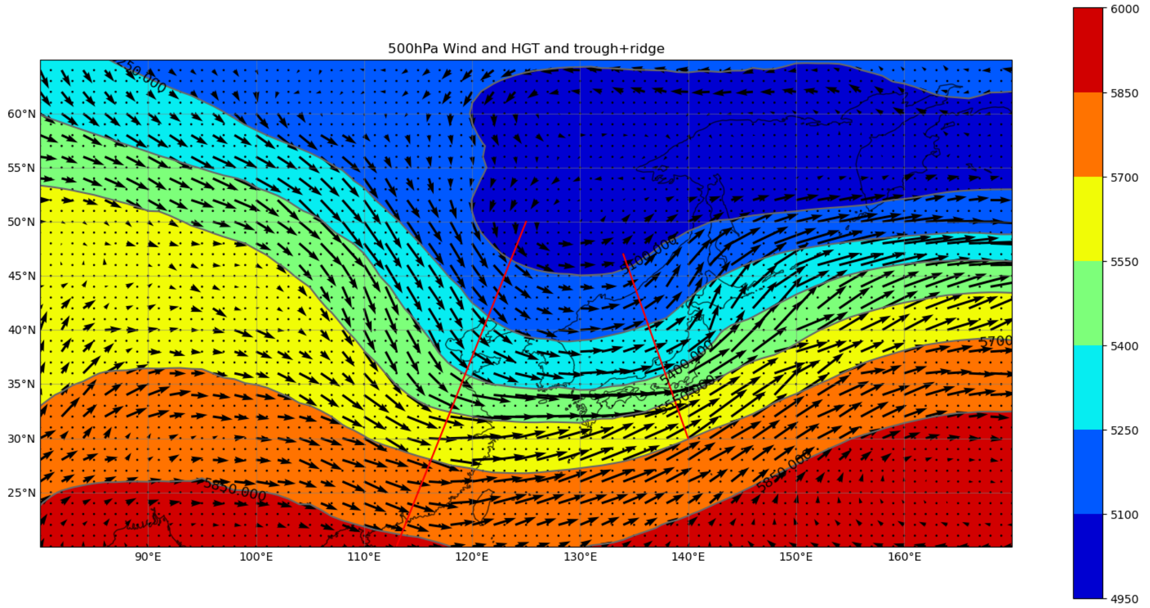
(一) 200 hPa 地轉風與重力位高度場



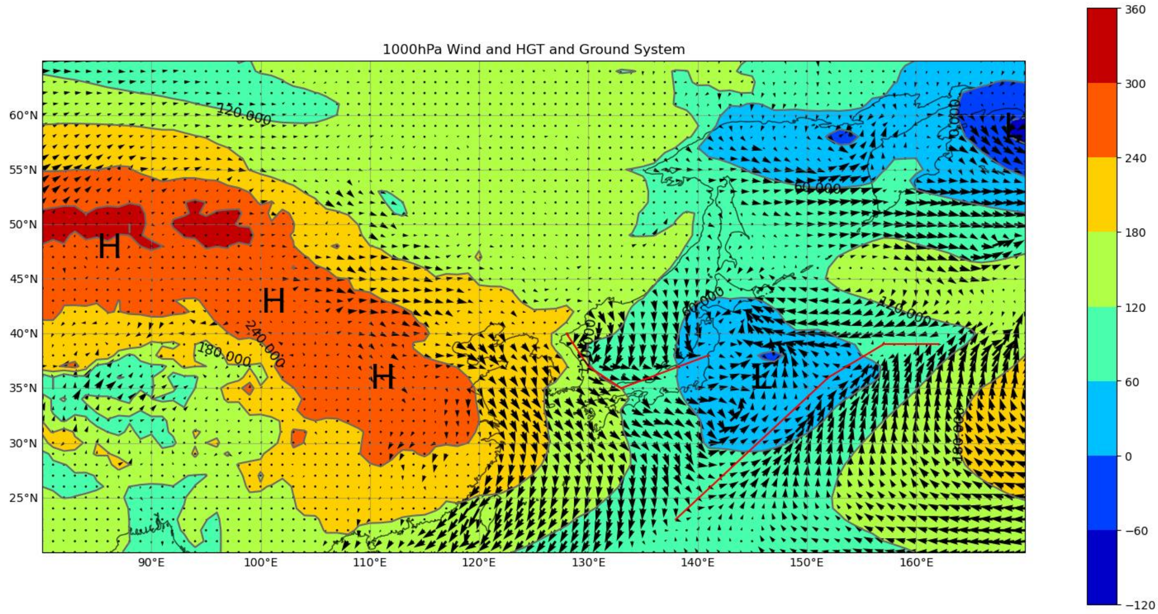
(二)200 hPa 非地轉風&輻合輻散與重力位高度場



(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_FORMATT
ER
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) = 376
74 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(pa
rameter) + 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
```

```python
# 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 r
ight 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+1
0, 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], d
y)
                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], d
x)
                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 = Co
unterclockwise 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=diver
gence, quiver_u=Vag_u, quiver_v=Vag_v, num_to_one=2, scale=1000)
    plt.savefig(hw3_root_path+'200hPa Ageostrophic Wind'+'.png', dpi=60
0)
    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=100
0)
    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=HGT10
00, contourf=HGT1000, quiver_u=U1000, quiver_v=V1000, num_to_one=1, sca
le=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)
```