数值天气预报课程作业

吴浩杰 20221170212

1 实习一

σ 和 P 坐标系的转换

1.1 实习目的

通过气象要素场在不同垂直坐标系间转换的学习，使学生掌握气象要素场在 σ 和 p 坐标系中相互转换的基本计算方法。

1.2 实习要求

编写并提交实现气象要素场在 σ 和 p 坐标系之间相互转换的 MATLAB、Python 或 NCL 程序。要求学生在机房现场操作，撰写实习报告，教师进行随堂讲解和指导。

1.3 实习内容

设大气上界 $p_T = 0$，编写程序，读入等压面层上的气象再分析数据，并把它转换为 σ 面上的值。

Listing 1: 坐标系的转换

```python
import numpy as np
import xarray as xr
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
from tqdm import tqdm

# 设置matplotlib中文字体
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 设置文件路径
```

```
surface_pressure_path = r'D:\zhuomian\shuzhi\shiyan1\ERA5_surface_pressure_201201.nc'
temperature_path = r'D:\zhuomian\shuzhi\shiyan1\ERA5_temperature_201201.nc'

# 读取数据
surface_pressure = xr.open_dataset(surface_pressure_path)['sp']
temperature = xr.open_dataset(temperature_path)['t']
pressure_levels = xr.open_dataset(temperature_path)['level'].values * 100  # 单位转换为 Pa，并转换为numpy数组
lons = xr.open_dataset(temperature_path)['longitude']
lats = xr.open_dataset(temperature_path)['latitude']

# 打印数据维度信息
print("Temperature shape:", temperature.shape)
print("Pressure levels shape:", pressure_levels.shape)
print("Surface pressure shape:", surface_pressure.shape)

# sigma_levels设定
sigma_levels = np.array([
    0.99500, 0.97999, 0.94995, 0.89988, 0.82977, 0.74468, 0.64954,
    0.54946, 0.45447, 0.36948, 0.29450, 0.22953, 0.17457, 0.12440,
    0.0846830, 0.0598005, 0.0449337, 0.0349146, 0.0248800, 0.00829901
])

nlev = len(sigma_levels)
nlat = len(lats)
nlon = len(lons)

# 初始化 面气温数组
temperature_sigma = np.zeros((nlev, nlat, nlon))

# 调整维度顺序，确保与pressure_levels匹配
temperature_data = temperature.values[0].transpose(1, 2, 0)  # 调整为(lat, lon, level)
surface_pressure_data = surface_pressure.values[0].transpose(0, 1)  # 获取第一个时间步的数据

# 插值到 面
```

```
47  print("正在从等压面插值到 面...")
48  for i in tqdm(range(nlon), desc='处理经度', ncols=100):
49      for j in range(nlat):
50          sp_value = surface_pressure_data[j, i]
51          sigma_plev = sp_value * sigma_levels
52          temp_profile = temperature_data[j, i, :]
53          if not np.any(np.isnan(temp_profile)):
54              interp_func = interp1d(pressure_levels,
                    temp_profile,
55                                     kind='cubic', bounds_error=
                                           False, fill_value=np.nan)
56              temperature_sigma[:, j, i] = interp_func(
                    sigma_plev)
57
58  # 定义等压面
59  p_levels = np.array
        ([925,875,825,775,725,675,625,575,550,475,425,375,325,275,225,175,125])
         * 100
60  nlev_p = len(p_levels)
61  temperature_pressure = np.zeros((nlev_p, nlat, nlon))
62
63  # 从 面插值回到等压面
64  print("\n正在从 面插值回到等压面...")
65  for i in tqdm(range(nlon), desc='处理经度', ncols=100):
66      for j in range(nlat):
67          pressure_siglev = p_levels / surface_pressure_data[j,
                i]
68          temp_profile = temperature_sigma[:, j, i]
69          if not np.any(np.isnan(temp_profile)):
70              interp_func = interp1d(sigma_levels, temp_profile,
71                                     kind='cubic', bounds_error=
                                           False, fill_value=np.nan)
72              temperature_pressure[:, j, i] = interp_func(
                    pressure_siglev)
73
74  # 绘制原始等压面825hPa的气温场
75  plt.figure(figsize=(12, 8))
76  ax0 = plt.axes(projection=ccrs.Robinson(central_longitude=181)
        )
77  ax0.coastlines()
78  cs0 = ax0.contour(lons, lats, temperature.sel(level=825).
        values[0],
79                    colors='k', levels=20, transform=ccrs.
```

```
                        PlateCarree())
ax0.clabel(cs0, inline=True, fontsize=7)
ax0.set_title('原始等压面825hPa气温场', fontsize=12)
plt.tight_layout()
plt.show()

# 绘制 面上第一层的气温场
plt.figure(figsize=(12, 8))
ax1 = plt.axes(projection=ccrs.Robinson(central_longitude=181))
ax1.coastlines()
cs1 = ax1.contour(lons, lats, temperature_sigma[0, :, :],
                  colors='k', levels=20, transform=ccrs.
                      PlateCarree())
ax1.clabel(cs1, inline=True, fontsize=7)
ax1.set_title(' 面第一层气温场', fontsize=12)
plt.tight_layout()
plt.show()

# 绘制插值回到825hPa的气温场
target_level_idx = np.where(p_levels == 82500)[0][0]
plt.figure(figsize=(12, 8))
ax2 = plt.axes(projection=ccrs.Robinson(central_longitude=181))
ax2.coastlines()
cs2 = ax2.contour(lons, lats, temperature_pressure[
    target_level_idx, :, :],
                  colors='k', levels=20, transform=ccrs.
                      PlateCarree())
ax2.clabel(cs2, inline=True, fontsize=7)
ax2.set_title('插值回到825hPa的气温场', fontsize=12)
plt.tight_layout()
plt.show()

# 计算插值前后的最大差值
original_temperature_at_825 = temperature.sel(level=825).
    values[0]
max_difference = np.nanmax(np.abs(temperature_pressure[
    target_level_idx, :, :] - original_temperature_at_825))

print(f"\n最大插值差值: {max_difference:.4f} K")
```

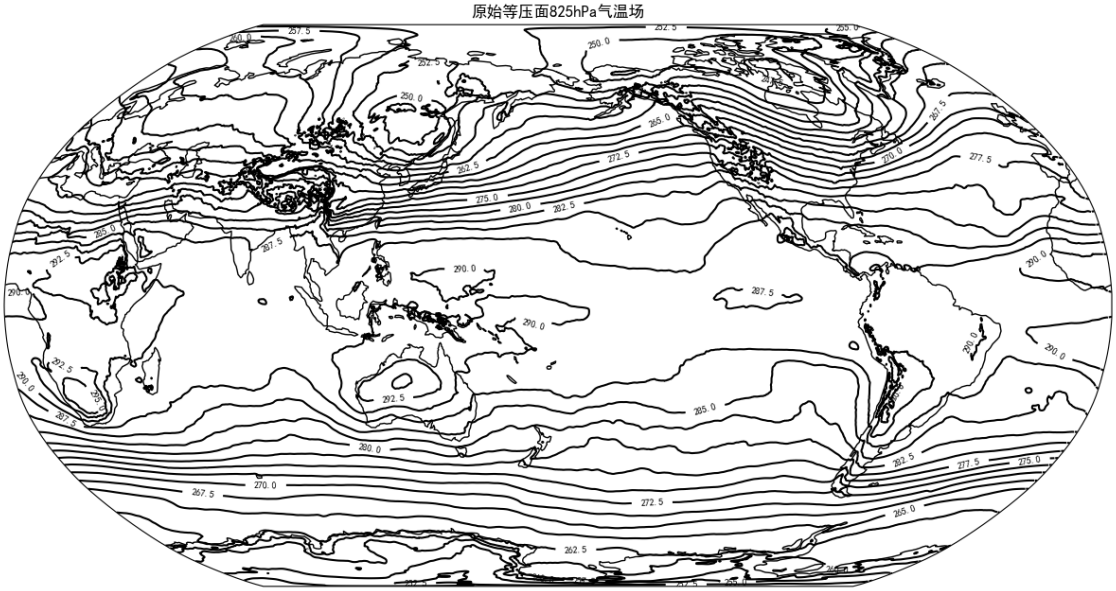

Figure 1: 原始等压面 825hPa 气温场

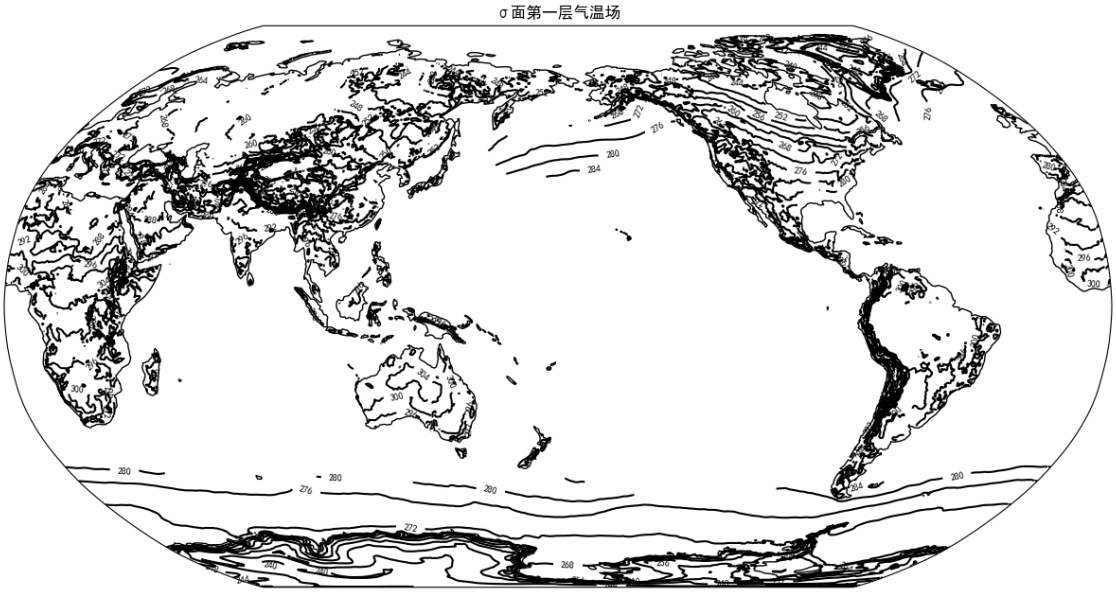

Figure 2: 面第一层气温场

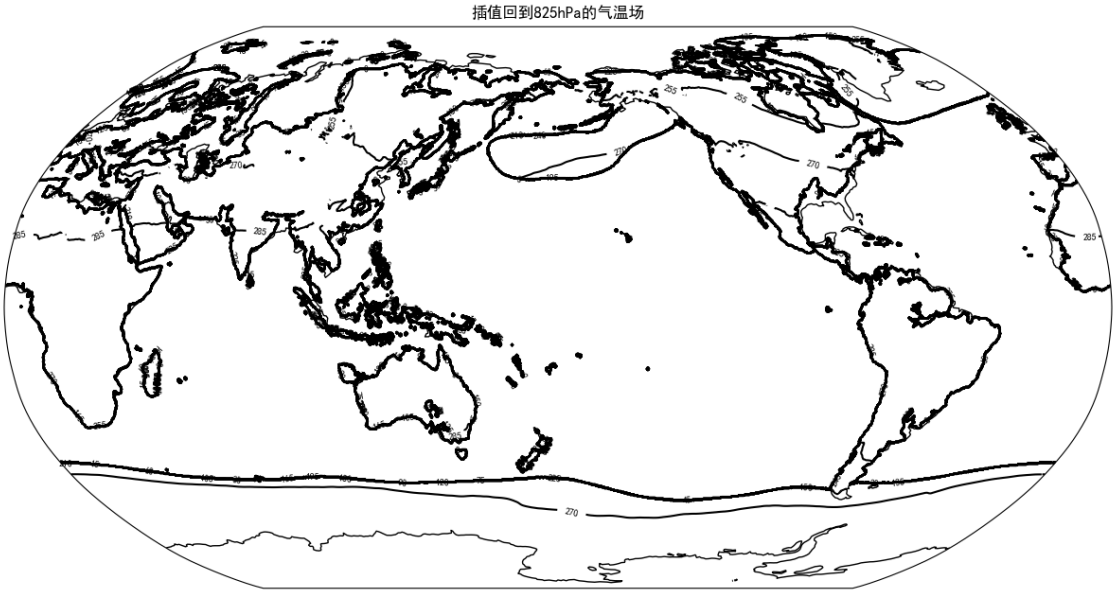

Figure 3: 插值回到 825hPa 的气温场

2 习题

2.1 习题一

根据示例程序，试对 2012 年 1 月平均 ERA5 重力位势高度场进行坐标系转换。

Listing 2: 重力位势高度场转换

```
import numpy as np
import xarray as xr
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
from scipy.interpolate import interp1d
from tqdm import tqdm

# 设置matplotlib中文字体
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# ========== 数据读取部分 ==========
# 设置文件路径
geo_path = r'D:\zhuomian\shuzhi\shiyan1\201201geo.nc'

```

```
    # 读取数据
    geo = xr.open_dataset(geo_path)['z']
    pressure_levels = xr.open_dataset(geo_path)['pressure_level'].values * 100  # 单位转换为 Pa
    lons = xr.open_dataset(geo_path)['longitude']
    lats = xr.open_dataset(geo_path)['latitude']

    # ====== 数据预处理部分 ======
    # 打印数据维度信息
    print("Geopotential height shape:", geo.shape)
    print("Pressure levels shape:", pressure_levels.shape)

    # 将重力位势转换为位势高度（单位：gpm）
    g = 9.80665  # 标准重力加速度，单位：m/s²
    geo_height = geo / g

    # 将经纬度数据转换为numpy数组
    lons_array = lons.values
    lats_array = lats.values

    # 打印经纬度信息
    print("\n经纬度信息:")
    print(f"经度范围: {lons_array.min():.2f}°E - {lons_array.max():.2f}°E")
    print(f"经度分辨率: {lons_array[1] - lons_array[0]:.2f}°")
    print(f"纬度范围: {lats_array.min():.2f}°N - {lats_array.max():.2f}°N")
    print(f"纬度分辨率: {lats_array[1] - lats_array[0]:.2f}°")

    # ====== 坐标系转换部分 ======
    # 设置 面
    sigma_levels = np.array([
        0.99500, 0.97999, 0.94995, 0.89988, 0.82977, 0.74468, 0.64954,
        0.54946, 0.45447, 0.36948, 0.29450, 0.22953, 0.17457, 0.12440,
        0.0846830, 0.0598005, 0.0449337, 0.0349146, 0.0248800, 0.00829901
    ])

    # 计算维度信息
    nlev = len(sigma_levels)  #  层的数量
    nlat = len(lats_array)    # 纬度的维数
```

```
    nlon = len(lons_array)     # 经度的维数

    print("\n维度信息:")
    print(f" 层数量: {nlev}")
    print(f"纬度维数: {nlat}")
    print(f"经度维数: {nlon}")

    # 初始化 面上的位势高度场
    geo_sigma = np.zeros((nlev, nlat, nlon))
    print("\n 面位势高度场数组形状:", geo_sigma.shape)

    # ======= 可视化部分 =======

    # 设置等压面值
    p_levels = np.array([925, 875, 825, 775, 725, 675, 625, 575,
        550, 475, 425, 375, 325, 275, 225, 175, 125]) * 100  #
        单位: Pa

    # 初始化等压面上的位势高度场
    geo_pressure = np.zeros((len(p_levels), nlat, nlon))

    # 调整维度顺序, 确保与pressure_levels匹配
    geo_data = geo.values[0].transpose(1, 2, 0)  # 调整为(lat,
        lon, level)

    # 从等压面插值到 面
    print("\n正在从等压面插值到 面...")
    for i in tqdm(range(nlon), desc='处理经度', ncols=100):
        for j in range(nlat):
            temp_profile = geo_data[j, i, :]
            if not np.any(np.isnan(temp_profile)):
                interp_func = interp1d(pressure_levels,
                    temp_profile,
                                       kind='cubic', bounds_error=
                                           False, fill_value=np.nan)
                sigma_plev = pressure_levels[0] * sigma_levels
                    # 使用最低层气压作为参考
                geo_sigma[:, j, i] = interp_func(sigma_plev)

    # 从 面插值回到等压面
    print("\n正在从 面插值回到等压面...")
    for i in tqdm(range(nlon), desc='处理经度', ncols=100):
        for j in range(nlat):
```

```
90              pressure_siglev = p_levels / pressure_levels[0]  # 使用最低层气压作为参考
91              temp_profile = geo_sigma[:, j, i]
92              if not np.any(np.isnan(temp_profile)):
93                  interp_func = interp1d(sigma_levels, temp_profile,
94                                         kind='cubic', bounds_error=False, fill_value=np.nan)
95                  geo_pressure[:, j, i] = interp_func(pressure_siglev)
96
97      # 计算插值前后的最大差值
98      target_level_idx = np.where(p_levels == 82500)[0][0]
99      original_geo_at_825 = geo.sel(pressure_level=825).values[0]
100     max_difference = np.nanmax(np.abs(geo_pressure[target_level_idx, :, :] - original_geo_at_825))
101
102     print(f"\n最大插值差值: {max_difference:.4f} gpm")
103
104     # 绘制三个子图
105     # 绘制825hPa等压面的位势高度场
106     plt.figure(figsize=(12, 8))
107     ax = plt.axes(projection=ccrs.Robinson(central_longitude=181))
108     ax.coastlines()
109     cs = ax.contour(lons, lats, geo_height.sel(pressure_level=825).values[0],
110                     colors='k', levels=20, transform=ccrs.PlateCarree())
111     ax.clabel(cs, inline=True, fontsize=7)
112     ax.set_title('825hPa等压面位势高度场', fontsize=12)
113     plt.tight_layout()
114     plt.show()
115
116     # 绘制 面第一层的位势高度场
117     plt.figure(figsize=(12, 8))
118     ax = plt.axes(projection=ccrs.Robinson(central_longitude=181))
119     ax.coastlines()
120     cs = ax.contour(lons, lats, geo_sigma[0, :, :] / g,  # 将重力位势转换为位势高度
121                     colors='k', levels=20, transform=ccrs.PlateCarree())
```

```
122     ax.clabel(cs, inline=True, fontsize=7)
123     ax.set_title('面第一层位势高度场', fontsize=12)
124     plt.tight_layout()
125     plt.show()
```

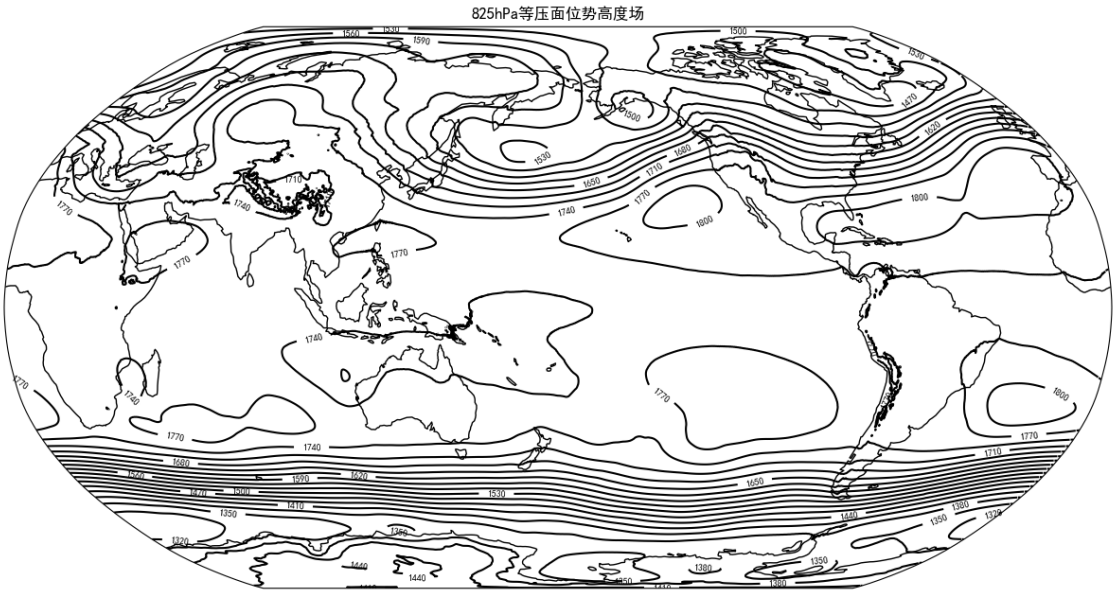

Figure 4: 原始等压面 825hPa 高度场

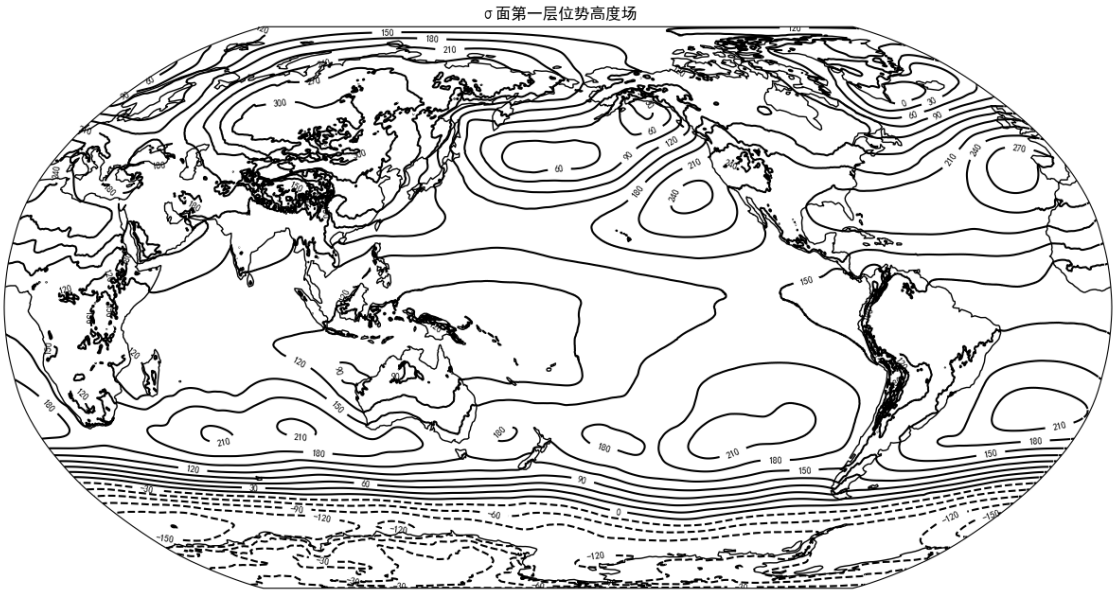

Figure 5: σ 面第一层位势高度场

2.2 习题二

德国马普气象研究院研发的大气环流模式 ECHAM6 采用地形追随混合坐标系（$\sigma - p$ 坐标系）。垂直坐标是基于气压的地形追随垂直坐标 $\eta(p, p_s)$, 它是气压 p 的单调函数，同时又依赖于表面气压 p_s。

$$\eta(0, p_s) = 0, \eta(p_s, p_s) = 1 \tag{1}$$

模式“半层”值的计算公式为

$$p_{k+1/2} = A_{k+1/2} + B_{k+1/2} p_s \tag{2}$$

其中，$k = 0, 1, 2, ..., NLEV(NLEV$ 表示模式在垂直方向上的总层数) $A_{k+1/2}$ 和 $B_{k+1/2}$ 为常数，p_s 表示地表气压。模式垂直层的气压为相邻半层值的平均。

$$p_k = \frac{1}{2}\left(p_{k+1/2} + p_{k-1/2}\right) \tag{3}$$

模式层 $\eta_{k+1/2}$ 的计算公式为

$$\eta_{k+1/2} = \frac{A_{k+1/2}}{p_0} + B_{k+1/2} \tag{4}$$

其中，P_0 是海平面气压的参考值 101325Pa，采用线性插值方法计算 η。

$$\eta = \eta_{k+1/2} + \frac{\left(p - p_{k+1/2}\right)\left(\eta_{k+1/2} - \eta_{k-1/2}\right)}{p_{k+1/2} - p_{k-1/2}} \tag{5}$$

其中，$p_{k-1/2} < p < p_{k+1/2}$ 根据计算模式垂直层的程序，尝试把 2012 年 1 月平均 ERA5 气温场从 p 坐标系转换到 ECHAM6 的 $\sigma - p$ 坐标系。

Listing 3: 混合坐标系

```
import numpy as np
import xarray as xr
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
from scipy.interpolate import interp1d
from tqdm import tqdm

# 设置matplotlib中文字体
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 设置文件路径
temperature_path = r'D:\zhuomian\shuzhi\shiyan1\
    ERA5_temperature_201201.nc'
surface_pressure_path = r'D:\zhuomian\shuzhi\shiyan1\
    ERA5_surface_pressure_201201.nc'
```

```
15
16      # 读取数据
17      temperature = xr.open_dataset(temperature_path)['t']
18      surface_pressure = xr.open_dataset(surface_pressure_path)['
            sp']
19      pressure_levels = xr.open_dataset(temperature_path)['level'
            ].values * 100  # 单位转换为 Pa
20      lons = xr.open_dataset(temperature_path)['longitude']
21      lats = xr.open_dataset(temperature_path)['latitude']
22
23      # 打印数据维度信息
24      print("Temperature shape:", temperature.shape)
25      print("Pressure levels shape:", pressure_levels.shape)
26      print("Surface pressure shape:", surface_pressure.shape)
27
28      # 定义混合坐标系参数（Ak+1/2和Bk+1/2）
29      A = np.array([
30          0.0, 2000.0, 4000.0, 6000.0, 8000.0, 9976.135, 11820.54,
31          13431.4, 14736.36, 15689.21, 16266.61, 16465.0,
                16297.62,
32          15791.6, 14985.27, 13925.52, 12665.29, 11261.23,
                9771.406,
33          8253.211, 6761.34, 5345.914, 4050.718, 2911.57,
                1954.805,
34          1195.89, 638.1489, 271.6265, 72.06358, 0.0, 0.0, 0.0
35      ])
36
37      B = np.array([
38          0.0, 0.0, 0.0, 0.0, 0.0, 0.000391, 0.002920, 0.009194,
39          0.020319, 0.036975, 0.059488, 0.087895, 0.122004,
                0.161442,
40          0.205703, 0.254189, 0.306235, 0.361145, 0.418202,
                0.476688,
41          0.535887, 0.595084, 0.653565, 0.710594, 0.765405,
                0.817167,
42          0.864956, 0.907716, 0.944213, 0.972985, 0.992282, 1.0
43      ])
44
45      # 计算维度信息
46      nlev = len(A) - 1  # 垂直层数
47      nlat = len(lats)   # 纬度的维数
48      nlon = len(lons)   # 经度的维数
49
```

```
    print("\n维度信息:")
    print(f"垂直层数: {nlev}")
    print(f"纬度维数: {nlat}")
    print(f"经度维数: {nlon}")

    # 初始化混合坐标系上的温度场
    temperature_hybrid = np.zeros((nlev, nlat, nlon))

    # 调整维度顺序，确保与pressure_levels匹配
    temperature_data = temperature.values[0].transpose(1, 2, 0)
        # 调整为(lat, lon, level)
    surface_pressure_data = surface_pressure.values[0].transpose
        (0, 1)  # 获取第一个时间步的数据

    # 计算半层气压值和混合坐标系上的温度场
    print("\n正在计算半层气压值和混合坐标系上的温度场...")
    for i in tqdm(range(nlon), desc='处理经度', ncols=100):
        for j in range(nlat):
            # 计算每个格点的半层气压值
            pk_half = np.zeros(nlev + 1)
            for k in range(nlev + 1):
                pk_half[k] = A[k] + B[k] * surface_pressure_data
                    [j, i]

            # 计算全层气压值
            pk = 0.5 * (pk_half[1:] + pk_half[:-1])

            # 计算混合坐标系的 值
            eta = np.zeros(nlev)
            for k in range(nlev):
                eta[k] = A[k] / surface_pressure_data[j, i] + B[
                    k]

            # 使用线性插值将温度从等压面插值到混合坐标系
            temp_profile = temperature_data[j, i, :]
            if not np.any(np.isnan(temp_profile)):
                interp_func = interp1d(pressure_levels,
                    temp_profile,
                                       kind='cubic', bounds_error=
                                           False, fill_value=np.nan)
                temperature_hybrid[:, j, i] = interp_func(pk)

    # 绘制原始等压面825hPa的温度场
```

```
    plt.figure(figsize=(12, 8))
    ax0 = plt.axes(projection=ccrs.Robinson(central_longitude
        =181))
    ax0.coastlines()
    cs0 = ax0.contour(lons, lats, temperature.sel(level=825).
        values[0],
                      colors='k', levels=20, transform=ccrs.
                          PlateCarree())
    ax0.clabel(cs0, inline=True, fontsize=7)
    ax0.set_title('原始等压面825hPa温度场', fontsize=12)
    plt.tight_layout()
    plt.show()

    # 绘制混合坐标系第15层的温度场
    plt.figure(figsize=(12, 8))
    ax1 = plt.axes(projection=ccrs.Robinson(central_longitude
        =181))
    ax1.coastlines()
    cs1 = ax1.contour(lons, lats, temperature_hybrid[15, :, :],
                      colors='k', levels=20, transform=ccrs.
                          PlateCarree())
    ax1.clabel(cs1, inline=True, fontsize=7)
    ax1.set_title('混合坐标系第15层温度场', fontsize=12)
    plt.tight_layout()
    plt.show()

    # 计算并打印第15层的 值
    eta_15 = A[15] / surface_pressure_data[0, 0] + B[15]
    print(f"\n第15层的 值: {eta_15:.6f}")
```

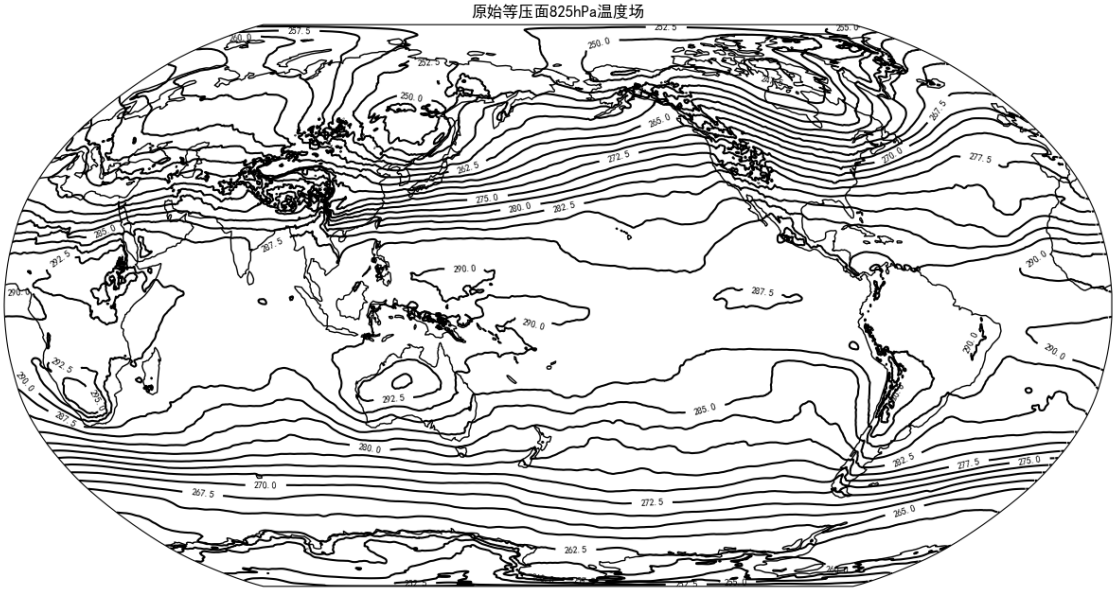

Figure 6: 原始等压面 825hPa 气温场

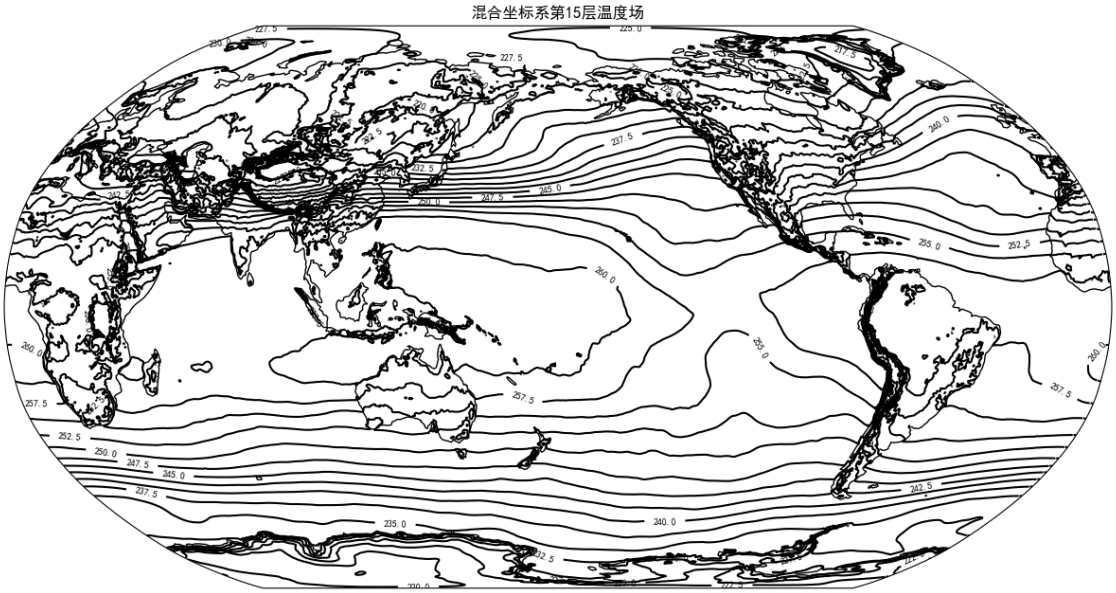

Figure 7: 混合坐标系第一层温度场