数值天气预报课程作业

吴浩杰 20221170212

1 实习三

麦卡托投影的地图放大系数和科里奥利参数

1.1 实习目的

通过编程计算，使学生掌握麦卡托投影的地图放大系数和科里奥利参数的计算方法。

1.2 实习要求

编写并提交计算麦卡托投影的地图放大系数和科里奥利参数的相关程序。要求学生在机房现场操作，参考沈桐立等 (2015) 关于麦卡托投影相关内容，撰写实习报告，教师进行随堂讲解和指导。

1.3 实习内容

根据实际计算麦卡托投影地图放大系数的公式，输入 a、d、Ω 和格点坐标，计算地图放大系数和科里奥利参数。

Listing 1: 麦卡托投影地图放大系数

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 设置matplotlib中文字体支持
plt.rcParams['font.sans-serif'] = ['SimHei']  # 设置默认字体为黑体
plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题

# 初始化参数
d = 100
```

```
    phy0 = 5
    a = 6371
    omega = 7.292e-5
    l_ref = a * np.cos(np.deg2rad(22.5)) * np.tan(np.deg2rad(
        phy0))

    # 定义起始经纬度
    lon_start = 100  # 起始经度（°E）
    lat_start = 22.5  # 起始纬度（°N）

    # 初始化矩阵
    m = np.zeros((6, 7))
    f = np.zeros((6, 7))

    # 计算映射系数m和科氏力参数f
    for In in range(7):
        for Jn in range(6):
            # 计算当前点的纬度（考虑到矩阵翻转）
            current_lat = lat_start - (5-Jn) * (d / (a * np.pi /
                180))  # 5-Jn是因为矩阵会被翻转
            # 地图放大系数只与纬度有关
            m[Jn, In] = 1 / np.cos(np.deg2rad(current_lat))
            # 科氏力参数只与纬度有关
            f[Jn, In] = 2 * omega * np.sin(np.deg2rad(
                current_lat))

    # 翻转矩阵
    m = np.flipud(m)
    f = np.flipud(f)

    # 打印结果
    print("映射系数m:")
    print(m)
    print("\n科氏力参数f:")
    print(f)

    # 将结果保存到Excel文件
    df_m = pd.DataFrame(m)
    df_f = pd.DataFrame(f)

    df_m.to_excel('map_scale.xlsx', index=False, header=False)
    df_f.to_excel('coriolis_force.xlsx', index=False, header=
        False)
```

```
50
51      # 计算经纬度网格
52      lon_start = 100  # 起始经度（°E）
53      lat_start = 22.5  # 起始纬度（°N）
54
55      # 计算每个网格点对应的经纬度
56      lon_grid = np.zeros((6, 7))
57      lat_grid = np.zeros((6, 7))
58
59      for i in range(7):
60          for j in range(6):
61              # 计算经度变化（根据距离和纬度计算）
62              delta_lon = (i * d) / (a * np.cos(np.deg2rad(
                    lat_start)) * np.pi / 180)
63              # 计算纬度变化（根据距离计算）
64              delta_lat = (j * d) / (a * np.pi / 180)
65
66              lon_grid[j, i] = lon_start + delta_lon
67              lat_grid[j, i] = lat_start - delta_lat  # 纬度向南递
                    减
68
69      # 翻转经纬度网格以匹配m和f矩阵
70      lon_grid = np.flipud(lon_grid)
71      lat_grid = np.flipud(lat_grid)
72
73      # 创建图形和子图
74      plt.figure(figsize=(15, 6))
75
76      # 绘制地图放大系数的热力图
77      plt.subplot(121)
78      im1 = plt.imshow(m, cmap='viridis')
79      plt.colorbar(im1, label='地图放大系数')
80      plt.title('地图放大系数分布')
81
82      # 设置经纬度刻度
83      xticks = np.arange(7)
84      yticks = np.arange(6)
85      plt.xticks(xticks, [f'{lon_grid[0,i]:.1f}°E' for i in range
            (7)])
86      plt.yticks(yticks, [f'{lat_grid[i,0]:.1f}°N' for i in range
            (6)])
87      plt.xlabel('经度')
88      plt.ylabel('纬度')
```

```
89
90      # 绘制科氏力参数的热力图
91      plt.subplot(122)
92      im2 = plt.imshow(f, cmap='viridis')
93      plt.colorbar(im2, label='科氏力参数')
94      plt.title('科氏力参数分布')
95
96      # 设置经纬度刻度
97      plt.xticks(xticks, [f'{lon_grid[0,i]:.1f}°E' for i in range
            (7)])
98      plt.yticks(yticks, [f'{lat_grid[i,0]:.1f}°N' for i in range
            (6)])
99      plt.xlabel('经度')
100     plt.ylabel('纬度')
101
102     # 调整子图之间的间距
103     plt.tight_layout()
104
105     # 显示图形
106     plt.show()
107
108     # 打印经纬度网格
109     print("\n经度网格 (°E):")
110     print(lon_grid)
111     print("\n纬度网格 (°N):")
112     print(lat_grid)
113
114     print("\n结果已保存到Excel文件中")
```

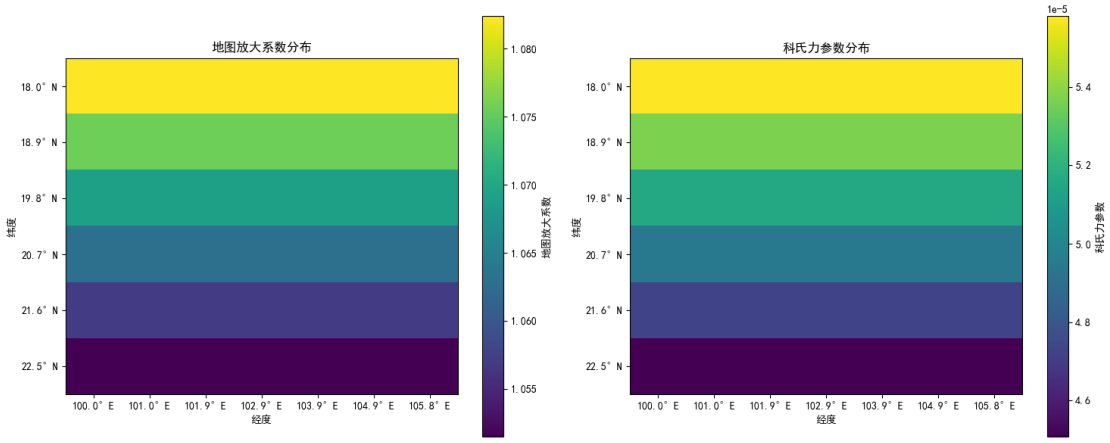

Figure 1: 参考点 O 为（100°E，5°N）

1.4 习题一

把麦卡托投影标准纬度变为 30°N，选取一矩形网格，左下角原点 O 为 (100°E，50°N)，网格距 100 km，网格点向北 6 个，向东 7 个。计算网格上各点的地图放大系数和科里奥利参数

Listing 2: 习题一

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 设置matplotlib中文字体支持
plt.rcParams['font.sans-serif'] = ['SimHei']  # 设置默认字体为黑体
plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题

# 初始化参数
d = 100  # 网格距离，单位km
phy0 = 30  # 标准纬度30°N
a = 6371  # 地球半径，单位km
omega = 7.292e-5  # 地球角速度

# 计算参考长度，使用原点位置(100°E，50°N)
l_ref = a * np.cos(np.deg2rad(50)) * np.tan(np.deg2rad(phy0))

# 定义起始经纬度
lon_start = 100  # 起始经度（°E）
lat_start = 50  # 起始纬度（°N）

# 初始化矩阵
m = np.zeros((6, 7))  # 地图放大系数矩阵
f = np.zeros((6, 7))  # 科氏力参数矩阵

# 计算映射系数m和科氏力参数f
for In in range(7):
    for Jn in range(6):
        # 计算当前点的纬度（考虑到矩阵翻转）
        current_lat = lat_start - (5-Jn) * (d / (a * np.pi / 180))  # 5-Jn是因为矩阵会被翻转
        # 地图放大系数只与纬度有关
        m[Jn, In] = 1 / np.cos(np.deg2rad(current_lat))
        # 科氏力参数只与纬度有关
```

```
            f[Jn, In] = 2 * omega * np.sin(np.deg2rad(
                current_lat))

    # 翻转矩阵使得左下角为原点
    m = np.flipud(m)
    f = np.flipud(f)

    # 打印结果
    print("地图放大系数m:")
    print(m)
    print("\n科氏力参数f:")
    print(f)

    # 将结果保存到Excel文件
    df_m = pd.DataFrame(m)
    df_f = pd.DataFrame(f)

    df_m.to_excel('map_scale_exp1.xlsx', index=False, header=
        False)
    df_f.to_excel('coriolis_force_exp1.xlsx', index=False,
        header=False)

    # 计算经纬度网格

    # 计算每个网格点对应的经纬度
    lon_grid = np.zeros((6, 7))
    lat_grid = np.zeros((6, 7))

    for i in range(7):
        for j in range(6):
            # 计算经度变化（根据距离和纬度计算）
            delta_lon = (i * d) / (a * np.cos(np.deg2rad(
                lat_start)) * np.pi / 180)
            # 计算纬度变化（根据距离计算）
            delta_lat = (j * d) / (a * np.pi / 180)

            lon_grid[j, i] = lon_start + delta_lon
            lat_grid[j, i] = lat_start - delta_lat  # 纬度向南递
                减

    # 翻转经纬度网格以匹配m和f矩阵
    lon_grid = np.flipud(lon_grid)
    lat_grid = np.flipud(lat_grid)
```

```
72
73      # 创建图形和子图
74      plt.figure(figsize=(15, 6))
75
76      # 绘制地图放大系数的热力图
77      plt.subplot(121)
78      im1 = plt.imshow(m, cmap='viridis')
79      plt.colorbar(im1, label='地图放大系数')
80      plt.title('地图放大系数分布')
81
82      # 设置经纬度刻度
83      xticks = np.arange(7)
84      yticks = np.arange(6)
85      plt.xticks(xticks, [f'{lon_grid[0,i]:.1f}°E' for i in range
            (7)])
86      plt.yticks(yticks, [f'{lat_grid[i,0]:.1f}°N' for i in range
            (6)])
87      plt.xlabel('经度')
88      plt.ylabel('纬度')
89
90      # 绘制科氏力参数的热力图
91      plt.subplot(122)
92      im2 = plt.imshow(f, cmap='viridis')
93      plt.colorbar(im2, label='科氏力参数')
94      plt.title('科氏力参数分布')
95
96      # 设置经纬度刻度
97      plt.xticks(xticks, [f'{lon_grid[0,i]:.1f}°E' for i in range
            (7)])
98      plt.yticks(yticks, [f'{lat_grid[i,0]:.1f}°N' for i in range
            (6)])
99      plt.xlabel('经度')
100     plt.ylabel('纬度')
101
102     # 调整子图之间的间距
103     plt.tight_layout()
104
105     # 显示图形
106     plt.show()
107
108     # 打印经纬度网格
109     print("\n经度网格 (°E):")
110     print(lon_grid)
```

```
    print("\n纬度网格 (°N):")
    print(lat_grid)

    print("\n结果已保存到Excel文件中")
```

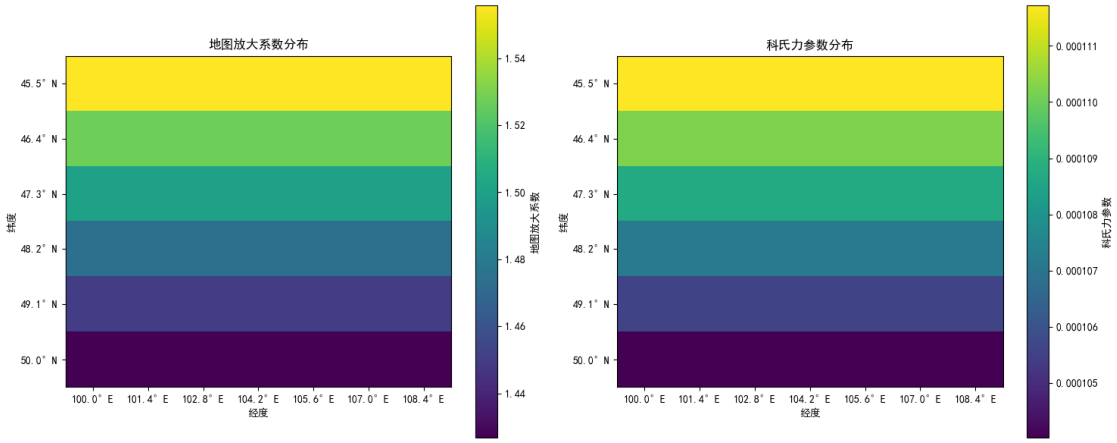

Figure 2: 中央经度为 45° 的麦卡托投影放大系数分布

1.5 习题二

中尺度天气预报模式 WRF 的麦卡托投影如书上所示，为减小模式网格上的投影形变，标准纬度可随投影范围改变。设模拟区域为中南半岛地区，尝试各种语言编写程序，计算麦卡托投影在东亚地区的地图放大系数和科里奥利参数，并与 WPS 相同设置的结果进行比较

Listing 3: 习题二

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 设置matplotlib中文字体支持
plt.rcParams['font.sans-serif'] = ['SimHei']  # 设置默认字体为黑体
plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题

# 初始化参数 - 中南半岛地区
d = 100  # 网格距离，单位km
phy0 = 15  # 标准纬度15°N（适合中南半岛地区）
a = 6371  # 地球半径，单位km
omega = 7.292e-5  # 地球角速度

# 计算参考长度，使用原点位置(100°E，20°N) - 中南半岛区域
```

```
    lon_start = 100  # 起始经度（°E）
    lat_start = 20   # 起始纬度（°N）
    l_ref = a * np.cos(np.deg2rad(lat_start)) * np.tan(np.
        deg2rad(phy0))

    # 初始化矩阵
    m = np.zeros((6, 7))  # 地图放大系数矩阵
    f = np.zeros((6, 7))  # 科氏力参数矩阵

    # 计算映射系数m和科氏力参数f
    for In in range(7):
        for Jn in range(6):
            # 计算当前点的纬度（考虑到矩阵翻转）
            current_lat = lat_start - (5-Jn) * (d / (a * np.pi /
                180))  # 5-Jn是因为矩阵会被翻转
            # 地图放大系数只与纬度有关
            m[Jn, In] = 1 / np.cos(np.deg2rad(current_lat))
            # 科氏力参数只与纬度有关
            f[Jn, In] = 2 * omega * np.sin(np.deg2rad(
                current_lat))

    # 翻转矩阵使得左下角为原点
    m = np.flipud(m)
    f = np.flipud(f)

    # 打印结果
    print("地图放大系数m（麦卡托投影，中南半岛区域）:")
    print(m)
    print("\n科氏力参数f（麦卡托投影，中南半岛区域）:")
    print(f)

    # 将结果保存到Excel文件
    df_m = pd.DataFrame(m)
    df_f = pd.DataFrame(f)

    df_m.to_excel('map_scale_exp2.xlsx', index=False, header=
        False)
    df_f.to_excel('coriolis_force_exp2.xlsx', index=False,
        header=False)

    # 计算每个网格点对应的经纬度
    lon_grid = np.zeros((6, 7))
    lat_grid = np.zeros((6, 7))
```

```
54
55      for i in range(7):
56          for j in range(6):
57              # 计算经度变化（根据距离和纬度计算）
58              delta_lon = (i * d) / (a * np.cos(np.deg2rad(
                    lat_start)) * np.pi / 180)
59              # 计算纬度变化（根据距离计算）
60              delta_lat = (j * d) / (a * np.pi / 180)
61
62              lon_grid[j, i] = lon_start + delta_lon
63              lat_grid[j, i] = lat_start - delta_lat  # 纬度向南递
                    减
64
65      # 翻转经纬度网格以匹配m和f矩阵
66      lon_grid = np.flipud(lon_grid)
67      lat_grid = np.flipud(lat_grid)
68
69      # 创建图形和子图
70      plt.figure(figsize=(15, 12))
71
72      # 绘制地图放大系数的热力图
73      plt.subplot(221)
74      im1 = plt.imshow(m, cmap='viridis')
75      plt.colorbar(im1, label='地图放大系数')
76      plt.title('麦卡托投影地图放大系数分布 (中南半岛区域)')
77
78      # 设置经纬度刻度
79      xticks = np.arange(7)
80      yticks = np.arange(6)
81      plt.xticks(xticks, [f'{lon_grid[0,i]:.1f}°E' for i in range
            (7)])
82      plt.yticks(yticks, [f'{lat_grid[i,0]:.1f}°N' for i in range
            (6)])
83      plt.xlabel('经度')
84      plt.ylabel('纬度')
85
86      # 绘制科氏力参数的热力图
87      plt.subplot(222)
88      im2 = plt.imshow(f, cmap='viridis')
89      plt.colorbar(im2, label='科氏力参数')
90      plt.title('麦卡托投影科氏力参数分布 (中南半岛区域)')
91
92      # 设置经纬度刻度
```

```
    plt.xticks(xticks, [f'{lon_grid[0,i]:.1f}°E' for i in range(7)])
    plt.yticks(yticks, [f'{lat_grid[i,0]:.1f}°N' for i in range(6)])
    plt.xlabel('经度')
    plt.ylabel('纬度')

    # 计算WPS设置下的地图放大系数和科氏力参数
    # 使用c.m文件中的公式进行计算
    m_wps = np.zeros((6, 7))
    f_wps = np.zeros((6, 7))

    for In in range(7):
        for Jn in range(6):
            # 根据c.m文件中的公式计算
            m_wps[Jn, In] = np.sqrt((a * np.cos(np.deg2rad(lat_start)))**2 + (Jn * d)**2) / a
            f_wps[Jn, In] = 2 * omega * np.sin(Jn * d / np.sqrt((a * np.cos(np.deg2rad(lat_start)))**2 + (Jn * d)**2))

    # 翻转矩阵
    m_wps = np.flipud(m_wps)
    f_wps = np.flipud(f_wps)

    # 绘制WPS设置下的地图放大系数热力图
    plt.subplot(223)
    im3 = plt.imshow(m_wps, cmap='viridis')
    plt.colorbar(im3, label='地图放大系数 (WPS)')
    plt.title('WPS设置下的地图放大系数分布')

    # 设置经纬度刻度
    plt.xticks(xticks, [f'{lon_grid[0,i]:.1f}°E' for i in range(7)])
    plt.yticks(yticks, [f'{lat_grid[i,0]:.1f}°N' for i in range(6)])
    plt.xlabel('经度')
    plt.ylabel('纬度')

    # 绘制WPS设置下的科氏力参数热力图
    plt.subplot(224)
    im4 = plt.imshow(f_wps, cmap='viridis')
    plt.colorbar(im4, label='科氏力参数 (WPS)')
```

```
    plt.title('WPS设置下的科氏力参数分布')

    # 设置经纬度刻度
    plt.xticks(xticks, [f'{lon_grid[0,i]:.1f}°E' for i in range(7)])
    plt.yticks(yticks, [f'{lat_grid[i,0]:.1f}°N' for i in range(6)])
    plt.xlabel('经度')
    plt.ylabel('纬度')

    # 调整子图之间的间距
    plt.tight_layout()

    # 显示图形
    plt.show()

    # 计算差异
    m_diff = m - m_wps
    f_diff = f - f_wps

    # 打印差异结果
    print("\n地图放大系数差异 (自定义 - WPS):")
    print(m_diff)
    print("\n科氏力参数差异 (自定义 - WPS):")
    print(f_diff)

    # 打印经纬度网格
    print("\n经度网格 (°E):")
    print(lon_grid)
    print("\n纬度网格 (°N):")
    print(lat_grid)

    print("\n结果已保存到Excel文件中")
```

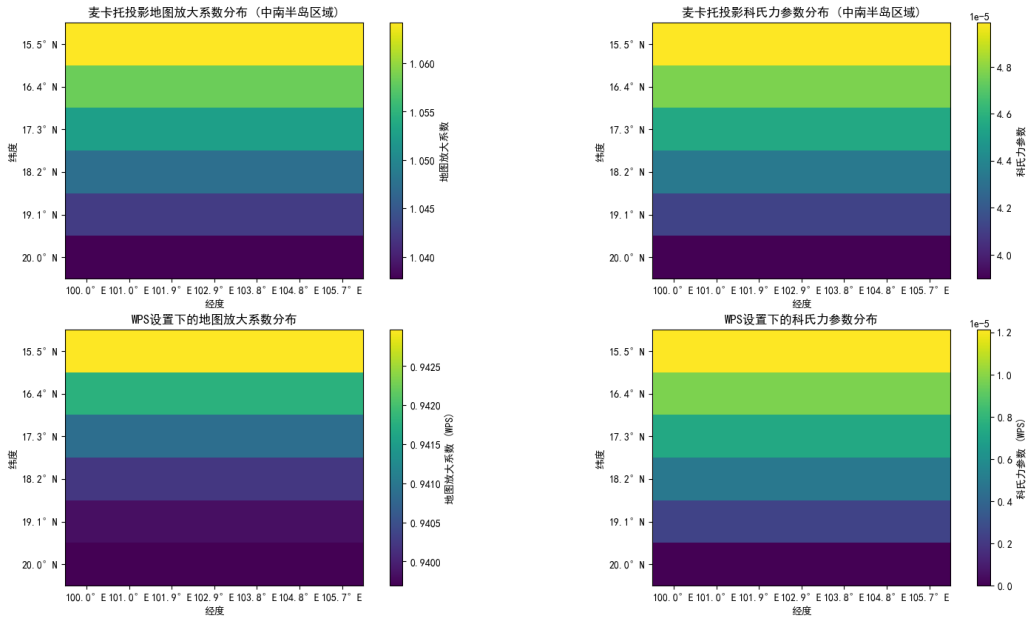

Figure 3: 对比分析