

5.3 数据插值

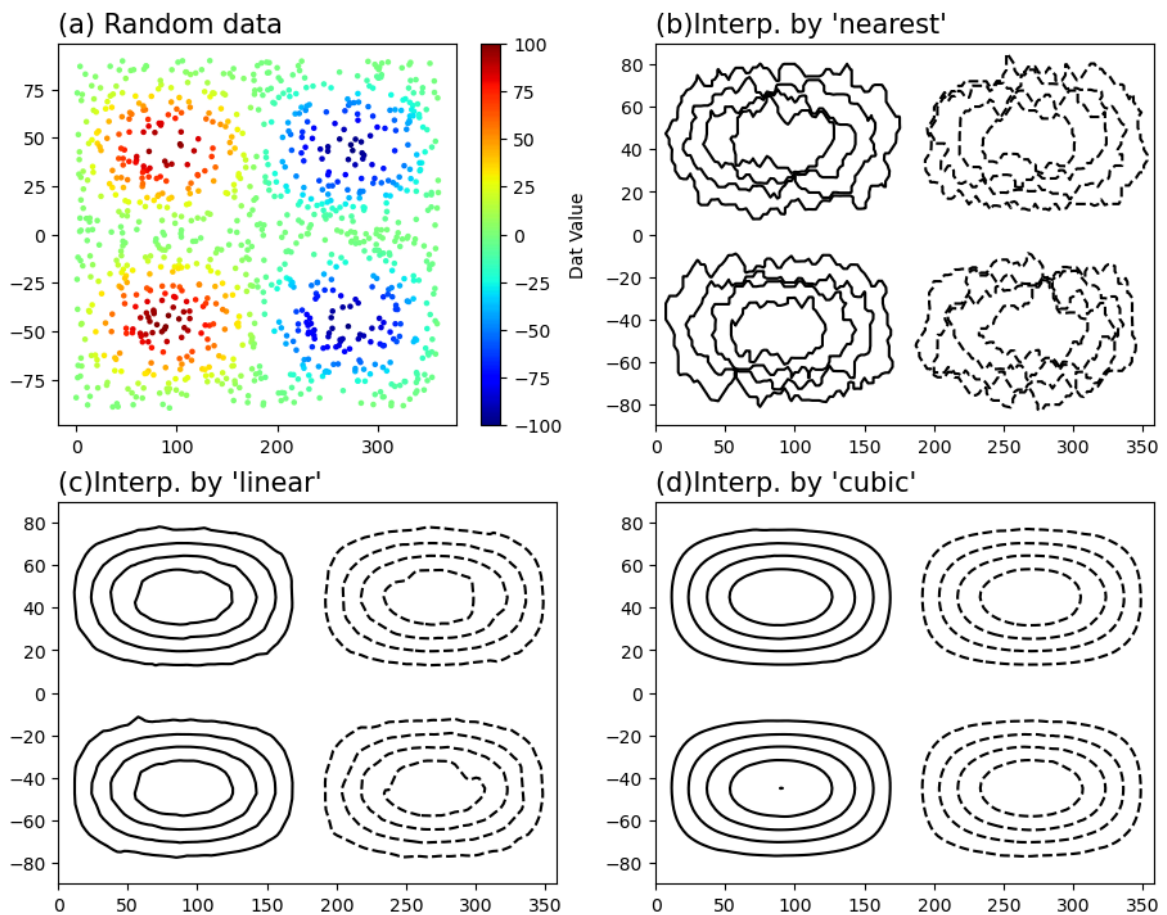
Markdown by 董慧杰

1. Scipy插值函数

a) griddata()

示例1:

```
In [1]: import numpy as np
from scipy.interpolate import griddata # 导入“scipy.interpolate”模块中的“griddat
import matplotlib.pyplot as plt
def crdat(): #生成随机经纬度数据
    np.random.seed(10)
    y = np.hstack((np.random.rand(500)*-90,np.random.rand(500)*90)) #随机生成100
    x = np.random.rand(1000)*360 #随机生成1000个经度数据
    m, n = (1,2)
    A = 100*np.sin(n*y/180*np.pi)**2
    dat = A*np.sin(m*x/180*np.pi)
    return x,y,dat
if __name__=='__main__':
    x,y,dat = crdat()
    lon0 = np.arange(0,360,2.5) #生成间隔为2.5的经度格点
    lat0 = np.linspace(-90,90,73) #生成间隔为2.5的纬度格点
    lon,lat = np.meshgrid(lon0,lat0) #生成2.5X2.5的经纬度二维网格
    #画出生成的随机散点图a
    fig = plt.figure(figsize=(11,8.5))
    ax = fig.add_subplot(2,2,1)
    pl1=ax.scatter(x,y,c=dat,cmap='jet',vmax=100,vmin=-100,s=5)
    ax.set_title('(a) Random data',fontsize=15,loc='left')
    plt.colorbar(pl1, ax=ax).set_label('Dat Value')
    levels = list(np.arange(-100,101,20))#设置等值线间隔
    levels.remove(0)
    for i,m in enumerate(['nearest','linear','cubic']): #循环三种插值方式，并绘图
        datg = griddata((x,y),dat,(lon,lat),method=m) # 利用griddata函数将dat(x,
        ax = fig.add_subplot(2,2,2+i)
        ax.contour(lon,lat,datg,levels=levels,colors='k')
        ax.set_title('('+chr(ord('b')+i)+ ')Interp. by '+'"+m+""',fontsize=15,1
```



b) interpolate.RBFInterpolator()

示例:

```
In [2]: import numpy as np
import matplotlib.pyplot as plt
from scipy import interpolate
def crdat():
    np.random.seed(10)
    y = np.hstack((np.random.rand(500)*-90, np.random.rand(500)*90))
    x = np.random.rand(1000)*360
    m, n = (1, 2)
    A = 100*np.sin(n*y/180*np.pi)**2
    dat = A*np.sin(m*x/180*np.pi)
    return x, y, dat
if __name__ == '__main__':
    x, y, dat = crdat()
    xy = np.vstack((y, x)).T
    print(xy.shape)
    lon0 = np.arange(0, 360, 2.5) # 经度范围
    lat0 = np.arange(-90, 91, 2.5) # 纬度范围
    lon, lat = np.meshgrid(lon0, lat0) # 创建网格
    lonlat = np.vstack((lat.flatten(), lon.flatten())).T # 经纬度扁平化, 组合为
    print(lon.shape, lat.shape, lonlat.shape)
    methods = ['multiquadric', 'inverse_quadratic', 'inverse_multiquadric', 'gau
               'linear', 'cubic', 'quintic', 'thin_plate_spline'] # 插值方法
    levels = list(np.arange(-100, 101, 20))
    levels.remove(0)
    fig = plt.figure(figsize=(8.5, 11))
    for i, m in enumerate(methods):
        ax = fig.add_subplot(4, 2, i+1)
```

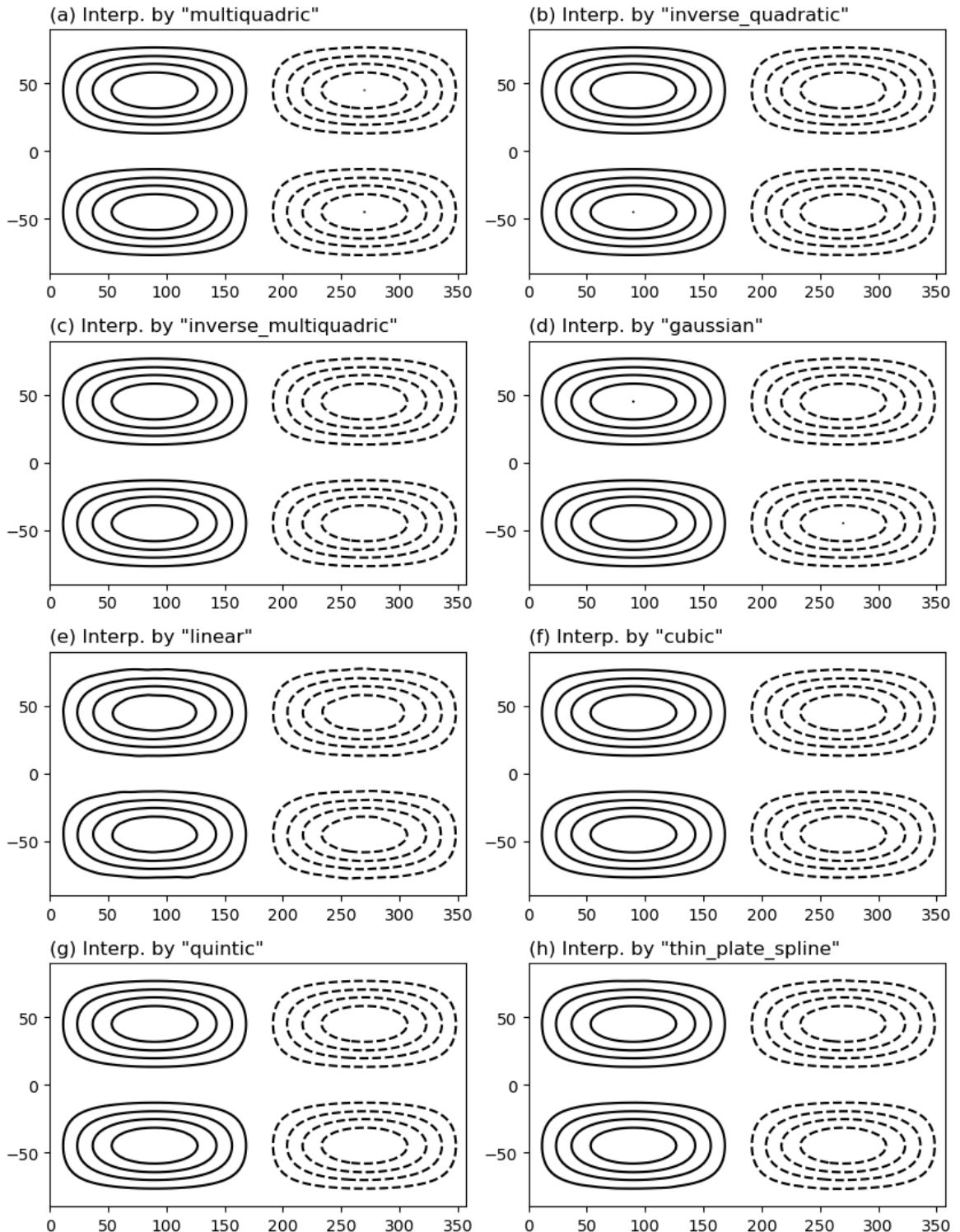
```

# 初始化一个 RBF 插值器对象rbf: xy(1000,2)表示1000个数据点, 2表示是2维; a
# kernel表示径向基函数类型; epsilon 控制插值平滑程度, 越小越平滑 (插值扫
rbf = interpolate.RBFInterpolator(xy, dat, kernel='m', epsilon=0.05)
# 将lonlat网格点传入已经初始化的插值器 rbf 中, 计算出这些网格点上对应的插
#注意: lonlat为(73X144,2), 表示有73*144格点, 2维数组
datg = rbf(lonlat)
# 将结果重新调整为2维 (73, 144) 形状, 便于绘图
datg = datg.reshape(lon.shape)
ax.contour(lon, lat, datg, levels=levels, colors='k')
ax.set_title('(' + chr(ord('a')) + i + ') Interp. by "' + m + '"', loc='
plt.tight_layout()
plt.show()

```

(1000, 2)

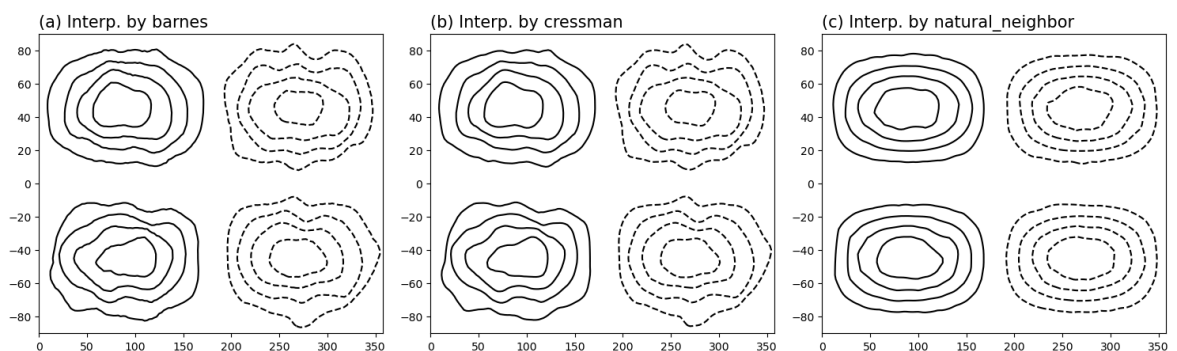
(73, 144) (73, 144) (10512, 2)



2. Metpy插值函数

示例：

```
In [3]: import numpy as np
from metpy.interpolate import interpolate_to_grid
import matplotlib.pyplot as plt
def crdat():
    np.random.seed(5)
    y = np.hstack((np.random.rand(500)*-90, np.random.rand(500)*90))
    x = np.random.rand(1000)*360
    m, n = (1, 2)
    A = 100 * np.sin(n * y / 180 * np.pi)**2
    dat = A * np.sin(m * x / 180 * np.pi)
    return x, y, dat
if __name__ == '__main__':
    x, y, dat = crdat()
    levels = list(np.arange(-100, 101, 20))
    levels.remove(0)
    fig = plt.figure(figsize=(15, 8.5))
    methods = ['barnes', 'cressman', 'natural_neighbor']
    for i, m in enumerate(methods):
        ax = fig.add_subplot(2, 3, i + 1)
        #将dat(x,y)插值到2.5的全球经纬度网格。hres表示格距；boundary_coords表示4个边界坐标
        #minimum_neighbors表示至少利用周边几个点，用于“cressman”和“barnes”插值，默认为3
        #“gamma”仅用于“barnes”插值的平滑，值越大平滑程度越大
        #search_radius”，默认值为散点间平均距离的5倍
        lon, lat, datg = interpolate_to_grid(x, y, dat,
                                             interp_type=m, hres=2.5, # 网格分辨率，2.5表示2.5度一
                                             boundary_coords={'west': 0, 'east': 357.5, 'south': -90, 'north': 90},
                                             minimum_neighbors=3, gamma=5.)
        cs = ax.contour(lon, lat, datg, levels=levels, colors='k')
        ax.set_title(f'({chr(ord("a")+i)}) Interp. by {m}', loc='left', fontsize=12)
    plt.tight_layout()
    plt.show()
```



注意：虽然Metpy 的计算均需要单位，但此代码不为任何变量提供单位，将所有变量的插值当作无量纲数进行处理，与动力诊断不同，这样做不会影响插值结果。

3. 高分辨率降为低分辨率

当某一数据的空间分辨率较高，需将其与低分辨率的网格数据进行比较时,除了前面介绍的插值方法，还可用[scipy.stats.binned_statistic_2d\(\)](#)函数

示例：

```

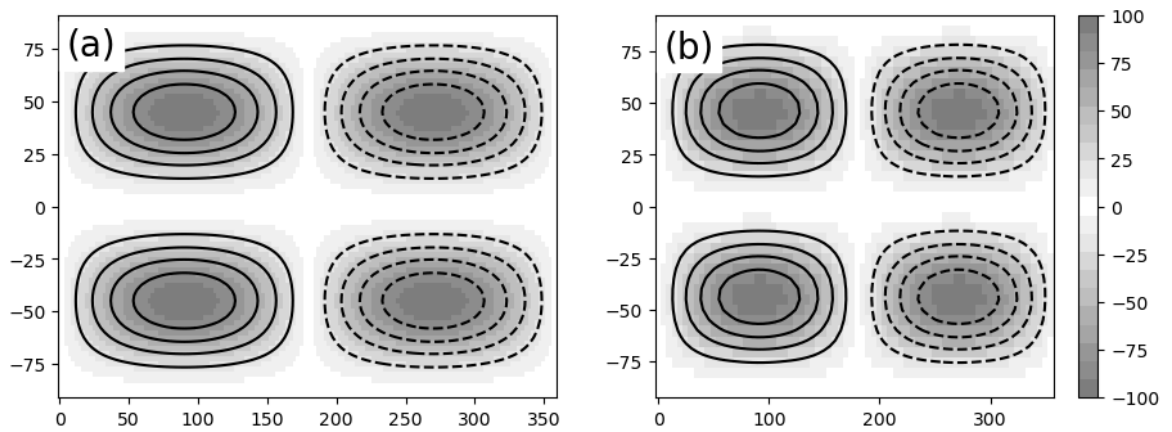
In [4]: import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import binned_statistic_2d #Scipy 的“scipy.stats.binned_statistic_2d”
from matplotlib.colors import LinearSegmentedColormap
def crdat():
    lon = np.arange(0, 360, 2.5)
    lat = np.arange(-90, 91, 2.5)
    x, y = np.meshgrid(lon, lat)
    m, n = (1, 2)
    A = 100 * np.sin(n * y / 180 * np.pi) ** 2
    dat = A * np.sin(m * x / 180 * np.pi)
    return lon, lat, x, y, dat

if __name__ == '__main__':
    lon, lat, x, y, dat = crdat()
    levels = list(np.arange(-100, 101, 20))
    x = x.flatten()
    y = y.flatten()
    z = dat.flatten()
    levels.remove(0)
    lonn = np.arange(-2.5, 362.5, 5) #分辨率为5度
    latn = np.arange(-92.5, 97.5, 5) #分辨率为5度

    # 对二维数据（经纬度x,y和数据值dat）进行分箱统计,统计方法为'mean',
    # latn 和 lonn 分别是纬度和经度的边界值，表示经纬度网格的边界点，即“箱子边界”
    datg = binned_statistic_2d(y, x, z, statistic='mean', bins=(latn, lonn))
    #提取出分箱统计结果
    datg=datg[0]
    #用网格中心经纬度表示位置新格点的经纬度
    lat1 = (latn[1:] + latn[:-1]) / 2 # 计算纬度网格的中点
    lon1 = (lonn[1:] + lonn[:-1]) / 2 # 计算经度网格的中点
    print(datg.shape)
    levels = list(np.arange(-100,101,20))
    levels.remove(0)
    fig = plt.figure(figsize=(11, 8.5))
    from matplotlib.colors import LinearSegmentedColormap
    colors = ['gray', 'w', 'gray']
    n_bin = 21
    cmap = LinearSegmentedColormap.from_list('gwg', colors, N=n_bin)
    for i, (x, y, z) in enumerate(zip([lon, lon1], [lat, lat1], [dat, datg])):
        ax = fig.add_subplot(2, 2, i + 1)
        c = ax.pcolor(x, y, z, vmin=-100, vmax=100, cmap=cmap)
        ax.contour(x, y, z, levels=levels, colors='k')
        ax.text(5, 72, f'({chr(ord("a") + i)})', fontsize=20, backgroundcolor='w')
    plt.colorbar(c, ax=ax)
    plt.show()

```

(37, 72)



注意：Xarray 提供的“`interp()`”函数也可实现对数据进行任意分辨率的插值。“`interp()`”函数的输入输出都是DataArray 或Dataset 结构的数据，不可用于Numpy 的Ndarray 数据（请参考

<https://docs.xarray.dev/en/stable/generated/xarray.DataArray.interp.html>）

课堂练习

1.综合练习题：使用不同插值方法对一维、二维数据进行插值 假设你有以下一维和二维数据，要求通过不同的插值方法，估算某些未知点的值。你将使用三种插值方法：

`interp1d()`、`griddata()` 和 `Rbf`。其中，`interp1d()` 用于一维插值，`griddata()` 用于二维网格数据插值，而 `Rbf` 用于径向基函数插值。

一维数据：

```
x = np.array([0, 1, 2, 3, 4, 5])
y = np.sin(x)
```

二维数据：

```
x = np.array([0, 1, 2, 3])
y = np.array([0, 1, 2, 3])
X, Y = np.meshgrid(x, y)
Z = np.sin(X) + np.cos(Y)
```

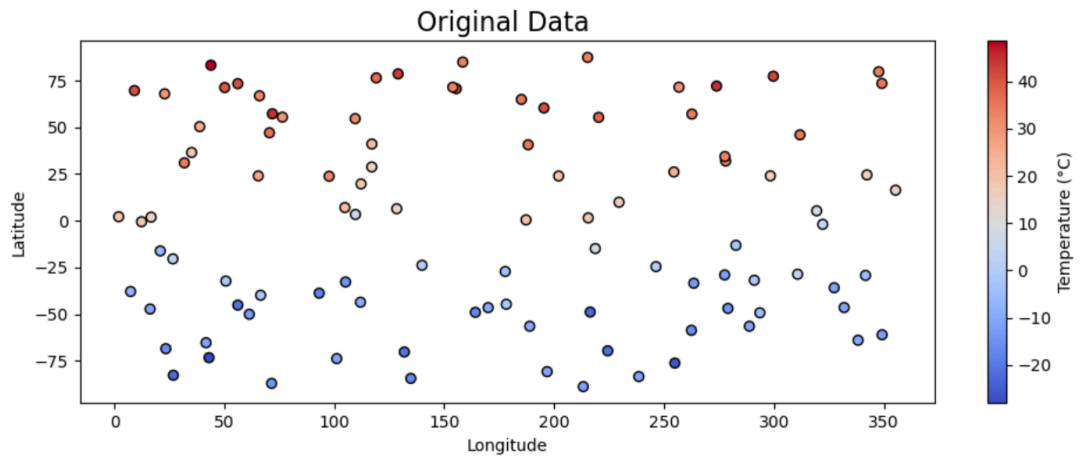
RBF插值点：

```
points = np.array([[0, 0], [1, 0], [0, 1], [1, 1]])
values = np.array([0, 1, 2, 3])
```

► 参考答案

2.请将下面程序补充完整，实现如下功能：利用 MetPy 的插值函数将下图所示的气象站点数据（包括经度、纬度和气温值），插值到一个规则的全球经纬度网格中，并展示插值

后的气温分布。



将如下程序空白处补充完整：

```
import numpy as np
from metpy.interpolate import interpolate_to_grid
import matplotlib.pyplot as plt

# 生成虚拟气象数据（经度、纬度、气温）
def generate_weather_data():
    # 随机生成100个气象站点的经纬度数据
    np.random.seed(42)
    lon = np.random.rand(100) * 360 # 经度 [0, 360)
    lat = np.random.rand(100) * 180 - 90 # 纬度 [-90, 90)

    # 生成与经纬度相关的气温数据：假设气温随纬度变化
    temperature = 30 * np.sin(np.radians(lat)) + 20 *
np.random.rand(100) # 假设气温随纬度变化

    return lon, lat, temperature

# 主程序
if __name__ == '__main__':
    # 获取气象数据
    lon, lat, temperature = generate_weather_data()

    # 设置插值网格的分辨率（这里是2度的网格）
    hres = 2.0 # 网格分辨率为2.0度
    levels = np.arange(-30, 50, 5) # 气温的等温线级别，范围从-30到50，步
长为5

    # 创建图形
    fig = plt.figure(figsize=(10, 8))

    # 请将此处补充完整,实现使用 MetPy 函数进行插值插值到全球经纬度网格（2.0
    度分辨率）

    # 请将此处补充完成，实现绘制插值后的等温线图
```

```
# 显示图形
plt.tight_layout()
plt.show()
```

► 参考答案