姓名：　栗浩宇　　学号：　20222490

《GNSS 定位新技术及数据处理方法》课后大作业（二）

GNSS 空间天气监测分析实验

任务部分：每位同学从 http://www.igs.gnsswhu.cn 或者 https://rinex.geodetic.gov.hk/rinex3 或者下载原始观测数据，然后利用 http://192.144.235.90/NEU2DTEC 处理生成 VTEC 数据，在此基础上，至少完成如下两项任务中的一项：

任务一： 1）对比分析至少 6 个不同纬度测站的电离层差异，并探讨原因；2）利用多项式拟合生成香港区域电离层模型。

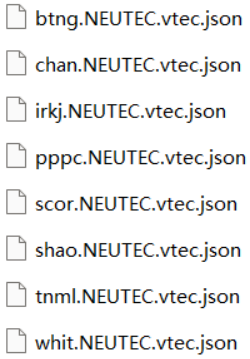
一、 算法

1 不同纬度测站的电离层差异分析

从 http://www.igs.gnsswhu.cn 下载了 2025 年 4 月 22 日（年积日 112）不同纬度测站的观测值文件, 一共下载了 8 个测站，其纬度信息如下表所示，然后下载了 2025 年 4 月 22 日的精密星历文件，导入 NEU2DTEC 生成 VTEC 数据。

测站	BTNG	PPPC	TNML	SHAO	CHAN	IRKJ	WHIT	SCOR
纬度/°	1.439	9.773	24.798	31.100	43.791	52.219	60.751	70.485

转换后的 VTEC 数据如下，共有 8 个 json 文件：



接下来可以用网站中提供的绘图代码绘制这 8 个测站的 VTEC 随时间（历元）的变化情况， 其中发现 BTNG 测站数据极少，TNML 测站数据不符合要求，

所以采用剩余的 6 个测站进行分析，用 Python 对这 **6 个**测站进行统计分析，统计内容包括：均值、最大值、最小值、变化幅度，以及相关系数的计算和相关性是否显著的判断。

相关系数的计算基于如下公式(Pearson 公式)：

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2 \sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

其中：

X_i和 Y_i是两个变量的观测值（此处为纬度和平均 VTEC 值）

$\bar{X}$和 $\bar{Y}$是它们各自的平均值

n是样本数量（此处为测站数量）

相关系数范围是 $[-1,1]$: 1 表示完全正相关，-1 表示完全负相关，0 表示无线性相关

相关性检验使用 T 分布，检验统计量 t 计算公式如下：

$$t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}}$$

其中r是$Pearson$相关系数，n是样本数量。

然后根据检验统计量t来确定p值，如果p值小于 0.05，则“相关性显著”。

以上算法结果的展示以及电离层差异原因的分析均在下文中进行。

2 多项式拟合香港地区电离层模型

同样地，下载了 2025 年 4 月 22 日香港地区的 18 个测站的观测值文件，然后进行了 VTEC 的计算，计算后发现，其中 HKFN、HKCL、HKMW、HKQT 测站的数据有问题，直接排除掉这 4 个测站，使用剩下的 **14 个**测站进行后续的模型生成。要注意的是：这里面还要用到测站的观测值文件(.rnx)，来读取测站的经纬度信息。

下面是本次多项式拟合香港地区电离层模型的算法介绍：

首先要进行经纬度的归一化，所用公式如下：

$$X_{norm} = 2 \cdot \frac{X - X_{min}}{X_{max} - X_{min}} - 1$$

将经纬度数据归一化到$[-1,1]$区间，这是为了防止高阶多项式计算中出现的数值不稳定问题。

接下来进行多项式特征的构建，将二维输入(经度,纬度)转换为多项式特征矩阵。例如，当多项式阶数为 5 时，对于输入特征(x,y)（归一化后的经纬度），会生成 21 个特征：

$$\Phi(x,y)=[1,x,y,x^2,xy,y^2,x^3,x^2y,xy^2,y^3,x^4,x^3y,x^2y^2,xy^3\ldots,xy^4,y^5]$$

接着就是多项式系数的求解问题，这里引入 ElasticNet 回归方法，主要是为了处理过拟合问题和提高模型泛化能力。

如果是传统的多项式拟合，使用普通的最小二乘法，其目标函数为：

$$\min_{\beta}|y-X\beta|_2^2$$

其中，X是多项式特征矩阵，β是待求系数向量，y是 VTEC 观测值。

ElasticNet 回归的目标函数为：

$$\min_{\beta}\ \frac{1}{2n}|y-X\beta|_2^2+\alpha\cdot\left[(1-l1\_ratio)\cdot\frac{1}{2}|\beta|_2^2+l1\_ratio\cdot|\beta|_1\right]$$

其中：$|y-X\beta|_2^2=\sum_{i=1}^{n}\left(y_i-\sum_{j=0}^{p}\beta_jX_{ij}\right)^2$，$|\beta|_2^2=\sum_{j=0}^{p}\beta_j^2$，$|\beta|_1=\sum_{j=0}^{p}|\beta_j|$。

综合上述过程，任意一点的 VTEC 值可通过下面这个公式预测：

$$\hat{z}=f(x,y)=\sum_{j=0}^{d}\sum_{k=0}^{d-j}\beta_{jk}\cdot x^jy^k$$

其中，x,y 是归一化到$[-1,1]$区间的经度和纬度坐标 ，d是多项式的最高阶数(代码中的 DEGREE 参数，β_{jk}是通过 ElasticNet 回归求解的系数)。

后续还用到的评估指标有：

决定系数(R^2)：$R^2=1-\frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}$

均方根误差$(RMSE)$：$RMSE=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}$

平均绝对误差(MAE)：$MAE=\frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i|$

对于残差插值，还用到三次样条插值公式：

$$f(x_q, y_q) = \sum_{i=1}^{n} w_i \cdot \phi(|p_i - p_q|)$$

以上就是多项式拟合香港地区电离层模型的所有算法，下面就用 14 个测站的数据，选择手动选择某一时刻进行电离层模型的多项式拟合。

二、 代码

1 不同纬度测站数据的统计分析（vtec_statistics.py）

```
import json
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
# 设置中文字体
plt.rcParams['font.sans-
serif'] = ['SimHei', 'DejaVu Sans', 'Arial Unicode MS']  # 优先使用的中文字
体
plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题
# 测站信息
stations = {
    "PPPC": 9.773,
    "SHAO": 31.100,
    "CHAN": 43.791,
    "IRKJ": 52.219,
    "WHIT": 60.751,
    "SCOR": 70.485
}
# 按纬度从低到高排序测站
sorted_stations = dict(sorted(stations.items(), key=lambda item: item[1]))
# 读取 VTEC 数据
def load_vtec_data(station, data_dir="./"):
    """
    读取 VTEC JSON 数据
    参数:
        station: 测站名称
        data_dir: 数据文件目录
    返回:
        vtec_data: 包含各卫星 VTEC 数据的字典
    """
    # 查找匹配的文件
    station_files = []
    for file in os.listdir(data_dir):
        # 检查文件名是否包含测站名称且为 JSON 文件
        if station.lower() in file.lower() and file.endswith(".vtec.json")
:
            station_files.append(os.path.join(data_dir, file))
    if not station_files:
```

```
        print(f"警告: 未找到测站 {station} 的 VTEC 数据文件")
        # 如果找不到文件，返回空字典或使用模拟数据
        return {}
    # 使用找到的第一个文件
    file_path = station_files[0]
    print(f"读取文件: {file_path}")
    try:
        with open(file_path, "r") as file:
            vtec_data = json.load(file)
        return vtec_data
    except Exception as e:
        print(f"读取文件 {file_path} 时出错: {e}")
        return {}
# 分析 VTEC 数据统计特征
def analyze_vtec_statistics(vtec_data):
    """分析 VTEC 数据的统计特征"""
    all_values = []
    for prn, values in vtec_data.items():
        # 过滤掉 0 值
        valid_values = [v for v in values if abs(v) > 1.0e-6]
        if valid_values:
            all_values.extend(valid_values)
    if not all_values:
        return {
            "mean": 0,
            "max": 0,
            "min": 0,
            "std": 0,
            "range": 0
        }
    return {
        "mean": np.mean(all_values),
        "max": np.max(all_values),
        "min": np.min(all_values),
        "std": np.std(all_values),
        "range": np.max(all_values) - np.min(all_values)
    }
# 计算每个时间点的平均 VTEC
def calculate_hourly_vtec(vtec_data):
    # 获取第一颗卫星的数据长度
    prns = list(vtec_data.keys())
    if not prns:
        return []
    epoch_count = len(vtec_data[prns[0]])
    hourly_vtec = []
    for epoch in range(epoch_count):
        values = []
        for prn, data in vtec_data.items():
            # 检查数据有效性并过滤零值和异常值
            if epoch < len(data) and abs(data[epoch]) > 1.0e-
6 and data[epoch] < 200:
```

```
                values.append(data[epoch])
        # 只有当有足够有效值时才计算平均值
        if len(values) >= 3:  # 至少需要3颗卫星的有效数据
            hourly_vtec.append(np.mean(values))
        else:
            # 使用插值而非0值
            if hourly_vtec:
                hourly_vtec.append(hourly_vtec[-1])  # 使用前一个有效值
            else:
                hourly_vtec.append(np.nan)  # 标记为缺失值
    # 对NaN值进行插值处理
    hourly_vtec = np.array(hourly_vtec)
    mask = np.isnan(hourly_vtec)
    hourly_vtec[mask] = np.interp(
        np.flatnonzero(mask),
        np.flatnonzero(~mask),
        hourly_vtec[~mask]
    )
    return hourly_vtec.tolist()
# 分析所有测站
def analyze_all_stations():
    """分析所有测站的VTEC数据"""
    station_stats = {}
    hourly_data = {}
    for station in sorted_stations:
        vtec_data = load_vtec_data(station)
        stats = analyze_vtec_statistics(vtec_data)
        hourly_vtec = calculate_hourly_vtec(vtec_data)
        station_stats[station] = stats
        hourly_data[station] = hourly_vtec
    return station_stats, hourly_data
# 绘制结果
def plot_results(station_stats, hourly_data):
    """绘制分析结果"""
    # 1. 不同测站VTEC统计值
    stations = list(station_stats.keys())
    latitudes = [sorted_stations[s] for s in stations]
    means = [station_stats[s]["mean"] for s in stations]
    maxes = [station_stats[s]["max"] for s in stations]
    mins = [station_stats[s]["min"] for s in stations]
    ranges = [station_stats[s]["range"] for s in stations]
    # 创建图形
    plt.figure(figsize=(10, 8))
    # VTEC随纬度变化
    plt.plot(latitudes, means, 'o-', label='平均VTEC')
    plt.plot(latitudes, maxes, '^-', label='最大VTEC')
    plt.plot(latitudes, mins, 'v-', label='最小VTEC')
    plt.plot(latitudes, ranges, 's-', label='VTEC变化范围')
    plt.xlabel('纬度 (°N)')
    plt.ylabel('VTEC (TECU)')
```

```
    plt.title('不同纬度测站的 VTEC 统计特征')
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.savefig('./result/vtec_latitude_analysis.png', dpi=300)
    plt.close()
    # 3. 生成统计表格
    stats_df = pd.DataFrame({
        '测站': stations,
        '纬度(°N)': latitudes,
        '平均 VTEC(TECU)': means,
        '最大 VTEC(TECU)': maxes,
        '最小 VTEC(TECU)': mins,
        'VTEC 变化幅度(TECU)': ranges
    })
    print(stats_df)
    try:
        os.makedirs('./result', exist_ok=True)
        stats_df.to_csv('./result/vtec_statistics.csv', index=False, encod
ing='utf-8-sig')
    except Exception as e:
        print(f"保存 UTF-8-sig 格式失败: {e}")
    return stats_df
# 主分析函数
def main_analysis(data_dir="./"):
    """
    主分析函数
    参数:
        data_dir: 数据文件目录
    """
    # 检查数据文件
    print(f"检查目录 {data_dir} 中的数据文件...")
    all_files = os.listdir(data_dir)
    vtec_files = [f for f in all_files if f.endswith(".vtec.json")]
    print(f"找到 {len(vtec_files)} 个 VTEC 数据文件: {vtec_files}")
    # 分析所有测站
    station_stats = {}
    hourly_data = {}
    for station in sorted_stations:
        print(f"\n 处理测站: {station}，纬度: {sorted_stations[station]}°N")
        vtec_data = load_vtec_data(station, data_dir)
        if not vtec_data:
            print(f"未找到测站 {station} 的有效数据，跳过此测站")
            continue
        print(f"成功读取 {station} 测站数据，包含 {len(vtec_data)} 颗卫星的
VTEC 记录")
        stats = analyze_vtec_statistics(vtec_data)
        hourly_vtec = calculate_hourly_vtec(vtec_data)
        station_stats[station] = stats
        hourly_data[station] = hourly_vtec
```

```
        print(f"测站 {station} 统计数据: 平均
VTEC={stats['mean']:.2f} TECU, 最大值={stats['max']:.2f} TECU")
    if not station_stats:
        print("未找到任何测站的有效数据，无法进行分析")
        return
    stats_df = plot_results(station_stats, hourly_data)
    # 分析结果
    print("\n不同纬度测站电离层 VTEC 差异分析结果:")
    print("-" * 60)
    # 找出最大 VTEC 的测站和纬度
    max_vtec_station = stats_df.loc[stats_df['最大 VTEC(TECU)'].idxmax()]
    print(f"最大 VTEC 值出现在 {max_vtec_station['测站']} 测站 (纬
度: {max_vtec_station['纬度(°N)']}°N)")
    # 低纬度和高纬度的差异
    low_lat_df = stats_df[stats_df['纬度(°N)'] < 30]
    high_lat_df = stats_df[stats_df['纬度(°N)'] > 55]
    if not low_lat_df.empty and not high_lat_df.empty:
        low_lat_mean = low_lat_df['平均 VTEC(TECU)'].mean()
        high_lat_mean = high_lat_df['平均 VTEC(TECU)'].mean()
        diff_percent = (low_lat_mean - high_lat_mean) / high_lat_mean * 10
0 if high_lat_mean > 0 else 0
        print(f"低纬度地区平均 VTEC 比高纬度地区高约 {diff_percent:.1f}%")
    # 电离层赤道异常特征
    eq_anomaly_stations = stats_df[(stats_df['纬度
(°N)'] >= 10) & (stats_df['纬度(°N)'] <= 20)]
    if not eq_anomaly_stations.empty:
        print(f"观察到电离层赤道异常现象，赤道附近±15°纬度区域 VTEC 值较高")
    # 尝试计算纬度相关性
    try:
        from scipy.stats import pearsonr
        if len(stats_df) >= 3:  # 至少需要 3 个点才能计算有意义的相关性
            corr, p_value = pearsonr(stats_df['纬度(°N)'], stats_df['平均
VTEC(TECU)'])
            print(f"VTEC 与纬度的相关系数为: {corr:.6f}")
            if p_value < 0.05:
                print(f"   相关性显著: (p-value={p_value:.6f})")
            else:
                print(f"   相关性不显著: (p-value={p_value:.6f})")
        else:
            print("测站数量不足，无法计算 VTEC 与纬度的相关性")
    except Exception as e:
        print(f"计算相关性时出错: {e}")
    return stats_df
# 执行分析
if __name__ == "__main__":
    data_dir = "vtecs"
    main_analysis(data_dir)
```

2 多项式拟合电离层模型(hk_ionosphere_polyfit.py)

```
import os
import json
import math
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.path import Path as MplPath
from scipy.spatial import ConvexHull, Voronoi, voronoi_plot_2d
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import ElasticNet
from scipy.interpolate import griddata
import matplotlib.pyplot as plt
# 设置中文字体
plt.rcParams['font.sans-
serif'] = ['SimHei', 'DejaVu Sans', 'Arial Unicode MS']
plt.rcParams['axes.unicode_minus'] = False
# 基本参数
RNX_DIR  = Path(r'hkdata')   # RINEX 数据目录
VTEC_DIR = Path(r'HKVTEC')  # VTEC 数据目录
DEGREE     = 5        # 多项式阶次
ALPHA      = 0.0009  # ElasticNet 正则化强度
L1_RATIO   = 0.5    # ElasticNet 的L1 比例
SAMPLE_SEC = 8*3600  # 选取 8:00
INTVL_SEC  = 30     # 观测采样间隔（秒）
def ecef2geodetic(x: float, y: float, z: float):
    """WGS-84 椭球下 ECEF → (lat, lon)  (度)"""
    a  = 6378137.0            # 长半轴
    e2 = 6.69437999014e-3     # 第一偏心率平方
    b  = a * math.sqrt(1 - e2)
    lon = math.atan2(y, x)
    p   = math.hypot(x, y)
    theta = math.atan2(z * a, p * b)
    lat = math.atan2(z + e2 * b * math.sin(theta)**3,
                     p - e2 * a * math.cos(theta)**3)
    return math.degrees(lat), math.degrees(lon)
def read_rinex_positions(folder: Path):
    """返回 {站名: (lon, lat)}"""
    pos = {}
    for file in folder.glob('*.rnx'):
        sta = file.stem[:4].lower()
        with file.open('r', errors='ignore') as fh:
            for line in fh:
                if 'APPROX POSITION XYZ' in line:
                    x, y, z = map(float, line.split()[:3])
```

```
                    lat, lon = ecef2geodetic(x, y, z)
                    pos[sta] = (lon, lat)
                    break
    return pos
def read_vtec_at_epoch(folder: Path, epoch_idx: int):
    """返回 {站名: vtec_value}"""
    vtec = {}
    for file in folder.glob('*.json'):
        sta = file.stem[:4].lower()
        with file.open('r', encoding='utf-8') as fh:
            data = json.load(fh)
        vals = []
        for prn, series in data.items():
            if epoch_idx < len(series):
                val = series[epoch_idx]
                if val is not None and not (isinstance(val, float) and mat
h.isnan(val)):
                    vals.append(val)
        if vals:
            vtec[sta] = float(np.mean(vals))
    return vtec
def normalize(arr):
    """线性映射到 [-1, 1]"""
    mn, mx = float(np.min(arr)), float(np.max(arr))
    return 2*(arr - mn)/(mx - mn) - 1, mn, mx
# 读取数据
stations_ll = read_rinex_positions(RNX_DIR)
epoch_idx   = SAMPLE_SEC // INTVL_SEC
stations_vt = read_vtec_at_epoch(VTEC_DIR, epoch_idx)
common_keys = sorted(set(stations_ll) & set(stations_vt))
if len(common_keys) < 4:
    raise RuntimeError('有效测站不足，无法拟合多项式')
lon_arr = np.array([stations_ll[k][0] for k in common_keys])
lat_arr = np.array([stations_ll[k][1] for k in common_keys])
vtec_arr = np.array([stations_vt[k]    for k in common_keys])
# 多项式拟合
lon_n, lon_min, lon_max = normalize(lon_arr)
lat_n, lat_min, lat_max = normalize(lat_arr)
X_norm = np.column_stack((lon_n, lat_n))
poly = PolynomialFeatures(degree=DEGREE, include_bias=True)
X_poly = poly.fit_transform(X_norm)
model = ElasticNet(alpha=ALPHA, l1_ratio=L1_RATIO, fit_intercept=False, to
l=1e-8,max_iter=10000)
model.fit(X_poly, vtec_arr)
r2 = model.score(X_poly, vtec_arr)
print(f'>>> 已拟合 {DEGREE} 阶多项式，R² = {r2:.3f}，样本
数 = {len(common_keys)}')
# 生成网格
lon_lin = np.linspace(lon_min, lon_max, 200)
lat_lin = np.linspace(lat_min, lat_max, 200)
grid_lon, grid_lat = np.meshgrid(lon_lin, lat_lin)
```

```
grid_pts = np.column_stack((grid_lon.ravel(), grid_lat.ravel()))
# 利用测站凸包裁剪，避免外插
hull = ConvexHull(np.column_stack((lon_arr, lat_arr)))
hull_path = MplPath(np.column_stack((lon_arr, lat_arr))[hull.vertices])
inside = hull_path.contains_points(grid_pts)
# 归一化 + 预测
lon_n_g = 2*(grid_pts[:, 0] - lon_min)/(lon_max - lon_min) - 1
lat_n_g = 2*(grid_pts[:, 1] - lat_min)/(lat_max - lat_min) - 1
Z_pred  = model.predict(poly.transform(np.column_stack((lon_n_g, lat_n_g))
))
Z_grid = np.full(grid_lon.shape, np.nan)
Z_grid.ravel()[inside] = Z_pred[inside]
# 计算残差
station_pred = model.predict(X_poly)
residuals = vtec_arr - station_pred
abs_residuals = np.abs(residuals)
# 图1: 残差热力图
plt.figure(figsize=(8, 6))
plt.title('残差空间分布热力图', fontsize=15)
grid_x, grid_y = np.mgrid[lon_min:lon_max:100j, lat_min:lat_max:100j]
grid_residuals = griddata((lon_arr, lat_arr), residuals, (grid_x, grid_y),
 method='cubic')
hull_mask = np.zeros_like(grid_residuals, dtype=bool)
for i in range(grid_x.shape[0]):
    for j in range(grid_x.shape[1]):
        if not hull_path.contains_point((grid_x[i, j], grid_y[i, j])):
            hull_mask[i, j] = True
grid_residuals = np.ma.array(grid_residuals, mask=hull_mask)
cmap = plt.cm.coolwarm
im = plt.pcolormesh(grid_x, grid_y, grid_residuals, cmap=cmap, shading='au
to',
                    vmin=-
np.max(abs_residuals), vmax=np.max(abs_residuals))
plt.colorbar(im, label='残差 (TECU)')
plt.scatter(lon_arr, lat_arr, c='k', s=30, marker='^', label='测站位置')
for i, txt in enumerate(common_keys):
    plt.annotate(f'{txt}\n({residuals[i]:.2f})', (lon_arr[i], lat_arr[i]),
                 fontsize=9, ha='center', va='bottom', color='k')
plt.xlabel('经度 (°E)')
plt.ylabel('纬度 (°N)')
plt.grid(True, linestyle='--', alpha=0.3)
plt.legend(loc='best')
rmse = np.sqrt(np.mean(residuals**2))
plt.figtext(0.5, 0.01,
            f"均方根误差(RMSE): {rmse:.3f} TECU | "
            f"最大残差: {np.max(abs_residuals):.3f} TECU | "
            f"平均绝对残差: {np.mean(abs_residuals):.3f} TECU",
            ha='center', fontsize=11, bbox=dict(boxstyle="round,pad=0.5",
                                                fc="white", ec="gray", alpha
=0.8))
plt.tight_layout(rect=[0, 0.05, 1, 0.95])
```

```
plt.savefig('./result/残差热力图.png', dpi=300, bbox_inches='tight')
plt.show()
# 图2：3D 表面
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
vtec_cmap = plt.cm.plasma
surf = ax.plot_surface(grid_lon, grid_lat, Z_grid,
                       rstride=4, cstride=4,
                       cmap=vtec_cmap, edgecolor='none', alpha=0.90)
ax.scatter(lon_arr, lat_arr, vtec_arr, c='white', s=35, edgecolor='black',
 label='测站')
ax.set_xlabel('经度 (°E)')
ax.set_ylabel('纬度 (°N)')
ax.set_zlabel('VTEC (TECU)')
ax.set_title(f'香港 VTEC 3D 表面，使用 {DEGREE} 阶多项式拟合\n'
             f't = {SAMPLE_SEC//3600:02d}:{SAMPLE_SEC%3600//60:02d}')
fig.colorbar(surf, shrink=0.55, aspect=12, label='VTEC (TECU)')
plt.legend()
plt.tight_layout()
plt.savefig('./result/VTEC_3D 表面.png', dpi=300, bbox_inches='tight')
plt.show()
# 图3：2D 空间分布图
plt.figure(figsize=(8, 6))
plt.title('香港地区 VTEC 2D 空间分布图', fontsize=15)
contour_filled = plt.contourf(grid_lon, grid_lat, Z_grid, levels=15, cmap=
'jet', alpha=0.8)
contour_lines = plt.contour(grid_lon, grid_lat, Z_grid, levels=8, colors='
k', linewidths=0.5, alpha=0.7)
plt.clabel(contour_lines, inline=True, fontsize=8, fmt='%.1f')
plt.scatter(lon_arr, lat_arr, c='white', s=40, edgecolor='black', marker='
^', label='测站位置')
for i, txt in enumerate(common_keys):
    plt.annotate(txt, (lon_arr[i], lat_arr[i]), fontsize=9, ha='center', v
a='bottom')
cbar = plt.colorbar(contour_filled, label='VTEC (TECU)')
cbar.set_label('VTEC (TECU)', fontsize=12)
plt.xlabel('经度 (°E)', fontsize=12)
plt.ylabel('纬度 (°N)', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.3)
plt.legend(loc='best')
plt.axis('equal')
plt.tight_layout()
plt.savefig('./result/VTEC_2D 空间分布图.png', dpi=300, bbox_inches='tight')
plt.show()
print("\n 图表生成完成！共生成 3 张图：")
print("1. 残差热力图")
print("2. VTEC_3D 表面图")
print("3. VTEC_2D 空间分布图")
```

三、 结果

1　不同纬度测站的电离层差异分析

1.1 电离层数据的时序可视化

这部分直接使用网站提供的代码，得到结果如下：

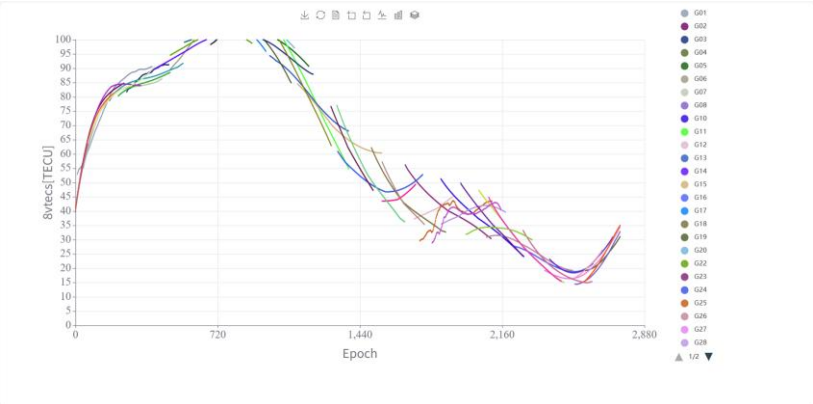


图 1 PPPC 测站

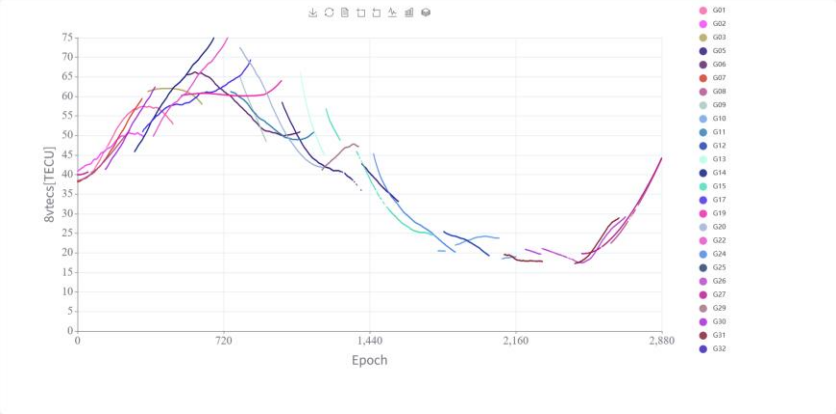


图 2 SHAO 测站

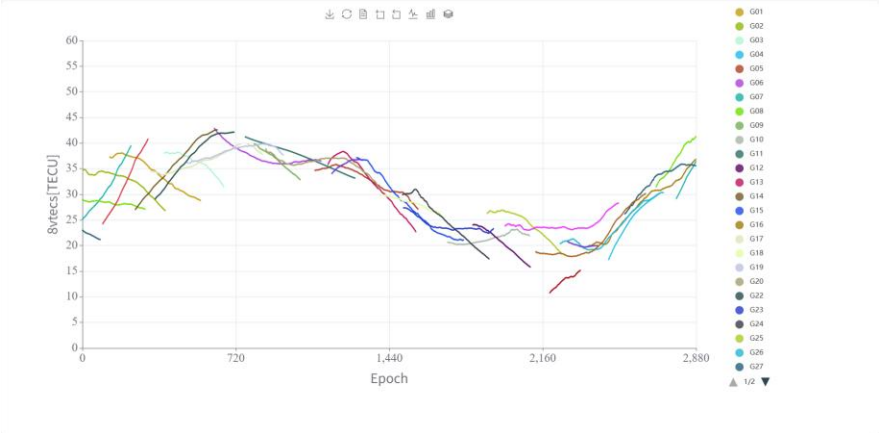


图 3 CHAN 测站

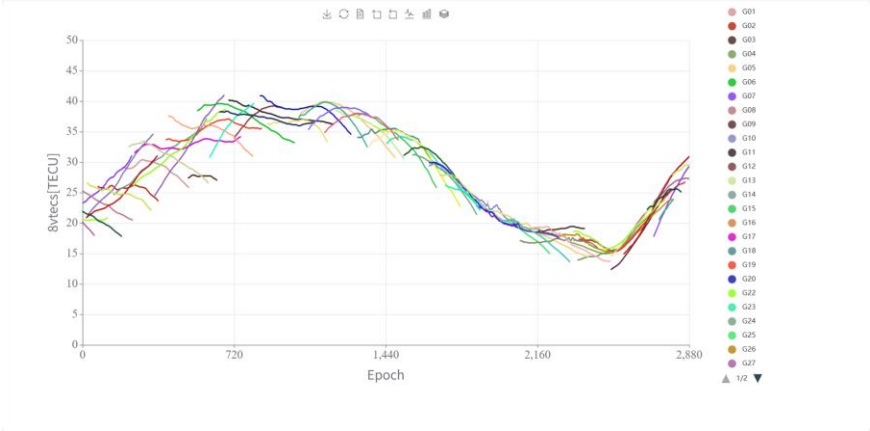


图 4 IRKJ 测站

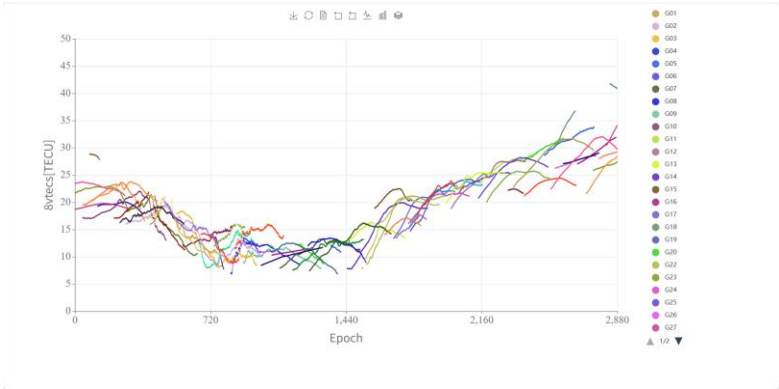


图 5 WHIT 测站

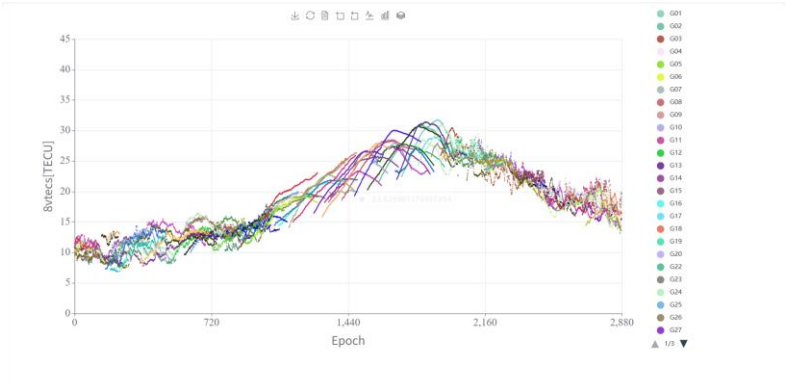


图 6 SCOR 测站

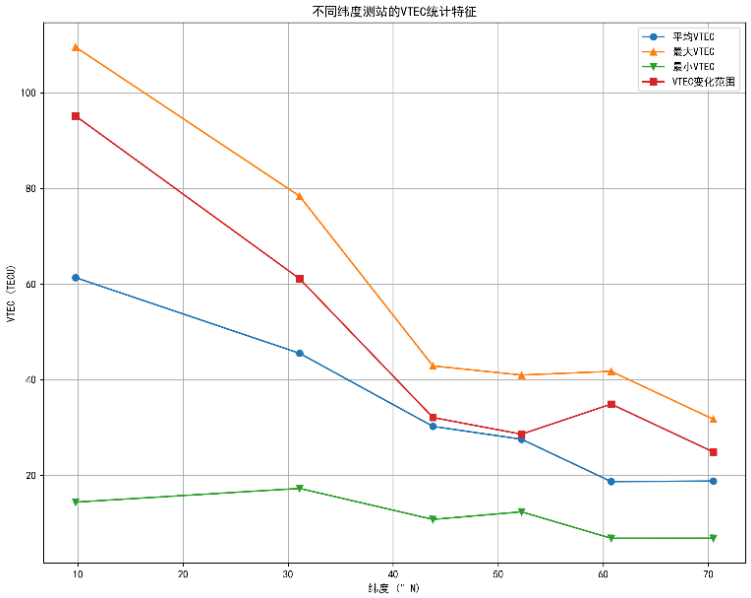
1.2 统计结果

运行代码后，会得到这 6 个测站的统计数据结果(打印并输出 csv 文件)，如下所示：

	测站	纬度(°N)	平均VTEC(TECU)	最大VTEC(TECU)	最小VTEC(TECU)	VTEC变化幅度(TECU)
0	PPPC	9.773	61.363585	109.580851	14.406015	95.174836
1	SHAO	31.100	45.526511	78.394165	17.269896	61.124269
2	CHAN	43.791	30.242274	42.890997	10.788254	32.102743
3	IRKJ	52.219	27.557443	40.984591	12.373135	28.611457
4	WHIT	60.751	18.696384	41.752476	6.844894	34.907582
5	SCOR	70.485	18.806857	31.734747	6.855275	24.879472

除此之外，还会得到差异分析结果和统计数据可视化结果：

```
不同纬度测站电离层VTEC差异分析结果:
------------------------------------------------------------
最大VTEC值出现在 PPPC 测站 (纬度: 9.773°N)
低纬度地区平均VTEC比高纬度地区高约 227.2%
VTEC与纬度的相关系数为：-0.981738
   相关性显著：(p-value=0.000497)
```



1.3 对比分析 6 个测站电离层差异

我把纬度低于 30° 划为低纬度地区，纬度在 30° 到 55° 之间划为中纬度地区，纬度高于 55° 划为高纬度地区。

统计发现：

低纬度地区的 VTEC 值最高，显示出明显的高 VTEC 特征。低纬度地区的 VTEC 波动较大，尤其是最大值和最小值之间的差距较大。

中纬度地区的 VTEC 值低于低纬度地区，整体呈现逐渐降低的趋势。虽然中纬度地区的 VTEC 值相较于低纬度有所下降，但仍然具有一定的变化幅度，尤其在最大值和最小值之间存在显著差距。

高纬度地区的 VTEC 值最低，且变化幅度相对较小，WHIT 和 SCOR 测站的 VTEC 值接近。高纬度地区的 VTEC 值明显低于低纬度和中纬度地区。

根据输出结果还可以看出：低纬度地区显示出比高纬度地区（纬度 > 55° N）更高的平均 VTEC 值。具体来说，低纬度地区的平均 VTEC 值比高纬度地区高约 227.2%。

通过计算 VTEC 与纬度的相关系数，结果为“-0.981738”，显示出强烈的负相关关系。这意味着**随着纬度的增加，VTEC 值通常下降**。并且相关性检验的 p 值为 0.000497，远小于 0.05，表明此相关性是**显著的**。

1.4 探讨差异原因

1. 太阳辐射的纬度分布效应

低纬度地区太阳入射角接近垂直，单位面积接收的太阳辐射能量显著高于高纬度地区。电离层主要由太阳极紫外(EUV)和 X 射线辐射电离大气产生，垂直入射时的电离效率最高。根据 Chapman 电离理论，电离产生率与太阳天顶角的余弦值成正比，从 PPPC(9.773°N)到 SCOR(70.485°N)，太阳平均天顶角逐渐增大，导致电离效率急剧下降，这是造成 VTEC 随纬度增加而递减的根本原因。

2. 地磁场配置的控制作用

地磁场在不同纬度的倾角差异显著影响等离子体的分布和输运。赤道地区地磁场近乎水平，而高纬度地区近乎垂直。由于等离子体扩散主要沿磁力线进行，赤道地区的水平磁场限制了等离子体的垂直扩散，使其更容易在电离层 F 层积累，形成较高的电子密度。相反，高纬度地区的垂直磁场允许等离子体更容易向下扩散并与中性成分复合，导致稳态电子密度较低。

3. 赤道电离层异常现象

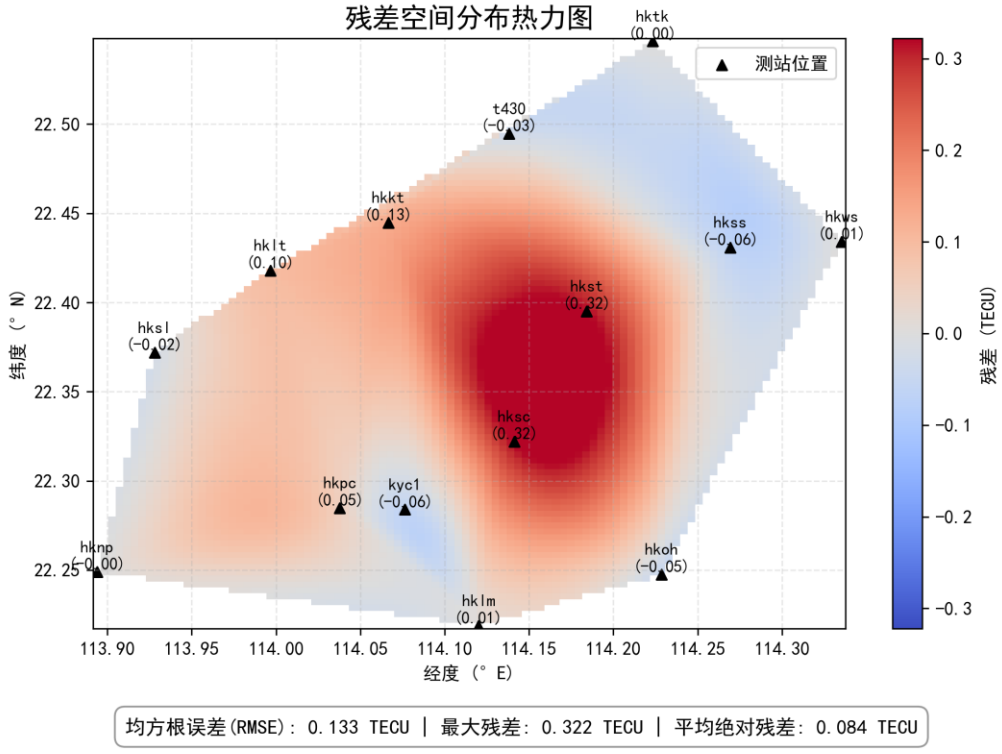
赤道电离层异常(EIA)是低纬度 VTEC 显著增强的重要机制。在地磁赤道地区，东向电场与水平磁场相互作用产生 E×B 垂直向上漂移，将等离子体抬升到高层。随后在重力和压力梯度作用下，等离子体沿磁力线向南北两侧扩散，在地磁纬

度±15-20°附近形成两个电子密度峰值。结果数据完美验证了这一现象：PPPC 测站(9.773°N)平均 VTEC 达 61.36 TECU，正好位于赤道异常北峰附近，而 SHAO 测站(31.100°N)仍部分受到异常区影响，VTEC 为 45.53 TECU。

4. 中性风系统的纬度差异

热层中性风对电离层的影响随纬度变化显著。白天从赤道向极地的子午风推动等离子体沿磁力线运动，在低纬度地区由于磁倾角小，上行中性风能有效提升电离层高度，延长等离子体寿命。而在高纬度地区，由于磁倾角大，中性风对电离层垂直分布的影响相对较小。此外，纬向风通过与磁场的相互作用在低纬度产生更强的垂直漂移，进一步增强了低纬度的 VTEC 值。

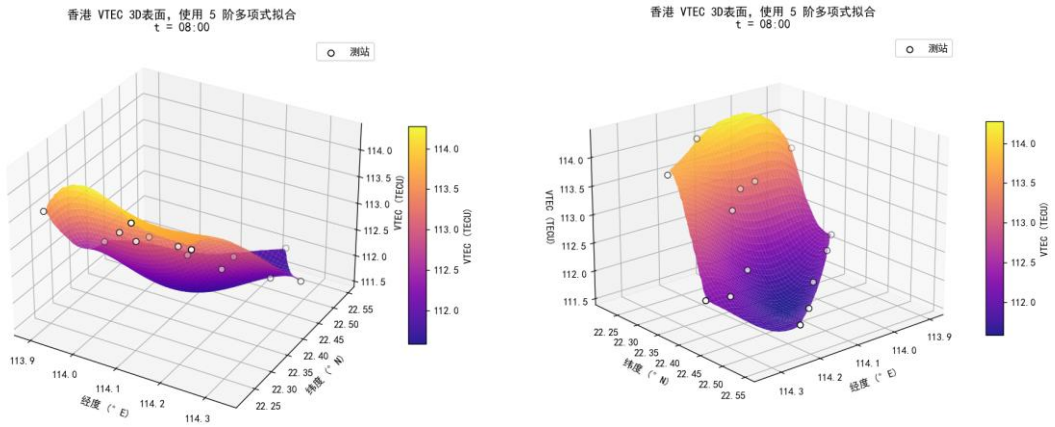
2 多项式拟合香港地区电离层模型



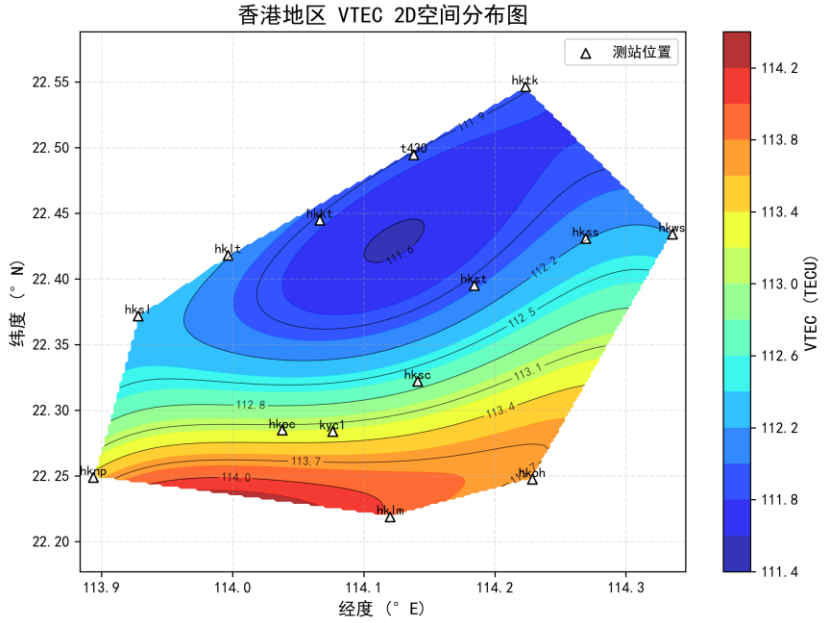
均方根误差(RMSE)：0.133 TECU | 最大残差：0.322 TECU | 平均绝对残差：0.084 TECU

从残差空间分布热力图我们可以看出，残差呈现明显的系统性空间分布，中心区域（约 114.15°E，22.35°N）显示较大正残差（0.2-0.3 TECU），周边区域为负残差（-0.1 到-0.2 TECU）。统计特征：RMSE（0.133 TECU）相对较小，表明整体建模精度良好最大残差（0.322 TECU）出现在中心区域。平均绝对残差（0.084 TECU）较低，说明模型总体性能可接受。

电离层模型的 3D 可视化(使用 **5 阶**多项式拟合)：



电离层模型的 2D 可视化：



控制台打印的模型信息：

```
已拟合 5 阶多项式，R² = 0.962，样本数 = 14
```

这里 R^2=0.962 意味着 5 阶多项式模型解释了 96.2%的电离层 VTEC 变化。效果还是非常好的。

四、 总结

本次任务围绕 GNSS 电离层监测展开，从指定网站下载 2025 年 4 月 22 日不同纬度测站的观测值文件及精密星历文件，利用 NEU2DTEC 生成 VTEC 数据，在筛选有效数据后，针对不同纬度测站电离层差异分析和香港区域电离层建模两项任务展开研究。

在不同纬度测站电离层差异分析中，选取 6 个有效测站（纬度范围 9.733°N 至 70.485°N），通过统计均值、最大值、最小值、变化幅度等特征并计算相关系数，发现低纬度地区 VTEC 值最高且波动较大，中纬度地区 VTEC 值逐渐降低，高纬度地区 VTEC 值最低且变化幅度较小，低纬度地区平均 VTEC 值比高纬度地区高约 227.2%，纬度与 VTEC 的相关系数为-0.9817，p 值为 0.000497，相关性显著，这主要是太阳辐射纬度分布、地磁场配置、赤道电离层异常及中性风系统差异等因素共同作用的结果。

在利用多项式拟合香港区域的电离层模型的任务中，使用 14 个有效测站数据，通过经纬度归一化、5 阶多项式特征构建及 ElasticNet 回归拟合模型，结果显示模型决定系数 R^2为 0.962，均方根误差为 0.133 TECU，平均绝对误差为 0.084 TECU，残差热力图显示中心区域有正残差、周边为负残差，3D 和 2D 可视化清晰呈现了香港地区 VTEC 的空间分布，模型精度较高且能有效捕捉区域电离层特征。

本次实验通过数据处理、统计分析和模型构建，验证了电离层纬度差异的物理机制，构建了适用于香港地区的电离层模型。在这个过程中，通过一个个问题的解决，也锻炼了我发现问题、处理问题的能力。