卫星数据可视化程序 用户手册

# 简介

卫星数据可视化程序(以下简称程序)是一个物理量可视化程序。程序的设计初衷是希望快速，高效地对空间卫星所探测的三维物理数据进行二维展示。操作人员可以通过程序可视化界面中的选择菜单来获得由不同切面、变量所组成的热力图。也可以随时将图像导出储存为本地文件。

程序是用Python实现的，因此可以在主流平台上运行（Mac, Windows）。在读取规定格式的数据表格(.csv)和坐标关系表格(.csv)后，程序会先将数据缩小到绘图所需要的范围，然后通过基于数据点在三维空间中的几何关系来填补缺失的物理量，最后以热力图的方式来呈现特定切面中物理量大小的分布情况。在运行环境条件满足的情况下，程序可以独立运行。

# 产品特性

2.1 环境及平台

可以在mac, windows 平台运行

需要Python 3.8.0及以上

需要预装的Python模块

数据存储/处理：Pandas, numpy

热力图绘制：matplotlib, seaborn

其他: tqdm, multiprocessing, sys, os

2.2 图形界面

互动绘图界面：两个选择器、文本框、绘制按钮（Process），四片绘图区

*Graphical user interface

Description automatically generated*

操作人员可以通过点击每一幅热力图左边的两个选择器来确认需要展示的物理量(如P), 二维切面(如Y-Z)，并通过文本框来选择第三切面维度(如2.0)，最后通过绘制按钮来完成四幅画面的呈现。

3. 模块功能及接口关系

Diagram

Description automatically generated

*1.模块接口关系图*

Plotter绘图模块

模块功能：

创建图形界面并监听用户在图形界面中选择的物理量，切面及第三维度数据。根据用户选择数据或默认值绘制热力图，并更新到图形界面上。

接口关系：

接收参数 (dat, stored, core\_num=4)：

字符串 文件路径 dat ;

字符串 文件路径 stored ；

整数类型 多线程处理器数量 core\_num 默认为4

内部方法：

Plotter.\_\_init\_\_(dat, stored, core\_num):

初始化plotter对象，并创建Processor对象。

Plotter.plot(plane, lev, var, axes, show\_cbar):

根据参数绘制热力图。

plane: 整数类型 绘制的切面 (0: Y-Z, 1: X-Z, 2: X-Y):

lev: 整数类型 第三维度数值。

var: 字符串 要绘制的物理量名称。

axes: matplotlib绘图对象。

show\_cbar: 布尔逻辑 是否展示数值条。

Plotter.interactive\_plot (para1, para2, para3, para4):

Para1,2,3,4: 元组 包含绘制每一幅图所需的plane, lev, var 数值。

Processor数据处理模块

模块功能

根据用户选择的绘图参数，处理并计算绘图所需的网格点的数据。

接口关系：

接受参数(dat, stored, core\_num):

字符串 文件路径 dat ;

字符串 文件路径 stored ；

整数类型 多线程处理器数量 core\_num；

内部方法：

Processor.\_\_init\_\_(dat, stored, core\_num):

初始化Processor对象。

Processor.\_\_repr\_\_():

返回映射数据行列数。

Processor.init\_in\_matrix(core\_num):

筛选file文件里符合绘图范围的数据点，赋值到processor.mapping中。

Core\_num: 多线程运行数量。

Processor.save\_in\_matrix(file):

将processor.mapping保存到本地。

File: 储存文件名。

Processor.load\_in\_matrix(file):

读取本地存储的映射文件。

File:读取文件名。

Processor.get\_df(plane, level, var, core\_num):

通过Processor.insert方法填补绘图数据表中空缺的物理量。

plane: 整数类型 绘制的切面 (0: Y-Z, 1: X-Z, 2: X-Y)。

lev: 整数类型 第三维度数值。

var: 字符串 要绘制的物理量名称。

core\_num: 多线程参数。

Processor.\_\_idw(idx, pt, var):

利用立方体idx和返距离加权法计算pt点的数据值。

Idx: 整数类型 pt点所在的空间立方体在all\_pos中的行号。

Pt: 元组 空间点坐标。

var: 字符串 要绘制的物理量名称。

Processor.insert(start, end, plane, level, var):

利用processor.\_idw计算空间上每一个点的物理值。

Start: 整数类型 绘图范围起点

End: 整数类型 绘图范围终点

plane: 整数类型 绘制的切面 (0: Y-Z, 1: X-Z, 2: X-Y)。

lev: 整数类型 第三维度数值

var: 字符串 要绘制的物理量名称。

返回：赋值后的映射数据表。

Util运算方法模块

模块功能：

辅助计算功能的集合。

接口关系：

Util.x\_to\_ind(x):

计算x在空间网格点中的坐标（整数）。

Util.x\_to\_flt\_ind(x):

计算x在空间网格点中的坐标（小数）。

Util.find\_points(c):

计算一个空间立方体三个维度的最大值和最小值。

C: 包含空间立方体八个顶点的表格行。

Util.combine\_3d(results):

筛选results中符合条件的非-1值。

返回: init 三维数组。

Util.combine\_2d(results):

筛选results中符合条件的非0值。

返回：init 二维数组。

Util.presort(df, df\_pos):

筛选8个顶点都在绘图范围内的立方体

Df: 三维坐标和物理量表格。

df\_pos: 空间坐标关系表

返回: temp 三维数组包含立方体序号

Util.dat2df(path):

将.dat文件读取成两个DataFrame。

Path: 字符串 文件路径

返回: df\_point，df\_pos DataFrame

# 3.执行过程

3.2 输入

用户在运行软件前应确保数据表格被存放在main.py运行路径上的 “./data”文档中。数据应该包括两个文件points.csv和all\_pos.csv。其中points.csv应该包括三位坐标及其对应的物理量; all\_pos.csv应该包括空间中的坐标关系,应有9列-第1列：行号；2至9列：顶点在points中的行号。

在再次运行前，如果用户想要利用预处理后数据来加速绘图过程，应确保在main.py运行路径上的 “./data”文档中有stored.npy文件。

在通过图形界面输入绘图参数时，每一幅热力图左边的两个选择器中应有且只有一个勾选值。每一幅热力图左边的第三维度选择文本框中应输入范围在正负3之间的整数或小数。

* 1. 启动及初始化状态

当用户初次运行软件时，程序会对存储在运行路径中名为data文件里的物理数据进行长约10分钟的预处理。在随后的执行过程中，程序会利用存在运行路径中的预处理文件来加快图像绘制过程。默认的多线程处理器数量为4，运行前应根据运行环境相应增加或减少。详细的初始默认绘图流程如下：

Diagram

Description automatically generated

*2.数据处理流程图*

1. 读取原始数据

程序开始运行，Main读取物理量,坐标关系数据(.csv格式)。

1. 创建相关对象

Main创建绘图模块Plotter对象，随后Plotter在初始化时创建自己的数据处理模块Processor对象。

1. 预处理文件

Processor查看本地预处理文件是否存在。如不存在进入4，如存在进入5。预处理文件会在第一次运行后被保存在本地。

1. 读取数据

如不存在，Processor读取物理量数据，通过Util中的方法筛选在热力图所需数据范围内的数据点。

* 1. 空间关系映射

建立热力图数据点与对应空间上网格点的映射。

1. 调用映射

如存在，Processor直接读取之前建立的映射关系文件。

1. 空缺数据

绘图数据准备阶段，Plotter根据默认值绘制热力图。

1. 插值计算

Processor遍历既定层所有的数据点,根据映射找到对应网格点，应用插值计算Inverse Distance Weighting(IDW)方法计算空缺数据点的物理值。

1. 图形界面

Plotter根据计算后完整的数据表绘制热力图及其他界面按钮。

* 1. 终止及输出

用户应在任意绘图过程结束后通过图形界面的关闭窗口按钮结束程序。绘图过程的输出是2-D热力图，展示物理量大小在绘图范围内的分布。在绘图运行过程中，停留在图形界面范围内的光标应显示为加载状态，切换选择器选项应不会产生变化。首次运行时用户应期待12分钟左右的预处理时间，随后的绘图过程时间为1分钟左右。

1. 操作流程

在操作系统的terminal中运行main.py文件 -> 等待默认热力图绘制，图形界面弹出 -> 绘图结束后可以选择本地保存绘图结果 -> 通过每个热力图左侧的选择器和文本框重新选择绘图参数 -> 点击Process按钮重新绘制 -> 处理时间约1分钟 -> 查看更新后的绘图结果 -> 关闭图形界面来结束程序