

# 程序运行环境说明文档

## 1. 本程序解决的问题说明

本程序针对 AIS 船舶轨迹数据残缺问题提供了解决方案。由于海洋环境复杂多变以及数据采集设备限制，实际获取的 AIS 轨迹数据经常存在不同程度的缺失（[研究报告.pdf](#)）。这些数据缺失给航运管理、港口调度和船舶 ETA（预计到港时间）预测等带来了极大困难（[研究报告.pdf](#)）。为应对这一挑战，本项目提出了一种基于子序列动态时间规整（Subsequence DTW）引导机制的智能轨迹补全方法（[研究报告.pdf](#)）。核心思想是在数据预处理中提升轨迹数据质量，随后利用历史完整轨迹模拟残缺样本进行模型训练，最后通过深度学习模型对残缺轨迹进行智能补全。该方法通过引入 LSTM 长短期记忆网络结合 DTW 引导信息，提高了对轨迹趋势和结构的捕捉能力，并通过贝叶斯优化自动调节模型超参数，以提升预测性能和模型稳定性（[研究报告.pdf](#)）。

简而言之，本程序旨在重建缺失的船舶轨迹：输入一段不完整的船舶轨迹，模型将基于历史相似轨迹对其进行补全，输出与真实轨迹走势相符的完整轨迹。这一能力对改进航运预测的准确性和可靠性具有重要意义（[研究报告.pdf](#)）。

## 2. 整体架构设计说明

本项目采用模块化的架构设计，将轨迹补全过程划分为若干阶段，形成完整的工程流程闭环。整个流程可概括为以下四个阶段：

- 数据预处理与分类：**对原始 AIS 轨迹数据进行结构解析、清洗和异常值修复，确保数据质量，并根据轨迹长度和起讫港口对完整轨迹样本进行聚类分型，划分为 8 类轨迹子集。这一步为后续模型训练构建多源异构的高质量训练样本集（[研究报告.pdf](#)）。
- 模型补全训练：**针对每一类轨迹数据，生成对应的“残缺片段-目标完整”样本对，用于模型训练。训练采用统一的 LSTM 神经网络结构，并融入子序列 DTW 匹配的引导机制（[研究报告.pdf](#)）。通过将残缺轨迹与从历史库中匹配的最相似轨迹段（引导轨迹）共同作为模型输入，网络学习输出完整轨迹的能力（[研究报告.pdf](#)）。
- 超参数优化：**为提升不同类别轨迹的补全效果，对模型训练的关键超参数引入贝叶斯优化策略逐类调优（[研究报告.pdf](#)）。每个类别都有独立的超参数搜索空间和优化过程，目标是最大化轨迹补全的综合评分（如 F1 得分）或最小化预测误差。贝叶斯优化迭代搜索得到各类别的最优参数组合，提高模型的预测精度和稳定性（[研究报告.pdf](#)）。
- 部署与应用：**将每个类别得到的最优超参数配置和训练得到的模型权重记录保存。在实际应用时，先对输入的残缺轨迹进行所属类别判别，然后快速调用对应类别的最优模型或参数配置进行补全。这一设计实现了不同类别轨迹的差异化补全效果，保证模型在真实场景中的鲁棒性和泛化能力（[研究报告.pdf](#)）。

通过上述架构设计，程序从数据处理、模型训练到实际部署形成了完整链条。各模块既相对独立又前后衔接，方便根据需要替换或调整组件。例如，可替换不同的深度学习模型模块，或对预处理模块升级，而不会影响整体流程的其他部分。这种模块化设计提高了程序的可维护性和可拓展性。

### 3. 数据流向说明

本程序的数据流程按照上述阶段有序进行，以下以步骤形式描述数据在系统中的流向：

- 原始数据导入与清洗：**首先将原始 AIS 轨迹数据读入程序。经过数据清洗（去除重复记录、检测并修复异常值、插值补全缺失片段）和结构标准化后，输出干净且格式统一的轨迹样本数据集（[研究报告.pdf](#)）。这一步的结果可以保存为清洗后的数据文件（例如 AIS 信息数据训练集\_处理完成\_TEST.csv），供后续使用。
- 轨迹分类与分组：**利用清洗后的完整轨迹样本，根据每条轨迹的步长长度范围以及起始-终止港口对进行聚类分类。分类输出将轨迹划分为预定的 8 个类别，每个类别对应相似的轨迹长度范围和航线性质。分类结果以类别标签的形式附加到轨迹数据，并可根据类别将数据拆分成多个类别。
- 残缺轨迹模拟生成：**针对每一类别的完整轨迹子集，模拟生成残缺轨迹样本对用于模型训练（[研究报告.pdf](#)）。具体方法是从完整轨迹中按照一定比例截断前段和后段，从而得到“残缺轨迹片段”，并将原完整轨迹作为目标进行对比学习。每条完整轨迹可生成对应的残缺版本，形成残缺-目标轨迹对。这一步产生的训练数据可以在内存中使用，或另行保存（例如每类各自的训练对文件）。
- 引导轨迹匹配与特征构建：**在将残缺轨迹输入模型之前，从历史轨迹库（训练集中本类别的其他完整轨迹）中通过子序列 DTW 算法匹配出与该残缺片段最相似的一段轨迹，作为“引导轨迹”（[研究报告.pdf](#)）。然后将“残缺轨迹 + 引导轨迹”拼接形成模型输入的特征序列，并进行必要的规范化（如 Z-score 标准化）（[研究报告.pdf](#)）。这样确保模型输入既包含当前轨迹的已知部分，又包含历史参考的趋势信息。
- LSTM 模型训练与预测：**将第 4 步得到的特征序列输入 LSTM 网络进行训练。模型以序列数据为输入，经过若干层 LSTM 单元提取时序特征，最终输出预测的轨迹点序列（[研究报告.pdf](#)）。训练过程中，模型的输出与真实完整轨迹（目标）进行比较，计算误差来调整网络权重。由于采用多步预测输出结构，模型一次前向传播即可生成残缺段后续若干步的轨迹点（[研究报告.pdf](#)）。针对每个轨迹类别，重复此训练过程，获得初步的补全模型。
- 贝叶斯超参数优化：**在模型训练基础上，引入贝叶斯优化循环调整关键训练超参数，提高模型性能（[研究报告.pdf](#)）。每类轨迹都有独立的优化过程：定义超参数搜索空间（如残缺截断比例、学习率、训练轮数等），以补全结果的评价指标为目标函数，不断迭代评估不同参数组合的效果。贝叶斯优化在参数空间中选择下一个评估点时考虑了以往结果，从而高效逼近最优值。经过若干次迭代后，输出每个类别的最优超参数配置，以及在该配置下训练得到的最佳模型。
- 模型评估与结果可视化：**在验证集或测试集上评估每个类别模型的补全效果。使用预先定义的指标体系（例如轨迹 F1 补全得分、DTW 距离等）对预测轨迹与真实轨迹的吻合度进行量化评估（[研究报告.pdf](#)）。评估结果可以通过二维表格或三维可视化图展示，以分析不同类别下模型性能的差异和趋势。例如，不同轨迹长度类别的 F1 得分分布，以及 DTW 相似度与预测准确性的关系。
- 实际部署应用：**在实际使用中，对于任意输入的一条残缺轨迹，首先根据其特征（长度、港口等）判断所属类别（可利用与训练时相同的分类标准）。然后调用该类别预先训练好的最优模型，对残缺轨迹进行补全预测，输出完整轨迹。最终结果可以存储或用于后续业务分析，例如更新船舶 ETA 预估或进行航线规划。整个流程确保了从原始数据到补全结果的数据流畅通，每个阶段的产出都为下一阶段提供了所需输入，实现了数据的逐级加工与价值提升。

上述数据流向反映了程序内部数据处理的先后顺序和依赖关系。评委在复现时应按照此流程依次执行相关步骤，以确保前一阶段的数据正确提供给下一阶段使用。

## 4. 各功能模块说明

本程序由多个功能模块组成，每个模块承担特定功能，相互之间通过数据接口衔接，形成完整的轨迹补全系统。以下对各模块的功能及其逻辑关系进行说明：

- **AIS 数据预处理模块：**该模块负责读入原始 AIS 轨迹数据并进行清洗和标准化处理。其功能包括：解析 AIS 数据结构（提取经度、纬度、时间戳、航速、航向等字段），去除重复或不合理的数据点，检测异常值并采用插值等方法进行修复（[研究报告.pdf](#)）。此外，还对数据进行规范化处理（如采用统一的字段格式和单位），以构建高质量的轨迹训练样本（[研究报告.pdf](#)）。经过此模块处理后，输出的轨迹数据具有一致的结构和较高的可靠性，为后续分析奠定基础。**逻辑关系：**此模块输出“干净的完整轨迹数据”，作为轨迹分类模块的输入。
- **轨迹分类模块：**此模块将预处理后的完整轨迹按照预定规则进行聚类分类。分类依据包括**轨迹步长（点数或持续时间）范围**以及**航线的起始港口和目的港口组合**。通过聚类分析，程序将轨迹划分为 8 个类别子集，每一类包含相似长度范围和相似航线特征的轨迹数据。这样做的目的是针对不同类型的轨迹分别建模，提高模型对各类别轨迹的针对性和泛化能力。**逻辑关系：**分类模块输出的每条轨迹的类别标签以及分类后的数据子集，将用于残缺轨迹生成和模型训练模块。其中类别标签也用于实际应用时对新轨迹选择相应模型。
- **残缺轨迹生成模块：**该模块针对每一类别的完整轨迹样本，模拟生成训练用的残缺轨迹数据。具体来说，程序会按照一定比例从完整轨迹的开头和结尾截取掉一部分，从中抽取中间残留的片段，作为“残缺轨迹”样本。被截去的部分相当于轨迹缺失的片段，而原完整轨迹则作为学习目标。这一模块通过**自监督学习**的思路构建了残缺-目标轨迹对，用于训练时让模型学会从前半段数据预测后半段轨迹（[研究报告.pdf](#)）。在生成过程中，不同轨迹类别可以采用不同的截取比例范围，以模拟各类别常见的缺失情况（这些比例范围在贝叶斯优化时会被进一步调整）。**逻辑关系：**本模块输出的残缺轨迹片段及对应目标完整轨迹，将作为 LSTM 模型训练模块的输入数据。
- **引导轨迹匹配与特征整合模块：**在将残缺轨迹输入 LSTM 模型之前，程序增加了一个引导匹配步骤。该模块通过**子序列 DTW（动态时间规整）算法**，在历史轨迹库中寻找与当前残缺轨迹**最相似的一段轨迹片段**（[研究报告.pdf](#)）。这个最相似片段被称为“引导轨迹”，代表了历史上与当前残缺片段走势相近的模式。找到引导轨迹后，将其与当前残缺轨迹拼接或并行作为模型的输入特征序列，经标准化处理后送入 LSTM 网络（[研究报告.pdf](#)）。通过引入引导轨迹，模型在预测时可参考历史相似案例，从而更准确地恢复残缺部分的轨迹走向。**逻辑关系：**该模块实质上丰富了 LSTM 模型的输入信息，其输出（残缺轨迹+引导轨迹的特征序列）直接进入下一模块的 LSTM 网络。
- **LSTM 轨迹补全模型模块：**这是整个系统的核心模块，采用\*\*轻量级长短期记忆网络（LSTM）\*\*实现轨迹序列的预测补全（[研究报告.pdf](#)）。模型输入为上一步整合后的特征序列，首先经过若干层堆叠的 LSTM 单元以提取时间序列中的动态模式特征，然后通过全连接层输出多步预测结果（[研究报告.pdf](#)）。输出的多步序列即为对残缺轨迹后续部分的预测位置序列，经过与引导轨迹及已知轨迹片段的时间对齐和插值处理，形成完整的轨迹补全结果（[研究报告.pdf](#)）。该模块的设计充分考虑了轨迹数据的时空相关性，利用 LSTM 的记忆能力来捕捉航迹走势。同时通过引导机制增强了对轨迹结构的感知，避免模型仅凭残缺片段进行盲目外推。**逻辑关系：**在训练阶段，LSTM 模块输出的预测序列将与真实完整轨迹进行比较用于计算损失；在应用阶段，LSTM 模块接收实际残缺轨迹（以及选取的引导轨迹）产生最终补全结果。
- **贝叶斯优化超参数模块：**为了确保上述 LSTM 模型在各轨迹类别上都达到最佳效果，程序引入贝叶斯优化模块对模型的关键训练超参数进行调节（[研究报告.pdf](#)）。该模块定义了一个超参数空间（例如：残缺截断起止比例、初始学习率、最大训练轮数等），并以轨迹补全效果评分作为目标函数。优化过程会迭代地选择参数组合运行模型训练（或微调），然后根据得到的



评分更新内部高斯过程模型，再选择下一组候选参数评估。与网格搜索或随机搜索相比，贝叶斯优化能更高效地找到接近全局最优的参数组合 ([研究报告.pdf](#))。本程序为每个轨迹类别单独运行优化，以便针对每类数据调整最优参数。**逻辑关系：**贝叶斯优化模块实际上包裹了 LSTM 模型训练过程，对其进行多次调用以评估不同参数配置。优化结束后，将最优参数用于最终模型训练，并将这些参数记录供部署使用。

- **模型评估与可视化模块：**该模块用于综合评估模型在轨迹补全任务中的表现，并以直观方式呈现结果。主要功能包括：计算各类别模型的预测误差指标（如 F1 补全得分、DTW 分数等）以及 DTW 轨迹相似度指标，生成评估表格；对部分补全结果进行可视化，将真实轨迹与模型预测轨迹绘制对比图；绘制评价指标的统计图或分布图，以分析模型性能随轨迹类别的变化趋势。例如，可生成三维散点图展示每类轨迹的 DTW 距离与 F1 得分之间的关系 ([研究报告.pdf](#))。**逻辑关系：**评估模块通常在模型训练完成后使用，其分析结果可反过来验证前面各模块（如预处理、分类、引导）的有效性，并为可能的改进提供依据。在完成模型评估后，最终确定模型配置进入部署阶段。
- **部署应用模块：**此模块负责将训练得到的模型应用于真实场景的残缺轨迹补全。部署时通常会加载先前保存的各类别最优模型参数或网络权重，然后等待外部输入的新轨迹数据。当有残缺轨迹输入时，模块首先按照与训练阶段相同的规则确定该轨迹所属类别，然后选择对应的 LSTM 模型及其参数进行补全计算。输出的完整轨迹可以进一步反馈给业务系统或可视化展示。该模块可能还包括一定的接口或用户交互代码，以方便评委或用户输入测试轨迹并获取结果。**逻辑关系：**部署模块使用前面所有模块的最终产出——模型和参数，对新的数据执行预测。因此它处于流程的末端，但对前序模块的正确执行和模型有效性高度依赖。

综上，各功能模块各司其职，依次联接形成完整流程。从原始数据到模型输出，每一步都有明确的模块处理，使程序结构清晰，职责分明。模块之间的数据依赖通过约定的中间结果（如标准化后的数据格式、分类标签、模型文件等）来传递。这种设计便于调试和扩展，例如评委在检查特定阶段效果时，可以针对某一模块独立运行验证，然后再连接到整体流程中。

## 5. 程序运行环境说明






为确保上述脚本和模块能够顺利运行，评委需要准备与开发环境一致的运行环境配置。具体说明如下：

- **MATLAB 版本要求：**本程序需在 **MATLAB R2023a** 环境下运行。建议使用 R2023a 或以上版本，以保证兼容性和获得最新功能支持。本项目开发和测试均基于 MATLAB R2023a，使用其他版本（尤其是较旧版本）可能会遇到函数不可用或行为差异的问题。
- **依赖工具箱：**请确保已安装以下 MATLAB 工具箱组件：
  - **Deep Learning Toolbox**（深度学习工具箱）：用于构建和训练 LSTM 神经网络。本程序的轨迹补全模型基于 LSTM 实现，需要该工具箱提供的 `sequenceInputLayer`, `lstmLayer`, `fullyConnectedLayer`, `trainingOptions`, `trainNetwork` 等函数。
  - **Statistics and Machine Learning Toolbox**（统计与机器学习工具箱）：用于执行贝叶斯优化。本程序调用 `bayesopt` 函数来自动调节超参数 ([研究报告.pdf](#))。该函数隶属于统计与机器学习工具箱，因此需确保安装。
  - **Parallel Computing Toolbox**（并行计算工具箱）：加速预测结果生成的效率，需要启用并行计算工具箱。程序中可能包含对并行池的支持（例如使用 `parpool` 或在 `bayesopt` 中设置并行选项），这在多核 CPU 环境下可提高优化效率。
  - **其他工具箱：**若脚本涉及地理坐标处理或地图可视化（例如绘制全球航迹图等），需要 **Mapping Toolbox**（地图工具箱）或 MATLAB 地理坐标函数支持。如果仅关注核心

补全功能，可以不安装此可选组件，但缺少时某些可视化功能可能无法使用。大部分通用绘图和数据处理操作无需额外工具箱即可运行。

- **操作系统兼容性：**MATLAB R2023a 支持 Windows、Linux、MacOS 等操作系统。本程序的代码未使用特定于操作系统的调用，因此在上述任一系统上均可运行。请注意在不同操作系统上，文件路径表示可能略有差异（如 Windows 使用反斜杠\，Linux/Mac 使用斜杠/）。
- **文件路径与组织结构：**为方便复现，需按照提交的代码和数据约定组织文件目录。建议将所有代码和数据文件分别放置在一个文件夹下，如图所示：

文件夹 1（如“支撑材料→代码运行所需数据”中所示）

名称	修改日期
 AIS信息数据测试集_处理完成_TEST	2025/4/2 2:58
 AIS信息数据训练集_处理完成_TEST	2025/4/4 20:41
 船舶AIS信息数据测试集	2025/1/15 15:08
 船舶AIS信息数据训练集	2025/3/20 21:30
 港口静态信息数据	2025/3/17 18:45

文件夹 2（如“支撑材料→源代码(.mlx)”中所示）

名称	修改日期
 BYS_LSTM_1	2025/4/7 2:06
 BYS_Prepare	2025/4/4 19:08
 Geo_5	2025/4/4 21:42
 LSTM_PAR	2025/4/9 19:19
 LSTM_ZHEN	2025/4/4 9:51
 TEST_END	2025/4/9 18:16
 TRAIN_END	2025/4/9 18:12

- **路径设置：**在运行 MATLAB 脚本前，请将 MATLAB 当前工作路径设定到项目主文件夹下（或直接将项目加入 MATLAB 路径）。这样脚本中使用的相对路径（如 data/xxx.mat）才能正确定位到文件。如果脚本未使用相对路径而是在开头定义了数据文件的完整路径变量，请根据实际存放位置修改这些路径变量。例如，在脚本开头可能有：
- `dataDir = 'C:\Users\Name\TrajectoryCompletionProject\data\';`

需要将其修改为评委当前机器上实际的路径。如果采用与上述建议相同的目录结构，只需确保工作目录为项目根目录，脚本内部相对路径即可直接使用，无需修改。

- **硬件及性能要求：**本程序主要为轨迹数据的清洗和模型训练，计算量取决于数据规模和训练迭代次数。一般的现代 PC（具有 8GB 以上内存，双核及以上 CPU）即可运行小规模示例。如

果数据集较大或 LSTM 模型较复杂，训练过程可能较耗时，建议在有 GPU 的计算机上运行以加速深度学习训练（需要安装支持的 CUDA 库以及 MATLAB Parallel Computing Toolbox 和 GPU 支持）。但对于评委复现而言，可使用子集数据或减少训练轮数来快速验证流程正确性。总之，请保证运行过程中有足够的内存来载入数据（例如数百万条 AIS 记录可能需要数 GB 内存），并耐心等待模型训练和优化完成。

- **MATLAB 交互设置：**由于脚本为 Live Script (.mlx) 格式，建议在 MATLAB 桌面界面中打开这些脚本以便查看文字说明、公式和可视化输出。在其他设备上复现时，请确保 MATLAB 环境支持 Live Script (R2023a 默认支持)。评委可以选择逐段运行脚本以查看中间结果和注释说明。

## 6. 操作指南

下面可以按照序号的顺序运行，逐一按顺序说明每个 .mlx 脚本的功能和使用方法。（也可以直接跳转到 **5.实例脚本：LSTM\_PAR.mlx** 参照对应操作指南直接运行“模型核心程序”输出‘预测结果.csv’文件，其余皆为参数/数据准备与其余功能的辅助代码）

### 1.实时脚本：TRAIN\_END.mlx

#### 功能描述：

该实时脚本文件用于对 AIS 船舶训练集数据进行预处理与清洗，从而生成标准化的训练样本数据。主要工作内容包括：

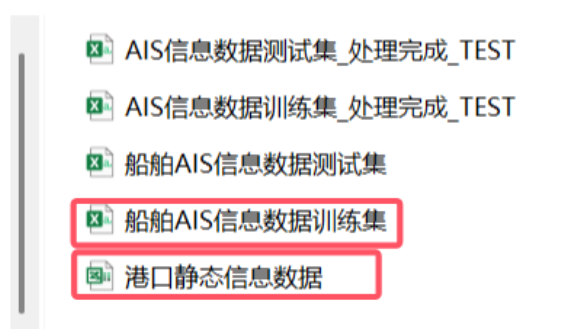
1. 环境初始化：
  - 清空变量和命令窗口，显示处理开始信息，并启动计时器。
2. 数据导入：
  - 从指定路径导入船舶 AIS 训练集数据（CSV 格式）以及港口静态信息数据（XLS 格式）。
3. 重复记录剔除：
  - 针对每艘船，通过船名(ship\_name)和时间(slice\_time)字段进行去重，只保留首次出现的记录，并输出剔除重复记录的数量。
4. 机器学习辅助校验（步骤 0）：
  - 利用决策树模型对原始数据进行初步评估。选取对地速度、纬度、经度作为特征，执行十折交叉验证，计算分类准确率，用以辅助检查数据的合理性（例如状态 status 字段中非 NaN 数据的有效性）。
5. 滑动窗口异常值检测与处理（步骤 1）：
  - 对连续变量（船速 sog、航迹向 cog、吃水 draught、经纬度）采用自适应滑动窗口技术，结合 modified Z-score 或 4IQR 法检测异常值。
  - 对检测出的异常点进行趋势线性插值修复，确保连续变量平滑合理。
6. 状态字段修正（步骤 2）：
  - 基于处理后的 sog 和与港口的距离，修正船舶状态 status。如果速度低且靠近港口，则设为靠泊状态（5），否则设为锚泊（1）或航行（0）。
7. 填充港口编码（步骤 3）：

- 针对 leg\_end\_port\_code 缺失或错误的记录，采用与港口表中最近港口匹配的方式，填充正确的港口编码。
- 8. 航段划分及尾部状态调整（步骤 4）：
  - 基于填充后的 leg\_end\_port\_code 重新划分航段，并在航段尾部调整连续 status 为 1 的记录，根据离港与靠港距离规则进行修改。
- 9. 航段内状态统一调整（步骤 5）：
  - 分组调整同一航段内的 status，解决同一航段内多组连续 5 的情况，通过循环将不合理的连续记录修正。
- 10. 生成起始港口与航迹编号（步骤 6）：
  - 根据每艘船轨迹的时间顺序，生成出发港口 leg\_begin\_port\_code 和新的连续航段编号 trip\_id。
- 11. 吃水异常修正（步骤 7）：
  - 对每个航段的吃水数据应用 4IQR 法进行异常检测和修正，保障吃水数据的合理性。
- 12. 经纬度异常修正（步骤 8）：
  - 针对每个航段独立，对经纬度数据采用滑动窗口结合 4IQR 法检测异常，并通过联合线性插值修正。
- 13. 航段内状态进一步调整（新增步骤 9/10）：
  - 依据航段编号重新调整各航段内的 status，处理跨航段的不一致问题。
- 14. 首尾航段删除与航段重新排序（步骤 10）：
  - 移除每艘船的首尾航段（或当航段数不足时全部删除），并对剩余航段重新按顺序编号。
- 15. 机器学习辅助校验（步骤 11）：
  - 再次使用决策树模型对处理后的数据进行校验，确保经过预处理后的数据具有较高的状态预测准确率，验证数据修正的有效性。
- 16. 最终数据输出（步骤 12）：
  - 根据修正后的各项变量拼装生成最终数据表，并按指定字段顺序输出至 CSV 文件，以便后续使用。
- 17. 特征热力图绘制（步骤 13）：
  - 计算并绘制主要连续变量之间的相关系数热力图，用自定义颜色映射展示低值为白、高值为红，并在图中标注具体数值。

## 使用方法：

### 前置准备：

1. 如图所示位置请确保修改脚本中数据路径，正确读取赛题提供的“训练集”与“港口信息”数据，（支撑材料→代码运行所需数据）（代码 8、9 行）





```

6    %% 导入原始数据
7    disp('>> 正在导入原始数据...');
8    AIS_Data = readtable('C:\Users\幻15\Desktop\赛题数据\船舶AIS信息数据训练集.csv');
9    Port_Data = readtable('C:\Users\幻15\Desktop\赛题数据\港口静态信息数据.xls');

```

2. 修改代码结尾的写入文件路径，却保输出处理后的“AIS 信息数据训练集\_处理完成\_TEST.csv”文件写入正确文件夹，方便后续调用（代码 699 行）

```






699  output_folder = 'C:\Users\幻15\Desktop\Result';
700  if ~exist(output_folder, 'dir')
701      mkdir(output_folder);
702  end
703  output_file = fullfile(output_folder, 'AIS信息数据训练集_处理完成_TEST.csv');

```

## • 运行流程：

1. 打开脚本：在 MATLAB R2023a 环境下，打开 TRAIN\_END.mlx。

## 2. 主要输出：

名称	修改日期
 AIS信息数据测试集_处理完成_TEST	2025/4/2 2:58
 AIS信息数据训练集_处理完成_TEST	2025/4/4 20:41
 船舶AIS信息数据测试集	2025/1/15 15:08
 船舶AIS信息数据训练集	2025/3/20 21:30
 港口静态信息数据	2025/3/17 18:45

## • 注意事项：

- 运行过程中，建议保持 MATLAB 会话不断开，以便后续脚本调用预处理结果；
- 输出 CSV 文件为标准化后的训练集数据，确保字段顺序与需求一

## 2.实时脚本：TEST\_END.mlx

### 功能描述：

该实时脚本文件主要用于对 **AIS 船舶测试集数据** 进行预处理与清洗，其目标是生成与训练集格式一致的标准化测试数据，便于后续模型测试和比较。由于测试集数据与训练集略有差异（例如测试集缺少 leg\_end\_port\_code 字段），因此该脚本在处理过程中采用了基于时间间隔的航段划分方法，而非基于港口编码变化的划分。总体上，TEST\_END.mlx 完成以下主要任务：

#### 1. 环境初始化

- 清空工作空间、命令窗口，并启动计时器，提示测试集数据处理开始。

#### 2. 导入测试集数据与港口静态数据



- 从指定路径读取测试集 AIS 数据（CSV 格式）和港口信息数据（XLS 格式）。
- 3. **重复记录剔除**
  - 依据船舶编号（ship\_name）与时间戳（slice\_time）去除重复记录，只保留第一次出现的记录，同时输出剔除记录的数量。
- 4. **基于时间间隔划分航段（新增步骤 1）**
  - 由于测试集没有 leg\_end\_port\_code 字段，故采用时间间隔划分法。设定时间间隔阈值为 4 小时，对每艘船的记录按时间排序，若连续记录之间的时间差大于 4 小时，则认为存在断点，将数据分为不同的航段，并为每个航段生成连续的航段编号（trip\_id）。
- 5. **滑动窗口异常值检测与处理（步骤 1）**
  - 对测试集中的连续变量（对地速度 sog、航迹向 cog、吃水 draught 以及经纬度）进行异常值检测和修复。采用自适应滑动窗口技术，利用 modified Z-score 方法对 sog、cog 和 draught 进行检测和线性插值修复；而对经纬度数据则结合 4IQR 法采用联合插值处理，确保每个航段内数据的平滑和一致性。
- 6. **status 列校正（步骤 2）**
  - 根据经过异常处理后的 sog 值和船舶与港口之间的距离，修正状态字段（status）。如果船速低于 0.5 节且距离最近港口在 30 公里以内，则将状态设为靠泊（5）；低速但距离较远则设为锚泊（1）；船速高则设为航行状态（0）。处理过程中，每 5000 条记录会打印一次处理进度。
- 7. **生成船舶出发港口信息（新增步骤 2）**
  - 由于测试集数据中缺乏 leg\_end\_port\_code，本步骤通过取每个航段首个记录的位置，根据与港口的距离匹配最近港口，生成出发港口（leg\_begin\_port\_code）列，用以保持输出数据的字段格式与训练集一致。
- 8. **吃水异常值修正（步骤 3）**
  - 针对每个航段的吃水（draught）数据，使用 4IQR 方法检测异常值，并用中位数进行修正，保证吃水数据的稳定性。
- 9. **经纬度异常值修正（步骤 4）**
  - 对每个航段中的经纬度数据，采用滑动窗口结合 4IQR 方法进行异常值检测，利用联合线性插值修正异常点，确保航迹位置数据的连续性和准确性。
- 10. **机器学习辅助校验（步骤 5）**
  - 利用决策树分类模型对处理后的测试数据进行辅助校验，选取 sog、纬度和经度作为特征，计算 10 折交叉验证准确率，以验证数据预处理质量。
- 11. **输出最终数据（步骤 6）**
  - 组装处理后的测试集数据。由于测试集原始数据没有 leg\_end\_port\_code，本步骤中补充空字符串作为该列，保留生成的 leg\_begin\_port\_code 和 trip\_id，调整字段顺序，确保输出文件格式与训练集一致。最终将处理后的数据输出为 CSV 文件（例如保存为“AIS 信息数据测试集\_处理完成\_TEST.csv”），供后续模型测试和比较使用。

## 使用方法：

### • 前置准备：

1. 如图所示位置请确保修改脚本中数据路径，正确读取赛题提供的“测试集”与“港口信息”数据，（支撑材料→代码运行所需数据）（代码 8、9 行）

名称	修改日期
AIS信息数据测试集_处理完成_TEST	2025/4/2 2:58
AIS信息数据训练集_处理完成_TEST	2025/4/4 20:41
船舶AIS信息数据测试集	2025/1/15 15:08
船舶AIS信息数据训练集	2025/3/20 21:30
港口静态信息数据	2025/3/17 18:45

```

6 %% 导入测试集原始数据与港口静态信息
7 disp(' >> 正在导入测试集原始数据... ');
8 AIS_Test = readtable('C:\Users\幻15\Desktop\赛题数据 - 副本\船舶AIS信息数据测试集.csv');
9 Port_Data = readtable('C:\Users\幻15\Desktop\赛题数据 - 副本\港口静态信息数据.xls');

```

2. 修改代码结尾的写入文件路径，却保输出处理后的“AIS 信息数据测试集\_处理完成\_TEST.csv”文件写入正确文件夹，方便后续调用（代码 348 行）

```

347 % 指定输出目录与文件名
348 output_folder = 'C:\Users\幻15\Desktop\Result';
349 if ~exist(output_folder, 'dir')
350     mkdir(output_folder);
351 end
352 output_file = fullfile(output_folder, 'AIS信息数据测试集_处理完成_TEST.csv');
353 disp([' >> 正在输出处理后的数据到文件: ', output_file]);

```

3. 最终输出.csv 示例（支撑材料中已提供）：

## 运行流程：

1. 打开脚本：在 MATLAB R2023a 环境下，打开 TEST\_END.mlx。

2. 主要输出：

AIS信息数据测试集_处理完成_TEST	2025/4/2 2:58
AIS信息数据训练集_处理完成_TEST	2025/4/4 20:41
船舶AIS信息数据测试集	2025/1/15 15:08
船舶AIS信息数据训练集	2025/3/20 21:30
港口静态信息数据	2025/3/17 18:45

## 注意事项：

- 运行过程中，建议保持 MATLAB 会话不断开，以便后续脚本调用预处理结果；
- 输出 CSV 文件为标准化后的训练集数据，确保字段顺序与需求一

### 3. 实例脚本：BYS\_Prepair.mlx






#### 功能描述：

- 1. 读取数据并将航段按步长分 8 类
  - 加载由前序脚本处理后的训练集文件（默认路径 AIS 信息数据训练集\_处理完成\_TEST.csv）。
  - 按 trip\_id 为每个航段计算步长（即每段数据行数），再利用分位数法将所有航段划分为 8 个类别，使不同长度范围的航段分布更均衡。
  - 脚本会输出这 8 类的步长统计，包括类别号、步长范围、平均步长以及对应的航段数量，便于后续分析或挑选典型航段。
- 2. 按港口对分组并筛选航段对
  - 从训练集中提取 trip\_id, leg\_begin\_port\_code, leg\_end\_port\_code 等信息，对航段进行分组统计（查看每种“起始港-目的港”组合的航段数）。
  - 根据新增功能，会在已划分的 8 类中筛选符合一定条件的航段对，并输出 ship\_name 与对应的中位步长等信息。该功能可用于快速定位有代表性或可比性的航段组合。
- 3. 加载测试集数据用于可视化
  - 脚本示例性地加载测试集文件 AIS 信息数据测试集\_处理完成\_TEST.csv，同样统计其步长分布，以便和训练集做可视化比较（例如直方图叠加对比）。
  - 进一步输出训练集与测试集的总步长比例及平均步长，对比两者的差异。
- 4. 可视化输出：分布直方图与小提琴图
  - 绘制训练集与测试集的航段步长分布直方图，并提供叠加图形以直观展示两者差异。
  - 最后绘制每个类别的小提琴图（2 行×4 列子图），显示不同类别的航段步长分布形态、中位数及离群点情况，帮助快速了解各类整体特点。

#### 使用方法：

##### 前置准备：

- 1. 请在脚本开头确认并修改数据的读入路径（如图所示 trainFile = 'C:\Users\幻15\Desktop\Result\AIS 信息数据训练集\_处理完成\_TEST.csv'）（代码第 5 行）

名称	修改日期
 AIS信息数据测试集_处理完成_TEST	2025/4/2 2:58
 AIS信息数据训练集_处理完成_TEST	2025/4/4 20:41
 船舶AIS信息数据测试集	2025/1/15 15:08
 船舶AIS信息数据训练集	2025/3/20 21:30
 港口静态信息数据	2025/3/17 18:45

```
4 %% =====【1】读取数据及基于步长分8类（代码2部分）=====
5 trainFile = 'C:\Users\幻15\Desktop\Result\AIS信息数据训练集_处理完成_TEST.csv';
6 T_train = readtable(trainFile);
```

- 2. 同时确认读取测试集数据的路径是否正确（如 testFile = 'C:\Users\幻15\Desktop\Result\AIS 信息数据测试集\_处理完成\_TEST.csv'）（代码 166 行）

名称	修改日期
 AIS信息数据测试集_处理完成_TEST	2025/4/2 2:58
 AIS信息数据训练集_处理完成_TEST	2025/4/4 20:41
 船舶AIS信息数据测试集	2025/1/15 15:08
 船舶AIS信息数据训练集	2025/3/20 21:30
 港口静态信息数据	2025/3/17 18:45

```
165 %% =====【4】加载测试集数据，仅用于绘图 =====
166 testFile = 'C:\Users\715\Desktop\Result\AIS信息数据测试集_处理完成_TEST.csv';
167 I_test = readtable(testFile);
168
```

## • 运行流程:

1. 打开脚本: 在 MATLAB R2023a 环境下, 打开 BYS\_Prepare.mlx。
2. 查看输出:
  - 实时脚本内右侧窗口内将显示 **8 类步长统计结果、港口分组统计以及新增功能筛选出的航段对等文本信息**;
  - 以及右侧窗口还会弹出若干图形窗口, 包含训练集与测试集的直方图比较, 以及各类别小提琴图。
3. 后续衔接: 该脚本的输出(步长分类与可视化结果等)主要用于辅助分析数据特征或筛选对比航段, 不会修改或覆盖训练集数据本身。评委可根据结果进一步选择典型航段或进行更深入的对比分析。

## • 注意事项:

- 运行过程中, 建议保持 MATLAB 会话不断开。

---

## 4. 实例脚本: **BYS\_LSTM\_1.mlx**

### 功能描述:

1. 可调参数与数据加载
  - 脚本开头定义了训练数据文件路径 filePath、采样间隔 sampleInterval, 以及贝叶斯优化需要搜索的超参数初始范围(如 startRemovalPct、endRemovalPct、initialLearnRate、maxEpochs 等)。
  - 读取包含 12 个字段的 AIS 数据文件后, 针对 hdg(航向)字段额外计算 sin\_phase 与 cos\_phase, 完善角度特征表示。
2. 航段分类信息设置
  - 以 8 个类别为单位(内含不同的出发-目的港注释及具体航段编号和步长信息), 用于后续贝叶斯优化的迭代测试。
  - 每个类别包含两条示例航段(seg1、seg2), 脚本会对这两条航段的预测得分进行平均, 并以此作为优化目标。
3. 贝叶斯优化流程



- 逐个类别调用 bayesopt 执行优化。脚本中定义了 objectiveFunction，其中分别调用 runPrediction 函数对 seg1 与 seg2 执行预测，然后计算两条航段得分的平均值。
- 由于 Bayesopt 默认最小化目标函数，因此脚本取“负的平均最终评分”为目标值，不断迭代寻找可让评分最大的超参数组合。






#### 4. 输出最优结果

- 每个类别完成优化后，会在命令窗口打印该类别的最优参数、最佳评分等信息。
- 最终可将这些参数应用于后续的完整模型训练或实际轨迹补全任务。

## 使用方法：

### • 前置准备：

1. 请在脚本开头确认并修改数据的读入路径，以便正确读取 AIS 信息数据训练集\_处理完成\_TEST.csv 文件。（代码第 6 行）

名称	修改日期
 AIS信息数据测试集_处理完成_TEST	2025/4/2 2:58
 AIS信息数据训练集_处理完成_TEST	2025/4/4 20:41
 船舶AIS信息数据测试集	2025/1/15 15:08
 船舶AIS信息数据训练集	2025/3/20 21:30
 港口静态信息数据	2025/3/17 18:45

```

4      %% 可调参数定义（均放在代码前面，不影响后续变量传递）
5      % 文件路径（请根据实际情况修改路径）
6      filePath = 'C:\Users\幻15\Desktop\Result\AIS信息数据训练集_处理完成_TEST.csv';

```

2. 根据需要调整贝叶斯优化的迭代次数（即 numIterations）以及各参数的搜索范围（如 startRemovalPct  $\in$  [0.05, 0.2] 等），以适配实际的训练规模与计算资源。（代码中以调试完成，无需更改）
3. 确认 8 个 categories 数组中各航段的编号、步长等信息无误，并与当前使用的数据文件中的实际航段保持一致。（代码中以调试完成，无需更改）

### • 运行流程：

1. 打开脚本：在 MATLAB R2023a 环境下，打开 BYS\_LSTM\_1.mlx。
2. 执行脚本：
  - 循环遍历 8 个类别，通过 bayesopt 进行超参数搜索，分别输出每个类别的最佳参数与得分；
3. 查看输出：
  - 在实时脚本右侧窗口中，您将看到每个类别的迭代过程信息，最终打印出“最优参数”及“平均最终评分”。
4. 衔接后续：
  - 脚本输出仅在实时脚本中显示参数与评分，不会自动保存。请手动记录或在脚本中添加保存步骤。
  - 这些最优参数可应用于更大规模的模型训练脚本或实际轨迹补全任务中。

## 5. 实例脚本：LSTM\_PAR.mlx

### 功能描述：

- 1. 训练数据与残缺轨迹数据的预处理与加载
  - 脚本在开头部分配置了训练数据路径 filePath（处理后的完整航段数据），以及测试集路径 testFilePath（存储需要补全的残缺轨迹）。
  - 对训练数据和测试数据统一进行经纬度平移与相位编码处理，以保证后续模型输入字段的一致性。港口静态信息表（portData）也同样执行平移处理，便于距离计算时保持坐标系一致。
- 2. 并行循环遍历测试集航段
  - 该脚本使用 parfor 并行循环，将测试集中的每条航段（ship\_name, trip\_id 对应的记录）作为独立任务进行轨迹补全，提升处理速度。
  - 对每条航段先根据实际步长划分区间，选取不同超参数（如 startRemovalPct、endRemovalPct、initialLearnRate、maxEpochs），随后调用 runPrediction 函数进行残缺轨迹补全。
- 3. 残缺轨迹补全并结果合并
  - runPrediction 函数在每条航段内执行“全局最优匹配 + DTW 对齐 + LSTM 补全”的流程。脚本在对该航段进行插值修正后，将预测结果（矩阵形式）和原始残缺轨迹表合并，并转换回原始经纬度坐标（减去偏移量）。
  - 将所有航段的补全结果累积到一个 outputCell 中，最终在脚本结尾使用 vertcat 合并，并写入同一 CSV 文件，以便统一查看所有航段的完整预测轨迹。
- 4. 结果输出与下一步衔接
  - 脚本完成后，会在指定路径输出 AIS 补全预测结果\_End.csv，其中包含原始残缺部分和模型补全部分的记录，供后续可视化或业务分析使用。
  - 如果需要自定义结果文件或添加更多统计指标，可在脚本中相应修改输出路径或新增分析步骤。

### 使用方法：

#### 前置准备：

- 1. 确认脚本中训练集与测试集路径：如图所示，修改 filePath 和 testFilePath 字符串以正确读取 AIS 信息数据训练集\_处理完成\_TEST.csv 与 AIS 信息数据测试集\_处理完成\_TEST.csv 文件，同样更新港口静态信息表 portData 的文件路径，注意脚本对经纬度进行了统一偏移，需要保持一致。（参见代码“读取港口静态信息数据”部分）（代码第 6、24、33 行）

名称	修改日期
 AIS信息数据测试集_处理完成_TEST	2025/4/2 2:58
 AIS信息数据训练集_处理完成_TEST	2025/4/4 20:41
 船舶AIS信息数据测试集	2025/1/15 15:08
 船舶AIS信息数据训练集	2025/3/20 21:30
 港口静态信息数据	2025/3/17 18:45

```

5 % 训练数据文件路径（用于完整航段匹配及后续处理）
6 filePath = 'C:\Users\幻15\Desktop\Result\AIS信息数据训练集_处理完成_TEST.csv';

23 %% 新增：读取TEST_1.csv，作为残缺轨迹构造的数据来源，并获取其中所有航段信息
24 testFilePath = "C:\Users\幻15\Desktop\Result\AIS信息数据测试集_处理完成_TEST.csv";

32 %% 新增：预先读取港口静态信息数据
33 portData = readtable('C:\Users\幻15\Desktop\赛题数据 - 副本\港口静态信息数据.xls');

```

2. 改变写入“预测结果.csv”文件的输入路径，确保正确输出/写入预测结果（此处以“支撑材料”中的“预测结果.csv 文件为例”，随意一个“‘空’.csv”文件即可）（代码第 154 行）

```

152
153 %% 写入CSV文件（所有航段结果存入同一文件）
154 outputPath = 'C:\Users\幻15\Desktop\Result\预测结果.csv';
155 writetable(fullOutputTable, outputPath);
156 fprintf('\n所有预测结果已写入文件：%s\n', outputPath);
157

```

## 运行流程：

1. 打开脚本：在 MATLAB R2023a 环境下，打开 LSTM\_PAR.mlx。
2. 依次执行：
  - 脚本读取训练数据、测试集数据和港口表，并对其中的经纬度进行平移、相位编码；
  - 通过遍历每条测试航段，根据实际步长区间选定 LSTM 训练/预测的超参数（startRemovalPct, endRemovalPct 等），并在 parfor 循环中调用 runPrediction；
  - 将补全后的残缺轨迹与预测轨迹合并保存到 outputCell 中。
3. 查看输出：
  - 脚本执行完成后，会在实时脚本右侧窗口内打印各航段处理进度与警告信息（若测试集中出现无有效数据的情况）。
  - 最后在指定 outputPath 写出“预测结果.csv”文件，包含所有航段补全后的结果。（如下图所示）

名称	修改日期
代码运行所需数据	2025/4/9 18:13
源代码(.mlx)	2025/4/9 15:25
附件3_研究报告	2025/4/9 14:23
预测结果	2025/4/4 13:53

## 注意事项：

1. 若计算环境不可用，可将 parfor 改为普通 for 循环（需确保 Parallel Computing Toolbox 已安装才可并行，否则脚本无法启动并行池）（代码第 38 行）

```
47 %% 并行循环遍历TEST_1中的每个航段（基于 ship_name 与 trip_id）
48 parfor i = 1:numVoyages
49     shipID = uniqueVoyages.ship_name(i);
50     tripID = uniqueVoyages.trip_id(i);
51
```

## 6. 实例脚本：LSTM\_ZHEN.mlx

### 功能描述：






1. 可调参数与数据读取
  - 在开头通过 filePath、sampleInterval、offsetLon、offsetLat 等变量配置训练数据文件路径和坐标偏移量；随后分别载入训练数据与测试数据（进行经纬度平移与 hdg 正余弦编码），并读取港口静态信息表以支持距离计算与港口匹配。
2. 单航段示例处理（船舶 1，航段 1）
  - 脚本仅针对测试集中的 shipID = 1、tripID = 1 这一条航段演示轨迹补全全过程。先根据该航段的行数确定其实际步长，再根据不同的步长区间自动选用相应的最优参数组合（如 startRemovalPct、endRemovalPct、initialLearnRate、maxEpochs 等），方便快速验证某单条航段的补全效果。
3. 轨迹补全与后处理
  - 调用 runPrediction 函数，对指定航段执行完整的“残缺轨迹截取 + DTW 匹配 + LSTM 自监督训练 + 预测插值”等步骤，最终得到补全部分的坐标矩阵 Y\_pred\_adjusted。然后生成一个表格 predictedTable，与原始残缺段合并后得到完整的轨迹信息（并减去偏移量恢复原始坐标）。
  - 脚本末尾通过 drawCompletionFrames 函数绘制关键帧图像，示例性展示多个时间断点下的补全轨迹和残缺轨迹对比，让用户直观了解补全后的趋向。
4. 可视化与输出
  - 运行完毕后，将在 MATLAB 中弹出若干图窗，展示分段关键帧下残缺段（橙色）与预测轨迹（绿色）的合并效果，以及关键点衔接处的箭头指示，以方便评估补全质量和检查轨迹走势连贯度。

### 使用方法：

#### 前置准备：

1. 在脚本前若干行（如 filePath、testFilePath 等）处，修改训练集与测试集路径，以保证可正确读入 AIS 信息数据训练集\_处理完成\_TEST.csv 和 AIS 信息数据测试集\_处理完成\_TEST.csv 文件。（代码第 5、18、26 行）



名称	修改日期
 AIS信息数据测试集_处理完成_TEST	2025/4/2 2:58
 AIS信息数据训练集_处理完成_TEST	2025/4/4 20:41
 船舶AIS信息数据测试集	2025/1/15 15:08
 船舶AIS信息数据训练集	2025/3/20 21:30
 港口静态信息数据	2025/3/17 18:45

```

3
4 %% 可调参数定义
5 filePath = 'C:\Users\幻15\Desktop\Result\AIS信息数据训练集_处理完成_TEST.csv';
6 sampleInterval = 3;

17 %% 读取TEST数据
18 testFilePath = "C:\Users\幻15\Desktop\Result\AIS信息数据测试集_处理完成_TEST.csv";
19 testData = readtable(testFilePath);

25 %% 读取港口静态信息
26 portData = readtable('C:\Users\幻15\Desktop\赛题数据 - 副本\港口静态信息数据.xls');
27 portData.lon = portData.lon + offsetLon;
--

```

2. 如果需要绘制其余编号船的第 X 条航段，可在脚本中自行更改对应参数，shipID=1、tripID=1，如图所示（代码第 31、32 行）

```

30 %% >>>> 只处理船舶1航段1
31 shipID = 1;
32 tripID = 1;
33

```

## • 运行流程：

1. 打开脚本：在 MATLAB R2023a 中打开 LSTM\_ZHEN.mlx;
2. 执行脚本：
  - 读取训练数据与测试数据，并统一坐标偏移与相位编码；
  - 自动判断当前航段行数所在范围，从而匹配相应的最优 LSTM 训练超参数；
  - 调用 runPrediction 函数运行子序列 DTW 匹配与 LSTM 补全，并获取最终预测结果；
  - 通过 drawCompletionFrames 生成多个时间关键帧的可视化窗口；
3. 查看结果：
  - 若脚本成功执行，会在实时脚本右侧窗口内输出图像窗口显示关键帧补全效果（共四张图像）

## 7. 实例脚本：Geo\_5.mlx

### 功能描述：

- 1. 数据读取与预处理
  - 从指定路径加载测试集 AIS 数据和港口静态信息数据。
  - 将 AIS 数据中的 slice\_time 转换为标准日期时间格式，并按 ship\_name 与 slice\_time 排序，确保数据时序正确。
- 2. 半径 3km 范围过滤
  - 利用 Haversine 公式计算每个航段（按 ship\_name 和 trip\_id 分组）首个数据点与目标经纬度（target\_lat = -13.2305, target\_lon = -76.7341）之间的距离。
  - 将距离小于 3km 的整组数据标记为异常，并从数据集中删除，以剔除受目标干扰的异常航段数据。
- 3. 地图设置与自定义配色
  - 选择地图底图样式（如 colorterrain）并定义经纬度调整函数 adjustLon（对经度进行调整，使得极值坐标正确显示）。
  - 设置自定义调色板，排除蓝/青色，以便在绘制船舶轨迹时使用足够的颜色区分不同航段。
- 4. 地理轨迹绘图
  - 使用 geoscatteer 将港口数据以黄色带黑边的点在地图上展示。
  - 按照 trip\_id 分组，利用 geoplot 对每艘船舶的不同航段绘制轨迹，颜色根据自定义调色板自动循环，确保每个航段轨迹有独特且清晰的颜色显示。
  - 最后，图形窗口标题设置为“测试集残缺舰船轨迹图”，以直观展现测试集数据的空间分布情况。

使用方法:

前置准备:

- 1. 如图所示，请确保修改脚本中数据路径，使其正确指向存放测试集数据文件此代码可以通过**更换不同数据集绘制不同数据集下的轨迹图像**，可选数据集如下图所示（共三个文件）

名称	修改日期
 AIS信息数据测试集_处理完成_TEST	2025/4/2 2:58
 AIS信息数据训练集_处理完成_TEST	2025/4/4 20:41
 船舶AIS信息数据测试集	2025/1/15 15:08
 船舶AIS信息数据训练集	2025/3/20 21:30
 港口静态信息数据	2025/3/17 18:45

名称	修改日期
 代码运行所需数据	2025/4/9 18:13
 源代码(.mlx)	2025/4/9 15:25
 附件3_研究报告	2025/4/9 14:23
 预测结果	2025/4/4 13:53

“AIS 信息数据训练集\_处理完成\_TEST.csv”、“AIS 信息数据测试集\_处理完成\_TEST.csv”、  
“预测结果.csv”（修改对应代码第 5 行读取路径即可，此处以测试集为例）

```
4 %% 数据读取
5 aisData = readtable("C:\Users\幻15\Desktop\Result\AIS信息数据测试集_处理完成_TEST.csv");
```

## • 运行流程：

1. 在 MATLAB R2023a 中打开 **Geo\_5.mlx** 实时脚本文件。
2. 运行后在实时脚本右侧窗口内生成对应轨迹图像。

# 7. 脚本代码展示

## 1.代码展示：TRAIN\_END.mlx

```
%% 清空环境变量，初始化
clear; clc;
disp('数据处理开始...');
tic;

%% 导入原始数据
disp('>> 正在导入原始数据...');
AIS_Data = readtable('C:\Users\幻 15\Desktop\赛题数据\船舶 AIS 信息数据训练集.csv');
Port_Data = readtable('C:\Users\幻 15\Desktop\赛题数据\港口静态信息数据.xls');

%% 剔除每艘船中时间重复的记录（依据 ship_name 和 slice_time），只保留第一次出现的记录
disp('>> 正在剔除每艘船中时间重复的记录...');
[~, ia, ~] = unique(AIS_Data(:, {'ship_name', 'slice_time'}), 'stable');
num_removed = height(AIS_Data) - length(ia);
AIS_Data = AIS_Data(ia,:);
fprintf('>> 已剔除 %d 条重复记录\n', num_removed);
n_records = height(AIS_Data);

%% 【步骤 0】原始数据机器学习辅助校验（决策树模型）
disp('>> [步骤 0] 正在对原始数据进行机器学习辅助校验...');

% 筛选出 status 非 NaN 的记录，避免 NaN 对模型训练的影响
validIdx = ~isnan(AIS_Data.status);
if sum(validIdx) < 1
```

```

disp('>> 原始数据中无有效 status 记录，跳过机器学习辅助校验。');
else
    % 提取用于模型的特征和标签（此处选取对地速度、纬度、经度作为特征）
    features_raw = [AIS_Data.sog(validIdx), AIS_Data.lat(validIdx),
AIS_Data.lon(validIdx)];
    labels_raw = AIS_Data.status(validIdx);

    % 构建决策树模型并进行交叉验证
    treeModel_raw = fitctree(features_raw, labels_raw, 'CrossVal', 'on');
    cvLoss = kfoldLoss(treeModel_raw);
    validationAccuracy = 1 - cvLoss;
    fprintf('>> 原始数据机器学习模型交叉验证准确率: %.2f%%\n', validationAccuracy * 100);
end
disp('>> 原始数据机器学习辅助校验完成! ');

%% 【步骤 1】滑动窗口异常值检测与处理（最前面）
disp('>> [步骤 1] 正在进行滑动窗口异常值检测与处理...');

% 先基于原始 leg_end_port_code 临时分段（仅用于滑窗处理）
new_trip_id_temp = zeros(n_records,1);
unique_ships = unique(AIS_Data.ship_name);
temp_trip_counter = 0;
for s = 1:length(unique_ships)
    ship_id = unique_ships(s);
    idx_ship = find(AIS_Data.ship_name == ship_id);
    % 按时间排序
    ship_times = datetime(AIS_Data.slice_time(idx_ship), 'InputFormat', 'yyyy/M/d HH:mm');
    [~, order] = sort(ship_times);
    idx_ship = idx_ship(order);
    % 分段：连续记录若 leg_end_port_code 相同则属于同一段
    seg_start = 1;
    current_leg = AIS_Data.leg_end_port_code{idx_ship(1)};
    for j = 2:length(idx_ship)
        if ~strcmp(AIS_Data.leg_end_port_code{idx_ship(j)}, current_leg)
            temp_trip_counter = temp_trip_counter + 1;
            new_trip_id_temp(idx_ship(seg_start:j-1)) = temp_trip_counter;
            seg_start = j;
            current_leg = AIS_Data.leg_end_port_code{idx_ship(j)};
        end
    end
    temp_trip_counter = temp_trip_counter + 1;
    new_trip_id_temp(idx_ship(seg_start:end)) = temp_trip_counter;
end

```



% 改进：自适应窗口、动态阈值、modified Z-score、趋势插值及加权窗口法

mod\_z\_threshold = 8; % modified Z-score 阈值

% 初始化连续变量校正版本（后续输出使用）

sog\_corr = AIS\_Data.sog;

cog\_corr = AIS\_Data.cog;

draught\_corr = AIS\_Data.draught;

lon\_corr = AIS\_Data.lon;

lat\_corr = AIS\_Data.lat;

% 按临时分段 new\_trip\_id\_temp 对连续变量进行处理

for seg = 1:temp\_trip\_counter

idx = find(new\_trip\_id\_temp == seg);

L = length(idx);

if L < 3

continue; % 数据点不足则跳过

end

% 自适应窗口：窗口大小取当前分段长度的 10%，至少 3，最多 11

win\_size = round(0.1 \* L);

if win\_size < 3, win\_size = 3; end

if mod(win\_size,2) == 0, win\_size = win\_size + 1; end

if win\_size > 11, win\_size = 11; end

half\_window\_adapt = floor(win\_size/2);

%% 对 sog 处理（使用 modified Z-score 和趋势线性插值替代异常值）

sog\_seg = sog\_corr(idx);

flag = false(size(sog\_seg));

for j = 1:L

win\_idx = max(1, j-half\_window\_adapt) : min(L, j+half\_window\_adapt);

window\_data = sog\_seg(win\_idx);

med\_val = median(window\_data);

mad\_val = median(abs(window\_data - med\_val));

if mad\_val == 0

mod\_z = 0;

else

mod\_z = 0.6745\*(sog\_seg(j)-med\_val)/mad\_val;

end

if abs(mod\_z) > mod\_z\_threshold

flag(j) = true;

end

end

valid\_idx = find(~flag);

if isempty(valid\_idx)

sog\_seg(:) = med\_val;

else

sog\_seg = interp1(valid\_idx, sog\_seg(valid\_idx), 1:L, 'linear', 'extrap');

end

```

sog_corr(idx) = sog_seg;

%% 对 cog 处理（单变量处理保持不变）
cog_seg = cog_corr(idx);
flag = false(size(cog_seg));
for j = 1:L
    win_idx = max(1, j-half_window_adapt) : min(L, j+half_window_adapt);
    window_data = cog_seg(win_idx);
    med_val = median(window_data);
    mad_val = median(abs(window_data - med_val));
    if mad_val == 0
        mod_z = 0;
    else
        mod_z = 0.6745*(cog_seg(j)-med_val)/mad_val;
    end
    if abs(mod_z) > mod_z_threshold
        flag(j) = true;
    end
end
valid_idx = find(~flag);
if isempty(valid_idx)
    cog_seg(:) = med_val;
else
    cog_seg = interp1(valid_idx, cog_seg(valid_idx), 1:L, 'linear', 'extrap');
end
cog_corr(idx) = cog_seg;

%% 对 draught 处理（单变量处理保持不变）
draught_seg = draught_corr(idx);
flag = false(size(draught_seg));
for j = 1:L
    win_idx = max(1, j-half_window_adapt) : min(L, j+half_window_adapt);
    window_data = draught_seg(win_idx);
    med_val = median(window_data);
    mad_val = median(abs(window_data - med_val));
    if mad_val == 0
        mod_z = 0;
    else
        mod_z = 0.6745*(draught_seg(j)-med_val)/mad_val;
    end
    if abs(mod_z) > mod_z_threshold
        flag(j) = true;
    end
end
valid_idx = find(~flag);
if isempty(valid_idx)
    draught_seg(:) = med_val;
else
    draught_seg = interp1(valid_idx, draught_seg(valid_idx), 1:L, 'linear', 'extrap');
end

```

```

end
draught_corr(idx) = draught_seg;

%% 对经纬度采用联合处理（改进部分：4IQR法和联合插值）
% 将经纬度作为整体处理：构成 Lx2 矩阵
coords_seg = [lon_corr(idx), lat_corr(idx)]; % 每行为 [lon, lat]（单位：度）
flag = false(L,1);
for j = 1:L
    win_idx = max(1, j-half_window_adapt) : min(L, j+half_window_adapt);
    window_data = coords_seg(win_idx, :);
    % 计算窗口内各点到中位向量的欧氏距离
    med_vec = median(window_data, 1);
    dists = sqrt(sum((window_data - med_vec).^2, 2));
    % 采用 4IQR 法计算异常判断界限
    Q1 = prctile(dists, 25);
    Q3 = prctile(dists, 75);
    IQR_val = Q3 - Q1;
    lower_bound = Q1 - 4 * IQR_val;
    upper_bound = Q3 + 4 * IQR_val;
    d_curr = sqrt(sum((coords_seg(j,:) - med_vec).^2));
    if d_curr < lower_bound || d_curr > upper_bound
        flag(j) = true;
    end
end

% 对于异常点采用插值法进行修正（联合处理经纬度）
for j = 1:L
    if flag(j)
        win_idx = max(1, j-half_window_adapt) : min(L, j+half_window_adapt);
        window_data = coords_seg(win_idx, :);
        % 取窗口内未被判定为异常的点（注意需要映射到当前窗口的索引）
        valid_mask = ~flag(win_idx);
        if ~any(valid_mask)
            % 如果窗口中无正常点，则用窗口中位向量替换
            coords_seg(j,:) = median(window_data, 1);
        else
            valid_idx = find(valid_mask);
            % 当前点在窗口中的位置
            pos = j - win_idx(1) + 1;
            % 对经度和纬度分别进行线性插值
            new_lon = interp1(valid_idx, window_data(valid_idx,1), pos, 'linear',
'extrap');
            new_lat = interp1(valid_idx, window_data(valid_idx,2), pos, 'linear',
'extrap');
            coords_seg(j,:) = [new_lon, new_lat];
        end
    end
end
end

```

```

% 更新校正后的经纬度（单位仍为度）
lon_corr(idx) = coords_seg(:,1);
lat_corr(idx) = coords_seg(:,2);
end
disp('>> 滑动窗口异常值检测与处理完成! ');

%% 【步骤 2】status 列校正（使用滑动窗口处理后的连续变量）
disp('>> [步骤 2] 正在处理 status 列异常值...');
% 定义阈值
speed_threshold = 0.5;      % 判定速度为 0 的阈值(节)
distance_threshold = 30;    % 判定靠泊的最大距离(km)
% 初始化 status_corr（使用原始值作为初始）
status_corr = AIS_Data.status;
wgs84 = wgs84Ellipsoid('kilometer'); % 使用同一椭球
for i = 1:n_records
    sog = sog_corr(i);
    lat = lat_corr(i);
    lon = lon_corr(i);
    % 将角度转换为弧度传入 distance 函数
    distances = distance(deg2rad(lat), deg2rad(lon), deg2rad(Port_Data.lat),
deg2rad(Port_Data.lon), wgs84);
    [min_dist, ~] = min(distances);
    if sog < speed_threshold && min_dist <= distance_threshold
        status_corr(i) = 5;
    elseif sog < speed_threshold && min_dist > distance_threshold
        status_corr(i) = 1;
    elseif sog >= speed_threshold
        status_corr(i) = 0;
    end
    if mod(i,5000)==0
        fprintf('已处理 status 记录: %d / %d\n', i, n_records);
    end
end
disp('>> status 列异常处理完成! ');

%% 【步骤 3】填充 leg_end_port_code 缺失或错误值
disp('>> [步骤 3] 正在处理 leg_end_port_code 缺失或错误值...');
leg_end_port_corr = AIS_Data.leg_end_port_code; % 复制原始数据
for i = 1:n_records
    current_port = AIS_Data.leg_end_port_code{i};
    lat = lat_corr(i);
    lon = lon_corr(i);
    if isempty(current_port) || ~ismember(current_port, Port_Data.port_code)
        distances = distance(deg2rad(lat), deg2rad(lon), deg2rad(Port_Data.lat),
deg2rad(Port_Data.lon), wgs84);
        [~, idx_min] = min(distances);
        leg_end_port_corr{i} = Port_Data.port_code{idx_min};
    end
end

```



```

end
if mod(i,5000)==0
    fprintf('已处理 leg_end_port_code 记录: %d / %d\n', i, n_records);
end
end
disp('>> leg_end_port_code 缺失或错误值处理完成! ');

%% 【步骤 4】基于填充后 leg_end_port_code 重新划分航段，并调整尾部连续的状态==1
disp('>> [步骤 4] 正在基于 leg_end_port_code 划分航段，并调整尾部连续的状态...');
new_trip_id = zeros(n_records,1);
unique_ships = unique(AIS_Data.ship_name);
trip_counter = 0;
for s = 1:length(unique_ships)
    ship_id = unique_ships(s);
    idx_ship = find(AIS_Data.ship_name == ship_id);
    ship_times = datetime(AIS_Data.slice_time(idx_ship), 'InputFormat', 'yyyy/M/d HH:mm');
    [~, order] = sort(ship_times);
    idx_ship = idx_ship(order);
    seg_start = 1;
    current_leg = leg_end_port_corr{idx_ship(1)};
    for j = 2:length(idx_ship)
        if ~strcmp(leg_end_port_corr{idx_ship(j)}, current_leg)
            trip_counter = trip_counter + 1;
            seg_idx = idx_ship(seg_start:j-1);
            new_trip_id(seg_idx) = trip_counter;
            if status_corr(seg_idx(end)) == 1
                tail_ptr = length(seg_idx);
                curr_lat = lat_corr(seg_idx(tail_ptr));
                curr_lon = lon_corr(seg_idx(tail_ptr));
                port_code = leg_end_port_corr{seg_idx(tail_ptr)};
                port_idx = find(strcmp(Port_Data.port_code, port_code), 1);
                port_lat = Port_Data.lat(port_idx);
                port_lon = Port_Data.lon(port_idx);
                dist_to_port = distance(deg2rad(curr_lat), deg2rad(curr_lon),
deg2rad(port_lat), deg2rad(port_lon), wgs84);
                if dist_to_port > 10
                    status_corr(seg_idx(tail_ptr)) = 5;
                else
                    while tail_ptr > 0 && status_corr(seg_idx(tail_ptr)) == 1
                        curr_lat = lat_corr(seg_idx(tail_ptr));
                        curr_lon = lon_corr(seg_idx(tail_ptr));
                        port_code = leg_end_port_corr{seg_idx(tail_ptr)};
                        port_idx = find(strcmp(Port_Data.port_code, port_code), 1);
                        port_lat = Port_Data.lat(port_idx);
                        port_lon = Port_Data.lon(port_idx);
                        dist_to_port = distance(deg2rad(curr_lat), deg2rad(curr_lon),
deg2rad(port_lat), deg2rad(port_lon), wgs84);
                        if dist_to_port <= 10

```

```

        status_corr(seg_idx(tail_ptr)) = 5;
    else
        break;
    end
    tail_ptr = tail_ptr - 1;
end
end
end
seg_start = j;
current_leg = leg_end_port_corr{idx_ship(j)};
end
end
trip_counter = trip_counter + 1;
seg_idx = idx_ship(seg_start:end);
new_trip_id(seg_idx) = trip_counter;
if status_corr(seg_idx(end)) == 1
    tail_ptr = length(seg_idx);
    curr_lat = lat_corr(seg_idx(tail_ptr));
    curr_lon = lon_corr(seg_idx(tail_ptr));
    port_code = leg_end_port_corr{seg_idx(tail_ptr)};
    port_idx = find(strcmp(Port_Data.port_code, port_code), 1);
    port_lat = Port_Data.lat(port_idx);
    port_lon = Port_Data.lon(port_idx);
    dist_to_port = distance(deg2rad(curr_lat), deg2rad(curr_lon), deg2rad(port_lat),
deg2rad(port_lon), wgs84);
    if dist_to_port > 10
        status_corr(seg_idx(tail_ptr)) = 5;
    else
        while tail_ptr > 0 && status_corr(seg_idx(tail_ptr)) == 1
            curr_lat = lat_corr(seg_idx(tail_ptr));
            curr_lon = lon_corr(seg_idx(tail_ptr));
            port_code = leg_end_port_corr{seg_idx(tail_ptr)};
            port_idx = find(strcmp(Port_Data.port_code, port_code), 1);
            port_lat = Port_Data.lat(port_idx);
            port_lon = Port_Data.lon(port_idx);
            dist_to_port = distance(deg2rad(curr_lat), deg2rad(curr_lon),
deg2rad(port_lat), deg2rad(port_lon), wgs84);
            if dist_to_port <= 10
                status_corr(seg_idx(tail_ptr)) = 5;
            else
                break;
            end
            tail_ptr = tail_ptr - 1;
        end
    end
end
end
disp('>> 航段划分及尾部 status 调整完成! ');

```

```

%% 【步骤 5】依据航段编号对数据分组，调整每个航段内的 status
disp('>> [步骤 5] 正在依据航段编号调整各航段内 status...');
unique_trip_ids = unique(new_trip_id);
for t = 1:length(unique_trip_ids)
    tid = unique_trip_ids(t);
    idx_trip = find(new_trip_id == tid);
    % 假设 idx_trip 已按时间排序（步骤 4 已保证）
    S = status_corr(idx_trip);

    % 重复循环检测整个航段内是否存在多组连续的 5
    while true
        blocks = {}; % 存储所有连续的 5 的起始和结束索引
        i = 1;
        while i <= length(S)
            if S(i) == 5
                start_idx = i;
                while i < length(S) && S(i+1) == 5
                    i = i + 1;
                end
                end_idx = i;
                blocks{end+1} = [start_idx, end_idx];
            end
            i = i + 1;
        end
        % 如果只有一组或没有 5，则退出循环
        if length(blocks) <= 1
            break;
        else
            % 将第一组（最早出现的连续 5）全部修正为 1
            pos = blocks{1};
            S(pos(1):pos(2)) = 1;
        end
    end
    status_corr(idx_trip) = S;
end
disp('>> 航段内 status 调整完成! ');

%% 【步骤 6】生成每艘船的出发港口 leg_begin_port_code 与航迹编号 trip_id
disp('>> [步骤 6] 正在生成每艘船的出发港口 leg_begin_port_code 与航迹划分编号 trip_id...');
trip_id = new_trip_id;
leg_begin_port_code = cell(n_records,1);
leg_begin_port_code(:) = {"NO"};
unique_ships = unique(AIS_Data.ship_name);
for s = 1:length(unique_ships)
    ship_id = unique_ships(s);

```

```

idx_ship = find(AIS_Data.ship_name == ship_id);
ship_times = datetime(AIS_Data.slice_time(idx_ship), 'InputFormat', 'yyyy/M/d HH:mm');
[~, order] = sort(ship_times);
idx_ship = idx_ship(order);
trips = unique(trip_id(idx_ship));
trips = sort(trips);
for k = 1:length(trips)
    current_trip = trips(k);
    idx_trip = idx_ship(trip_id(idx_ship)==current_trip);
    if k == 1
        leg_begin_port_code(idx_trip) = {"NO"};
    else
        prev_trip = trips(k-1);
        idx_prev = idx_ship(trip_id(idx_ship)==prev_trip);
        leg_begin = leg_end_port_corr{idx_prev(end)};
        leg_begin_port_code(idx_trip) = {leg_begin};
    end
end
end
disp('>> 航迹出发港口与划分编号生成完成! ');

```

%% 【步骤 7】对每个航段的吃水应用 4IQR 法检测与修正

```

disp('>> [新增步骤] 正在对每个航段的吃水应用 4IQR 法检测与修正...');
unique_trip = unique(trip_id);
for t = 1:length(unique_trip)
    idx_trip = find(trip_id == unique_trip(t));
    draught_seg = draught_corr(idx_trip);
    Q1 = prctile(draught_seg,25);
    Q3 = prctile(draught_seg,75);
    IQR_val = Q3 - Q1;
    lower_bound = Q1 - 4*IQR_val;
    upper_bound = Q3 + 4*IQR_val;
    med_val = median(draught_seg);
    outlier_idx = (draught_seg < lower_bound) | (draught_seg > upper_bound);
    draught_seg(outlier_idx) = med_val;
    draught_corr(idx_trip) = draught_seg;
end
disp('>> 每个航段吃水 4IQR 修正完成! ');

```

```

%% 【步骤 8】对每个航段单独应用滑动窗口 4IQR 法与联合插值法处理经纬度
disp('>> 【步骤 8】 正在对每个航段独立处理经纬度异常值...');
unique_trip_ids = unique(trip_id);
for t = 1:length(unique_trip_ids)
    tid = unique_trip_ids(t);
    idx_seg = find(trip_id == tid);
    % 按时间排序, 确保 idx_seg 中数据顺序正确
    ship_times = datetime(AIS_Data.slice_time(idx_seg), 'InputFormat', 'yyyy/M/d HH:mm');
    [~, order] = sort(ship_times);
    idx_seg = idx_seg(order);

    L_seg = length(idx_seg);
    if L_seg < 3
        continue;
    end

    % 自适应窗口: 窗口大小取当前航段长度的 10%, 控制在 3 到 11 之间
    win_size = round(0.1 * L_seg);
    if win_size < 3, win_size = 3; end
    if mod(win_size,2)==0, win_size = win_size + 1; end
    if win_size > 11, win_size = 11; end
    half_window = floor(win_size/2);

    % 提取当前航段经纬度数据 (单位: 度)
    coords_seg = [lon_corr(idx_seg), lat_corr(idx_seg)];

    % 滑动窗口内采用 4IQR 法检测异常: 以窗口内各点到中位向量的欧氏距离为依据
    flag = false(L_seg,1);
    for j = 1:L_seg
        win_idx = max(1, j-half_window) : min(L_seg, j+half_window);
        window_data = coords_seg(win_idx, :);
        med_vec = median(window_data, 1);
        dists = sqrt(sum((window_data - med_vec).^2, 2));
        Q1 = prctile(dists,25);
        Q3 = prctile(dists,75);
        IQR_val = Q3 - Q1;
        lower_bound = Q1 - 4 * IQR_val;
        upper_bound = Q3 + 4 * IQR_val;
        d_curr = sqrt(sum((coords_seg(j,:) - med_vec).^2));
        if d_curr < lower_bound || d_curr > upper_bound
            flag(j) = true;
        end
    end

    % 对异常点采用联合线性插值进行修正
    for j = 1:L_seg
        if flag(j)
            win_idx = max(1, j-half_window) : min(L_seg, j+half_window);
            window_data = coords_seg(win_idx, :);

```



```

        valid_mask = ~flag(win_idx);
        if ~any(valid_mask)
            % 无正常数据：用窗口中位向量替换
            coords_seg(j,:) = median(window_data, 1);
        else
            valid_idx = find(valid_mask);
            % 计算当前点在窗口内的相对位置
            pos = j - win_idx(1) + 1;
            new_lon = interp1(valid_idx, window_data(valid_idx,1), pos, 'linear',
'extrap');
            new_lat = interp1(valid_idx, window_data(valid_idx,2), pos, 'linear',
'extrap');
            coords_seg(j,:) = [new_lon, new_lat];
        end
    end
end
% 更新当前航段内的经纬度数据
lon_corr(idx_seg) = coords_seg(:,1);
lat_corr(idx_seg) = coords_seg(:,2);
end
disp('>> 每个航段经纬度独立处理完成! ');

%% 【步骤 9】依据航段编号对数据分组，调整航段内的 status
disp('>> [新增步骤 10] 正在依据航段编号重新调整各航段内 status...');

unique_ships = unique(AIS_Data.ship_name);
for s = 1:length(unique_ships)
    ship_id = unique_ships(s);
    idx_ship = find(AIS_Data.ship_name == ship_id);
    % 按时间排序
    ship_times = datetime(AIS_Data.slice_time(idx_ship), 'InputFormat', 'yyyy/M/d HH:mm');
    [~, order] = sort(ship_times);
    idx_ship = idx_ship(order);

    % 对该船内的记录，循环检测并调整每个航段
    changeMade = true;
    while changeMade
        changeMade = false;
        ship_trip_ids = trip_id(idx_ship);
        unique_trips = unique(ship_trip_ids, 'stable'); % 保持原有顺序
        % 对每个航段（除最后一段）进行检查
        for t = 1:length(unique_trips)-1
            currentTrip = unique_trips(t);
            idx_current = idx_ship(ship_trip_ids == currentTrip);
            S = status_corr(idx_current);
            % 查找当前航段中最后一个 status==5 的位置

```

```

        p = find(S == 5, 1, 'last');
        if ~isempty(p) && p < length(S)
            % 若从 p+1 到段尾中存在 status==0, 则将这些记录归入下一航段
            if any(S(p+1:end) == 0)
                nextTrip = unique_trips(t+1);
                idx_to_move = idx_current(p+1:end);
                trip_id(idx_to_move) = nextTrip;
                changeMade = true;
            end
        end
    end
end
end
disp('>> 航段内 status 重新调整完成! ');

```

%% 【步骤 10】去除各船第一和最后航段并重新排序航段编号

```
disp('>> [步骤 10] 正在去除每艘船的首尾航段并重新排序航段编号...');
```

% 初始化逻辑索引数组，用于标记待删除的记录

```
remove_idx = false(height(AIS_Data), 1);
```

% 获取所有唯一船舶编号

```
unique_ships = unique(AIS_Data.ship_name);
```

```
for s = 1:length(unique_ships)
```

```
    ship_id = unique_ships(s);
```

% 获取当前船舶所有记录的索引

```
    idx_ship = find(AIS_Data.ship_name == ship_id);
```

% 按时间排序（切片时间格式为 'yyyy/M/d HH:mm'）

```
    ship_times = datetime(AIS_Data.slice_time(idx_ship), 'InputFormat', 'yyyy/M/d HH:mm');
```

```
    [~, order] = sort(ship_times);
```

```
    idx_ship_sorted = idx_ship(order);
```

% 获取当前船舶中的唯一航段编号（保持原始顺序）

```
    ship_trip_ids = trip_id(idx_ship_sorted);
```

```
    unique_trips = unique(ship_trip_ids, 'stable');
```

% 如果航段数不少于 2，则标记首尾两个航段，否则全部标记

```
    if length(unique_trips) >= 2
```

```
        remove_trip_ids = [unique_trips(1), unique_trips(end)];
```

```
    else
```

```
        remove_trip_ids = unique_trips;
```

```
    end
```

% 标记该船中属于首尾航段的记录

```
    remove_idx(idx_ship_sorted(ismember(ship_trip_ids, remove_trip_ids))) = true;
```

```
end
```

```

% 删除被标记的记录，并同步更新所有相关变量
AIS_Data(remove_idx, :) = [];
trip_id(remove_idx) = [];
status_corr(remove_idx) = [];
leg_begin_port_code(remove_idx) = [];
leg_end_port_corr(remove_idx) = [];
lon_corr(remove_idx) = [];
lat_corr(remove_idx) = [];
sog_corr(remove_idx) = [];
cog_corr(remove_idx) = [];
draught_corr(remove_idx) = [];
n_records = height(AIS_Data);

% 重新对剩余数据中的航段编号进行连续排序
new_trip_id_final = zeros(n_records,1);
new_trip_counter = 0;
unique_ships = unique(AIS_Data.ship_name);
for s = 1:length(unique_ships)
    ship_id = unique_ships(s);
    idx_ship = find(AIS_Data.ship_name == ship_id);
    % 按时间排序
    ship_times = datetime(AIS_Data.slice_time(idx_ship), 'InputFormat', 'yyyy/M/d HH:mm');
    [~, order] = sort(ship_times);
    idx_ship_sorted = idx_ship(order);
    unique_trips = unique(trip_id(idx_ship_sorted), 'stable');
    for k = 1:length(unique_trips)
        new_trip_counter = new_trip_counter + 1;
        % 将当前航段的所有记录赋予新的连续编号
        new_trip_id_final(trip_id == unique_trips(k) & AIS_Data.ship_name == ship_id) =
new_trip_counter;
    end
end
trip_id = new_trip_id_final;
disp('>> 首尾航段移除及航段编号重新排序完成! ');

%% 【步骤 11】机器学习辅助校验（决策树模型）
disp('>> [步骤 11] 正在进行机器学习辅助校验（分类模型）...');
features = [sog_corr, lat_corr, lon_corr];
labels = status_corr;
classificationTree = fitctree(features, labels, 'CrossVal', 'on');
validationAccuracy = 1 - kfoldLoss(classificationTree);
fprintf('机器学习模型交叉验证准确率: %.2f%%\n', validationAccuracy*100);
disp('>> 机器学习辅助校验完成！（注意：实际应用中，可进一步细化）');

```

%% 【步骤 12】输出最终处理后的数据

```
disp('>> [步骤 8] 正在输出处理后的数据到文件...');
Final_Data = AIS_Data;
Final_Data.lon = lon_corr;
Final_Data.lat = lat_corr;
Final_Data.sog = sog_corr;
Final_Data.cog = cog_corr;
Final_Data.draught = draught_corr;
Final_Data.status = status_corr;
Final_Data.leg_end_port_code = leg_end_port_corr;
Final_Data.leg_begin_port_code = leg_begin_port_code;
Final_Data.trip_id = trip_id;
Final_Data = Final_Data(:,
{'ship_name', 'slice_time', 'lon', 'lat', 'hdg', 'sog', 'cog', 'draught', 'status', 'leg_end_port_c
ode', 'leg_begin_port_code', 'trip_id'});
```

%% 【步骤 13】绘制特征间热力图（低值为白色，高值为红色，并在图中标注数值）

```
disp('>> [步骤 13] 正在绘制特征间热力图...');
```

% 选择需要计算相关系数的数值特征

```
featureNames = {'lon', 'lat', 'hdg', 'sog', 'cog', 'draught'};
dataMatrix = Final_Data(:, featureNames);
```

% 计算相关系数矩阵

```
corrMatrix = corrcoef(dataMatrix);
```

% 新建图形窗口并绘制热力图

```
figure;
imagesc(corrMatrix);
axis square; % 保持图形为正方形
title('特征相关性热力图', 'FontSize', 14, 'FontWeight', 'bold');
colorbar;
set(gca, 'XTick', 1:length(featureNames), 'XTickLabel', featureNames, ...
'YTick', 1:length(featureNames), 'YTickLabel', featureNames, ...
'FontSize', 12, 'FontWeight', 'bold');
```

% 创建自定义渐变色图：低值用白色，高值用红色

```
N = 256;
```

```

lowColor = [1, 1, 1]; % 白色
highColor = [1, 0, 0]; % 红色
customColormap = [linspace(lowColor(1), highColor(1), N)', ...
                  linspace(lowColor(2), highColor(2), N)', ...
                  linspace(lowColor(3), highColor(3), N)'];
colormap(customColormap);

% 根据相关系数矩阵的最小值与最大值调整颜色映射
caxis([min(corrMatrix(:)) max(corrMatrix(:))]);

% 在热力图上添加数值标签，并根据背景亮度选择文本颜色
for i = 1:length(featureNames)
    for j = 1:length(featureNames)
        value = corrMatrix(i,j);
        % 计算该值在当前色图中的位置比例 t (0 对应最低值, 1 对应最高值)
        t = (value - min(corrMatrix(:))) / (max(corrMatrix(:)) - min(corrMatrix(:)));
        % 线性插值计算该处颜色
        currColor = lowColor + (highColor - lowColor) * t;
        brightness = mean(currColor);
        if brightness < 0.5
            txtColor = 'w'; % 背景较暗时用白色文字
        else
            txtColor = 'k'; % 背景较亮时用黑色文字
        end
        text(j, i, sprintf('%.2f', value), 'HorizontalAlignment', 'center', ...
            'FontSize', 12, 'FontWeight', 'bold', 'Color', txtColor);
    end
end

disp('>> 热力图绘制完成! ');

output_folder = 'C:\Users\幻 15\Desktop\Result';
if ~exist(output_folder, 'dir')
    mkdir(output_folder);
end
output_file = fullfile(output_folder, 'AIS 信息数据训练集_处理完成_TEST.csv');
disp(['>> 正在输出处理后的数据到文件: ', output_file]);
writetable(Final_Data, output_file);
disp('>> 数据输出完成! 处理结束。');
toc;

```



## 2.代码展示: TEST\_END.mlx

```
%% 清空环境变量，初始化
clear; clc;
disp('测试集数据处理开始...');
tic;

%% 导入测试集原始数据与港口静态信息
disp('>> 正在导入测试集原始数据...');
AIS_Test = readtable('C:\Users\幻 15\Desktop\赛题数据 - 副本\船舶 AIS 信息数据测试集.csv');
Port_Data = readtable('C:\Users\幻 15\Desktop\赛题数据 - 副本\港口静态信息数据.xls');

%% 剔除每艘船中时间重复的记录（依据 ship_name 和 slice_time）
disp('>> 正在剔除每艘船中时间重复的记录...');
[~, ia, ~] = unique(AIS_Test(:, {'ship_name', 'slice_time'}), 'stable');
num_removed = height(AIS_Test) - length(ia);
AIS_Test = AIS_Test(ia,:);
fprintf('>> 已剔除 %d 条重复记录\n', num_removed);
n_records = height(AIS_Test);

%% 【新增步骤 1】基于时间间隔对航段进行划分
disp('>> [新增步骤] 正在基于时间间隔划分航段...');
% 设定时间间隔阈值：考虑到切片时间一般为 3 小时，设定阈值为 4 小时（可根据实际情况调整）
time_gap_threshold = hours(4);
trip_id = zeros(n_records,1);
unique_ships = unique(AIS_Test.ship_name);
trip_counter = 0;
for s = 1:length(unique_ships)
    ship_id = unique_ships(s);
    idx_ship = find(AIS_Test.ship_name == ship_id);
    % 按时间排序
    ship_times = datetime(AIS_Test.slice_time(idx_ship), 'InputFormat', 'yyyy/M/d HH:mm');
    [~, order] = sort(ship_times);
    idx_ship = idx_ship(order);
    % 按时间间隔划分航段：若相邻记录时间差大于阈值，则认为存在断点，划分为不同航段
    seg_start = 1;
    for j = 2:length(idx_ship)
        curr_time = datetime(AIS_Test.slice_time(idx_ship(j)), 'InputFormat', 'yyyy/M/d HH:mm');
        prev_time = datetime(AIS_Test.slice_time(idx_ship(j-1)), 'InputFormat', 'yyyy/M/d HH:mm');
        if curr_time - prev_time > time_gap_threshold
            trip_counter = trip_counter + 1;
```

```

        trip_id(idx_ship(seg_start:j-1)) = trip_counter;
        seg_start = j;
    end
end
trip_counter = trip_counter + 1;
trip_id(idx_ship(seg_start:end)) = trip_counter;
end
disp('>> 航段划分完成! ');

```

```

%% 【步骤 1】滑动窗口异常值检测与处理（针对连续变量）
disp('>> [步骤 1] 正在进行滑动窗口异常值检测与处理...');

```

```

% 初始化连续变量的校正版本

```

```

sog_corr      = AIS_Test.sog;
cog_corr      = AIS_Test.cog;
draught_corr  = AIS_Test.draught;
lon_corr      = AIS_Test.lon;
lat_corr      = AIS_Test.lat;

```

```

unique_trip_ids = unique(trip_id);
mod_z_threshold = 8; % modified Z-score 阈值

```

```

for t = 1:length(unique_trip_ids)
    idx = find(trip_id == unique_trip_ids(t));
    L = length(idx);
    if L < 3
        continue; % 数据点不足则跳过
    end
    % 自适应窗口：窗口大小取当前航段长度的 10%，范围控制在 3 到 11 之间
    win_size = round(0.1 * L);
    if win_size < 3, win_size = 3; end
    if mod(win_size,2)==0, win_size = win_size + 1; end
    if win_size > 11, win_size = 11; end
    half_window = floor(win_size/2);

    %% sog 处理（使用 modified Z-score 与线性插值修正）
    sog_seg = sog_corr(idx);
    flag = false(size(sog_seg));
    for j = 1:L
        win_idx = max(1, j-half_window) : min(L, j+half_window);
        window_data = sog_seg(win_idx);
        med_val = median(window_data);
        mad_val = median(abs(window_data - med_val));
        if mad_val == 0
            mod_z = 0;
        else

```

```

        mod_z = 0.6745*(sog_seg(j)-med_val)/mad_val;
    end
    if abs(mod_z) > mod_z_threshold
        flag(j) = true;
    end
end
valid_idx = find(~flag);
if isempty(valid_idx)
    sog_seg(:) = med_val;
else
    sog_seg = interp1(valid_idx, sog_seg(valid_idx), 1:L, 'linear', 'extrap');
end
sog_corr(idx) = sog_seg;

%% cog 处理（同样方法）
cog_seg = cog_corr(idx);
flag = false(size(cog_seg));
for j = 1:L
    win_idx = max(1, j-half_window) : min(L, j+half_window);
    window_data = cog_seg(win_idx);
    med_val = median(window_data);
    mad_val = median(abs(window_data - med_val));
    if mad_val == 0
        mod_z = 0;
    else
        mod_z = 0.6745*(cog_seg(j)-med_val)/mad_val;
    end
    if abs(mod_z) > mod_z_threshold
        flag(j) = true;
    end
end
valid_idx = find(~flag);
if isempty(valid_idx)
    cog_seg(:) = med_val;
else
    cog_seg = interp1(valid_idx, cog_seg(valid_idx), 1:L, 'linear', 'extrap');
end
cog_corr(idx) = cog_seg;

%% draught 处理（同样方法）
draught_seg = draught_corr(idx);
flag = false(size(draught_seg));
for j = 1:L
    win_idx = max(1, j-half_window) : min(L, j+half_window);
    window_data = draught_seg(win_idx);
    med_val = median(window_data);
    mad_val = median(abs(window_data - med_val));
    if mad_val == 0
        mod_z = 0;
    end
end

```

```

        else
            mod_z = 0.6745*(draught_seg(j)-med_val)/mad_val;
        end
        if abs(mod_z) > mod_z_threshold
            flag(j) = true;
        end
    end
end
valid_idx = find(~flag);
if isempty(valid_idx)
    draught_seg(:) = med_val;
else
    draught_seg = interp1(valid_idx, draught_seg(valid_idx), 1:L, 'linear', 'extrap');
end
draught_corr(idx) = draught_seg;

%% 经纬度联合处理（采用 4IQR 法和联合插值）
coords_seg = [lon_corr(idx), lat_corr(idx)];
flag = false(L,1);
for j = 1:L
    win_idx = max(1, j-half_window) : min(L, j+half_window);
    window_data = coords_seg(win_idx, :);
    med_vec = median(window_data, 1);
    dists = sqrt(sum((window_data - med_vec).^2, 2));
    Q1 = prctile(dists,25);
    Q3 = prctile(dists,75);
    IQR_val = Q3 - Q1;
    lower_bound = Q1 - 4 * IQR_val;
    upper_bound = Q3 + 4 * IQR_val;
    d_curr = sqrt(sum((coords_seg(j,:) - med_vec).^2));
    if d_curr < lower_bound || d_curr > upper_bound
        flag(j) = true;
    end
end
for j = 1:L
    if flag(j)
        win_idx = max(1, j-half_window) : min(L, j+half_window);
        window_data = coords_seg(win_idx, :);
        valid_mask = ~flag(win_idx);
        if ~any(valid_mask)
            coords_seg(j,:) = median(window_data, 1);
        else
            valid_idx = find(valid_mask);
            pos = j - win_idx(1) + 1;
            new_lon = interp1(valid_idx, window_data(valid_idx,1), pos, 'linear',
'extrap');
            new_lat = interp1(valid_idx, window_data(valid_idx,2), pos, 'linear',
'extrap');
            coords_seg(j,:) = [new_lon, new_lat];
        end
    end
end

```

```

        end
    end
    lon_corr(idx) = coords_seg(:,1);
    lat_corr(idx) = coords_seg(:,2);
end
disp('>> 滑动窗口异常值检测与处理完成! ');

%% 【步骤 2】status 列校正（基于处理后的连续变量与港口信息）
disp('>> [步骤 2] 正在处理 status 列异常值...');
speed_threshold = 0.5;      % 判定低速的阈值（节）
distance_threshold = 30;    % 判定靠泊的最大距离（km）
wgs84 = wgs84Ellipsoid('kilometer'); % 采用 WGS84 椭球模型
status_corr = AIS_Test.status;
for i = 1:n_records
    sog = sog_corr(i);
    lat = lat_corr(i);
    lon = lon_corr(i);
    distances = distance(deg2rad(lat), deg2rad(lon), deg2rad(Port_Data.lat),
deg2rad(Port_Data.lon), wgs84);
    [min_dist, ~] = min(distances);
    if sog < speed_threshold && min_dist <= distance_threshold
        status_corr(i) = 5;
    elseif sog < speed_threshold && min_dist > distance_threshold
        status_corr(i) = 1;
    elseif sog >= speed_threshold
        status_corr(i) = 0;
    end
    if mod(i,5000)==0
        fprintf('已处理 status 记录: %d / %d\n', i, n_records);
    end
end
disp('>> status 列异常处理完成! ');

%% 【新增步骤 2】生成每艘船的出发港口（leg_begin_port_code）信息
disp('>> [新增步骤] 正在生成每艘船的出发港口 leg_begin_port_code...');
% 由于测试集无 leg_end_port_code，因此采用每个航段首个记录的位置匹配最近港口作为出发港
leg_begin_port_code = cell(n_records,1);
for s = 1:length(unique_ships)
    ship_id = unique_ships(s);
    idx_ship = find(AIS_Test.ship_name == ship_id);
    % 按时间排序
    ship_times = datetime(AIS_Test.slice_time(idx_ship), 'InputFormat', 'yyyy/M/d HH:mm');
    [~, order] = sort(ship_times);
    idx_ship = idx_ship(order);
    % 根据前面生成的 trip_id 对数据分段
    trips = unique(trip_id(idx_ship));
    trips = sort(trips);

```



```

for k = 1:length(trips)
    idx_trip = idx_ship(trip_id(idx_ship)==trips(k));
    % 以航段首个记录为出发点，根据该点与港口的距离匹配最近港口
    curr_lat = lat_corr(idx_trip(1));
    curr_lon = lon_corr(idx_trip(1));
    distances = distance(deg2rad(curr_lat), deg2rad(curr_lon), deg2rad(Port_Data.lat),
deg2rad(Port_Data.lon), wgs84);
    [~, idx_min] = min(distances);
    leg_begin_port_code(idx_trip) = {Port_Data.port_code{idx_min}};
end
end
disp('>> 航迹出发港口生成完成! ');

```

%% 【步骤 3】对每个航段的吃水（draught）应用 4IQR 法检测与修正

```

disp('>> [步骤 3] 正在对每个航段吃水进行 4IQR 修正...');
unique_trip = unique(trip_id);
for t = 1:length(unique_trip)
    idx_trip = find(trip_id == unique_trip(t));
    draught_seg = draught_corr(idx_trip);
    Q1 = prctile(draught_seg,25);
    Q3 = prctile(draught_seg,75);
    IQR_val = Q3 - Q1;
    lower_bound = Q1 - 4*IQR_val;
    upper_bound = Q3 + 4*IQR_val;
    med_val = median(draught_seg);
    outlier_idx = (draught_seg < lower_bound) | (draught_seg > upper_bound);
    draught_seg(outlier_idx) = med_val;
    draught_corr(idx_trip) = draught_seg;
end
disp('>> 吃水 4IQR 修正完成! ');

```

%% 【步骤 4】对每个航段独立应用滑动窗口 4IQR 法处理经纬度异常值

```

disp('>> [步骤 4] 正在对每个航段独立处理经纬度异常值...');
unique_trip_ids = unique(trip_id);
for t = 1:length(unique_trip_ids)
    tid = unique_trip_ids(t);
    idx_seg = find(trip_id == tid);
    ship_times = datetime(AIS_Test.slice_time(idx_seg), 'InputFormat', 'yyyy/M/d HH:mm');
    [~, order] = sort(ship_times);
    idx_seg = idx_seg(order);
    L_seg = length(idx_seg);
    if L_seg < 3
        continue;
    end
    win_size = round(0.1 * L_seg);
    if win_size < 3, win_size = 3; end
    if mod(win_size,2)==0, win_size = win_size + 1; end

```

```

if win_size > 11, win_size = 11; end
half_window = floor(win_size/2);

coords_seg = [lon_corr(idx_seg), lat_corr(idx_seg)];
flag = false(L_seg,1);
for j = 1:L_seg
    win_idx = max(1, j-half_window) : min(L_seg, j+half_window);
    window_data = coords_seg(win_idx, :);
    med_vec = median(window_data, 1);
    dists = sqrt(sum((window_data - med_vec).^2, 2));
    Q1 = prctile(dists,25);
    Q3 = prctile(dists,75);
    IQR_val = Q3 - Q1;
    lower_bound = Q1 - 4 * IQR_val;
    upper_bound = Q3 + 4 * IQR_val;
    d_curr = sqrt(sum((coords_seg(j,:) - med_vec).^2));
    if d_curr < lower_bound || d_curr > upper_bound
        flag(j) = true;
    end
end
for j = 1:L_seg
    if flag(j)
        win_idx = max(1, j-half_window) : min(L_seg, j+half_window);
        window_data = coords_seg(win_idx, :);
        valid_mask = ~flag(win_idx);
        if ~any(valid_mask)
            coords_seg(j,:) = median(window_data, 1);
        else
            valid_idx = find(valid_mask);
            pos = j - win_idx(1) + 1;
            new_lon = interp1(valid_idx, window_data(valid_idx,1), pos, 'linear',
'extrap');
            new_lat = interp1(valid_idx, window_data(valid_idx,2), pos, 'linear',
'extrap');
            coords_seg(j,:) = [new_lon, new_lat];
        end
    end
end
lon_corr(idx_seg) = coords_seg(:,1);
lat_corr(idx_seg) = coords_seg(:,2);
end
disp('>> 每个航段经纬度处理完成! ');

```

**%% 【步骤 5】机器学习辅助校验（决策树模型示例）**

```

disp('>> [步骤 5] 正在进行机器学习辅助校验（分类模型）...');
features = [sog_corr, lat_corr, lon_corr];
labels = status_corr;
classificationTree = fitctree(features, labels, 'CrossVal', 'on');

```

```

validationAccuracy = 1 - kfoldLoss(classificationTree);
fprintf('机器学习模型交叉验证准确率: %.2f%%\n', validationAccuracy*100);
disp('>> 机器学习辅助校验完成! ');

%% 【步骤 6】输出最终处理后的数据（确保排版与训练集一致）
disp('>> [步骤 6] 正在输出处理后的数据到文件...');

% 将处理结果赋值回原始数据表
Final_Data = AIS_Test;
Final_Data.lon = lon_corr;
Final_Data.lat = lat_corr;
Final_Data.sog = sog_corr;
Final_Data.cog = cog_corr;
Final_Data.draught = draught_corr;
Final_Data.status = status_corr;

% 测试集原始数据无 leg_end_port_code，但要求输出文件格式与训练集一致，
% 因此补充 leg_end_port_code 列（全部填空字符串或空值均可）
Final_Data.leg_end_port_code = repmat({''}, n_records, 1);

% 此外，保留之前生成的 leg_begin_port_code 与 trip_id 信息
Final_Data.leg_begin_port_code = leg_begin_port_code;
Final_Data.trip_id = trip_id;

% 调整列顺序，确保与训练集输出文件一致
Final_Data = Final_Data(:,
{'ship_name', 'slice_time', 'lon', 'lat', 'hdg', 'sog', 'cog', 'draught', 'status', 'leg_end_port_c
ode', 'leg_begin_port_code', 'trip_id'});

% 指定输出目录与文件名
output_folder = 'C:\Users\幻 15\Desktop\Result';
if ~exist(output_folder, 'dir')
    mkdir(output_folder);
end
output_file = fullfile(output_folder, 'AIS 信息数据测试集_处理完成_TEST.csv');
disp(['>> 正在输出处理后的数据到文件: ', output_file]);

writetable(Final_Data, output_file);
disp('>> 数据输出完成! 处理结束。');
toc;

```

### 3.代码展示：BYS\_Prepare.mlx

```
%% ===== 清空工作环境 =====
clear; clc;

%% ===== 【1】读取数据及基于步长分 8 类（代码 2 部分） =====
trainFile = 'C:\Users\幻 15\Desktop\Result\AIS 信息数据训练集_处理完成_TEST.csv';
T_train = readtable(trainFile);

% 按航迹编号（trip_id）分组，计算每个航段的数据点数量（步长）
uniqueTripIDs_train = unique(T_train.trip_id);
nTrips_train = length(uniqueTripIDs_train);

% 初始化数组存储每个航段的步长（单位：行数）
stepLengths_train = zeros(nTrips_train,1);

% 同时记录每个航段对应的 ship_name（假定同一航段船编号一致）
shipNames_train = zeros(nTrips_train,1); % 此处 ship_name 为数字

for i = 1:nTrips_train
    tid = uniqueTripIDs_train(i);
    idx = (T_train.trip_id == tid);
    stepLengths_train(i) = sum(idx);
    % 取该航段第一条记录的 ship_name
    shipNames_train(i) = T_train.ship_name(find(idx,1,'first'));
end

% 基于步长采用分位数法将航段分为 8 类，使每类航段数量大致相等
if all(stepLengths_train == stepLengths_train(1))
    binEdges = [stepLengths_train(1), stepLengths_train(1)];
    binIdx_train = ones(size(stepLengths_train));
    nBins = 1;
else
    % 计算 8 组分位数（9 个分位点：0%，12.5%，25%，37.5%，50%，62.5%，75%，87.5%，100%）
    quantiles = quantile(stepLengths_train, [0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875, 1]);
    binEdges = quantiles;
    binIdx_train = discretize(stepLengths_train, binEdges, 'IncludedEdge','right');
    nBins = 8;
end
```

```

% 统计每个类别的步长范围、平均步长（四舍五入为整数）以及航段数量
stats = cell(nBins,1);
avgVals = zeros(nBins,1);
for k = 1:nBins
    idxBin = (binIdx_train == k);
    segCount = sum(idxBin);
    if segCount > 0
        binSteps = stepLengths_train(idxBin);
        rangeMin = min(binSteps);
        rangeMax = max(binSteps);
        avgVal = round(mean(binSteps));
    else
        rangeMin = NaN; rangeMax = NaN; avgVal = NaN;
    end
    stats{k} = struct('Category', k, ...
        'StepRange', [rangeMin, rangeMax], ...
        'AvgStep', avgVal, ...
        'Count', segCount);
    avgVals(k) = avgVal;
end

% 将统计结果按平均步长从小到大排序
[~, sortIdx] = sort(avgVals, 'ascend');
sortedStats = stats(sortIdx);

% 输出分 8 类统计结果（包含类别号、步长范围、平均步长、航段数量）
fprintf('===== 代码 2 输出：航段步长（数据行数）分 8 类统计结果 =====\n');
fprintf('类别\t步长范围\t平均步长\t航段数量\n');
for i = 1:length(sortedStats)
    sStat = sortedStats{i};
    fprintf('%d\t[%d, %d]\t%d\t%d\n', ...
        sStat.Category, sStat.StepRange(1), sStat.StepRange(2), sStat.AvgStep,
        sStat.Count);
end

%% ===== 【2】按 (leg_begin_port_code, leg_end_port_code) 分组统计（代码 3 部分）
=====
% 提取训练集唯一航段信息：包含 trip_id、leg_begin_port_code 与 leg_end_port_code
unique_segments = unique(T_train(:, {'trip_id', 'leg_begin_port_code',
'leg_end_port_code'}));

% 按 leg_begin_port_code 与 leg_end_port_code 分组统计航段数量
group_stats = groupsummary(unique_segments, {'leg_begin_port_code', 'leg_end_port_code'});
% 按 GroupCount 升序排序（从少到多排列）
group_stats_sorted = sortrows(group_stats, 'GroupCount', 'ascend');

```



```

fprintf('\n===== 代码 3 输出：各分组（出发港+目的港）航段数量统计（从少到多）
===== \n');
disp(group_stats_sorted);

%% ===== 【3】新增功能：在 8 类中筛选符合条件的航段对，并输出附带 ship_name =====
fprintf('\n===== 新增功能输出：每个类别中挑选一对航段（如有） ===== \n');

% 构造一个含有 trip_id, steplen, ship_name, leg_begin_port_code, leg_end_port_code 信息的表
tripInfo = table(uniqueTripIDs_train, stepLengths_train, shipNames_train, 'VariableNames',
{'trip_id', 'steplen', 'ship_name'});
% 补充港口信息：利用 unique_segments 中的 trip_id 对应的 leg_begin_port_code 和
leg_end_port_code
tripInfo.leg_begin_port_code = strings(nTrips_train,1);
tripInfo.leg_end_port_code   = strings(nTrips_train,1);
for i = 1:nTrips_train
    tid = tripInfo.trip_id(i);
    matchIdx = find(unique_segments.trip_id == tid, 1);
    if ~isempty(matchIdx)
        tripInfo.leg_begin_port_code(i) = unique_segments.leg_begin_port_code(matchIdx);
        tripInfo.leg_end_port_code(i)   = unique_segments.leg_end_port_code(matchIdx);
    end
end

% 遍历每个类别（类别编号取自训练集的 binIdx_train）
for k = 1:nBins
    catIdx = (binIdx_train == k);
    if ~any(catIdx)
        continue;
    end

    catTripIDs = uniqueTripIDs_train(catIdx);
    catTripInfo = tripInfo(ismember(tripInfo.trip_id, catTripIDs), :);
    catMed = median(catTripInfo.steplen);

    [groupKeys, ~, groupIdx] = unique(catTripInfo(:,
{'leg_begin_port_code', 'leg_end_port_code'}));
    groupCount = accumarray(groupIdx, 1);

    validGroupIds = find(groupCount >= 2);
    if isempty(validGroupIds)
        continue;
    end

    bestPair = [];
    bestCost = Inf;

    for g = 1:length(validGroupIds)

```

```

gid = validGroupIds(g);
subsetIdx = (groupIdx == gid);
groupData = catTripInfo(subsetIdx, :);
if height(groupData) < 2
    continue;
end
for x = 1:height(groupData)-1
    for y = x+1:height(groupData)
        s1 = groupData.stepLen(x);
        s2 = groupData.stepLen(y);
        meanVal = (s1 + s2) / 2;
        distToMed = abs(meanVal - catMed);
        diffVal = abs(s1 - s2);
        cost = distToMed + 0.001 * diffVal;
        if cost < bestCost
            bestCost = cost;
            bestPair = struct('category', k, ...
                              'trip_id1', groupData.trip_id(x), ...
                              'trip_id2', groupData.trip_id(y), ...
                              'stepLen1', s1, ...
                              'stepLen2', s2, ...
                              'ship1', groupData.ship_name(x), ...
                              'ship2', groupData.ship_name(y), ...
                              'begin_port', groupData.leg_begin_port_code(x), ...
                              'end_port', groupData.leg_end_port_code(x), ...
                              'catMedian', catMed);
        end
    end
end
end

if ~isempty(bestPair)
    fprintf('\n 类别 %d 中找到一对航段:\n', bestPair.category);
    fprintf(' 出发港/目的港: (%s, %s)\n', bestPair.begin_port, bestPair.end_port);
    fprintf(' 航段 1: trip_id = %d, ship_name = %d, 步长 = %d\n', bestPair.trip_id1,
bestPair.ship1, bestPair.stepLen1);
    fprintf(' 航段 2: trip_id = %d, ship_name = %d, 步长 = %d\n', bestPair.trip_id2,
bestPair.ship2, bestPair.stepLen2);
    fprintf(' 本类别步长中位数 = %.2f\n', bestPair.catMedian);
end
end

%% ===== 【4】 加载测试集数据, 仅用于绘图 =====
testFile = 'C:\Users\幻 15\Desktop\Result\AIS 信息数据测试集_处理完成_TEST.csv';
T_test = readtable(testFile);

% 对测试集同样按 trip_id 分组计算步长

```

```

uniqueTripIDs_test = unique(T_test.trip_id);
nTrips_test = length(uniqueTripIDs_test);
stepLengths_test = zeros(nTrips_test,1);
for i = 1:nTrips_test
    tid = uniqueTripIDs_test(i);
    idx = (T_test.trip_id == tid);
    stepLengths_test(i) = sum(idx);
end

% ===== 【5】 分别绘制训练集和测试集直方图 =====

% 更加鲜明的颜色设置
histColorTrain = [0.3, 0.6, 1]; % 深蓝
histColorTest  = [1, 0.5, 0.1]; % 橙红
edgeColor = 'w';                % 白色边框

% ----- 绘制训练集直方图 -----
figure('Color',[1 1 1],'Position',[100 100 900 600]);
histogram(stepLengths_train, 'FaceColor', histColorTrain, 'EdgeColor', edgeColor,
'LineWidth',1.5);
title('训练集航段步长分布直方图','FontSize',14,'FontWeight','bold');
xlabel('航段步长（行数）','FontSize',12);
ylabel('航段数量','FontSize',12);
grid on;
set(gca, 'FontSize',12);

% ----- 绘制测试集直方图 -----
figure('Color',[1 1 1],'Position',[150 150 900 600]);
histogram(stepLengths_test, 'FaceColor', histColorTest, 'EdgeColor', edgeColor,
'LineWidth',1.5);
title('测试集航段步长分布直方图','FontSize',14,'FontWeight','bold');
xlabel('航段步长（行数）','FontSize',12);
ylabel('航段数量','FontSize',12);
grid on;
set(gca, 'FontSize',12);

% ===== 【6】 绘制叠加直方图（不含正态分布曲线） =====
figure('Color',[1 1 1],'Position',[200 200 900 600]);
hold on;
histogram(stepLengths_train, 'FaceColor', histColorTrain, 'EdgeColor', edgeColor,
'LineWidth',1.5);
histogram(stepLengths_test, 'FaceColor', histColorTest, 'EdgeColor', edgeColor,
'LineWidth',1.5, 'FaceAlpha',0.7);
title('训练集与测试集航段步长叠加直方图','FontSize',14,'FontWeight','bold');
xlabel('航段步长（行数）','FontSize',12);

```

```

ylabel('航段数量','FontSize',12);
grid on;
set(gca, 'FontSize',12);
hold off;

%% ===== 【6】 输出训练集与测试集总步长比例及平均步长信息 =====
totalTrain = sum(stepLengths_train);
totalTest  = sum(stepLengths_test);

avgTrain = mean(stepLengths_train);
avgTest  = mean(stepLengths_test);

ratioTotal = totalTrain / totalTest;
ratioAvg   = avgTest / avgTrain;

fprintf('\n===== 总步长及平均步长比例输出 =====\n');
fprintf('训练集总步长 = %d, 航段数 = %d\n', totalTrain, nTrips_train);
fprintf('测试集总步长 = %d, 航段数 = %d\n', totalTest, nTrips_test);
fprintf('训练集与测试集总步长比例 = %.4f\n', ratioTotal);

fprintf('\n训练集平均步长 = %.2f\n', avgTrain);
fprintf('测试集平均步长 = %.2f\n', avgTest);
fprintf('测试集与训练集平均步长比例 = %.4f\n', ratioAvg);

%% ===== 【7】 绘制每个类别单独的小提琴图（2行4列子图，使用明亮鲜艳颜色） =====
figure('Color',[1 1 1],'Position',[100 100 1200 600]);

brightColors = [
    0.529, 0.808, 0.922; % 淡蓝 (LightSkyBlue)
    0.596, 0.984, 0.596; % 薄荷绿 (PaleGreen)
    1.000, 1.000, 0.600; % 柠檬黄 (LightYellow)
    1.000, 0.647, 0.529; % 珊瑚橙 (LightCoral)
    0.933, 0.510, 0.933; % 浅紫 (Violet)
    0.678, 0.847, 0.902; % 天蓝 (LightBlue)
    1.000, 0.714, 0.757; % 粉红 (LightPink)
    0.565, 0.933, 0.565 % 青绿 (MediumAquamarine)
];

```

```

uniqueGroups = unique(binIdx_train);
for i = 1:length(uniqueGroups)
    g = uniqueGroups(i);
    groupData = stepLengths_train(binIdx_train == g);

    % 核密度估计
    [f, xi] = ksdensity(groupData);
    f = f / max(f) * 0.3;

    subplot(2,4,i);
    hold on;
    % 小提琴图: 对称填充
    fill([1 - f, fliplr(1 + f)], [xi, fliplr(xi)], brightColors(i,:), ...
        'EdgeColor', 'k', 'FaceAlpha', 0.7);
    % 中位数线
    medVal = median(groupData);
    plot([0.7, 1.3], [medVal, medVal], 'k-', 'LineWidth', 2);

    title(sprintf('类别 %d', g), 'FontWeight', 'bold', 'FontSize', 12);
    xlabel(''); ylabel('步长');
    xlim([0.5 1.5]);
    ylim([min(groupData)*0.95, max(groupData)*1.05]);
    set(gca, 'XTick', []);
    grid on;
    hold off;
end

sgtitle('各类别航段步长小提琴图', 'FontSize', 16, 'FontWeight', 'bold');

```

## 4.代码展示: BYS\_LSTM\_1.mlx

```

%% 主程序
clc; clear; close all; tic;

%% 可调参数定义（均放在代码前面，不影响后续变量传递）
% 文件路径（请根据实际情况修改路径）
filePath = 'C:\Users\幻15\Desktop\Result\AIS 信息数据训练集_处理完成_TEST.csv';
% 采样间隔（小时），用于后续预测结果时间尺度的确定
sampleInterval = 3;
% 贝叶斯优化待调参数初始范围（优化不限范围，可根据需要调整）
optimVars = [...

```

```

optimizableVariable('startRemovalPct',[0.05, 0.2], 'Type', 'real'); ...
optimizableVariable('endRemovalPct',[0.18, 0.3], 'Type', 'real'); ...
optimizableVariable('initialLearnRate',[1e-4, 3e-3], 'Transform', 'log'); ...
optimizableVariable('maxEpochs',[1000, 4000], 'Type', 'integer');]
numIterations = 100; % 迭代次数

```

% 定义各类别航段信息（共 8 组），表中每组每个结构体包含：类别、出发港→目的港（仅作注释）、航段 1 信息、航段 2 信息

% 航段信息格式: [trip\_id, ship\_name, 步长（行数）]

```

categories = { ...
    struct('cat',1, 'route','DELM→DESNZ', 'seg1',[289,8,14], 'seg2',[303,8,14]); ...
    struct('cat',2, 'route','DZAZW→TRALI', 'seg1',[1086,34,40], 'seg2',[1130,34,41]); ...
    struct('cat',3, 'route','AUDPR→CNSHZ', 'seg1',[1242,37,63], 'seg2',[1779,58,63]); ...
    struct('cat',4, 'route','BEZEE→RUSAB', 'seg1',[42,2,79], 'seg2',[106,4,79]); ...
    struct('cat',5, 'route','AUGLE→CNTJN', 'seg1',[1287,39,100], 'seg2',[1392,42,100]); ...
    struct('cat',6, 'route','QARAS→ITPVT', 'seg1',[2284,86,125], 'seg2',[2324,88,129]); ...
    struct('cat',7, 'route','CNSHZ→AUDPR', 'seg1',[1231,37,160], 'seg2',[1840,60,160]); ...
    struct('cat',8, 'route','BEZEE→CNSHZ', 'seg1',[393,11,220], 'seg2',[1602,50,220]);
}

```

%% 读取数据（全局只读一次）

% 数据文件包含 12 个字段，自动读取为 table 格式

```
data = readtable(filePath);
```

%% 预处理：计算相位的 sin 编码与 cos 编码

% 注意：hdg 假定以角度表示，因此需要转为弧度

```
data.sin_phase = sin(deg2rad(data.hdg));
```

```
data.cos_phase = cos(deg2rad(data.hdg));
```

%% 对每个类别进行贝叶斯优化（共 8 组），目标：使该类别中航段 1 与航段 2 的最终评分均值最大

```

for idx = 1:length(categories)
    catInfo = categories{idx};
    fprintf('\n===== 开始类别 %d (%s) 的贝叶斯优化 =====\n',
catInfo.cat, catInfo.route);

```

% 定义目标函数，输入待调参数，输出目标值（取负的平均最终评分）

```
ObjFcn = @(x) objectiveFunction(x, catInfo, data, sampleInterval);
```

```

results = bayesopt(ObjFcn,optimVars, 'MaxObjectiveEvaluations', numIterations, ...
    'AcquisitionFunctionName','expected-improvement-plus', 'Verbose',1, 'UseParallel',
false);

```

```
bestParams = results.XAtMinObjective;
```

```
bestScore = -results.MinObjective; % 转换回最大平均最终评分
```



```

    fprintf('类别 %d 最优参数: startRemovalPct=%.4f, endRemovalPct=%.4f,
initialLearnRate=%.4e, maxEpochs=%d\n', ...
        catInfo.cat, bestParams.startRemovalPct, bestParams.endRemovalPct,
bestParams.initialLearnRate, bestParams.maxEpochs);
    fprintf('类别 %d 最优平均最终评分: %.4f\n', catInfo.cat, bestScore);
end

toc;

%% ----- 辅助函数部分 -----

```

```

function obj = objectiveFunction(optParams, catInfo, data, sampleInterval)
    % objectiveFunction - 目标函数，对给定类别（包含航段 1 和航段 2）分别运行预测流程，
    % 返回两个航段最终评分的平均值的负数（因 bayesopt 为最小化）
    % 输入：
    %   optParams - 待调参数结构体，包含 startRemovalPct, endRemovalPct, initialLearnRate,
maxEpochs
    %   catInfo   - 当前类别信息结构体，字段 seg1 和 seg2 为[trip_id, ship_name, stepLength]
    %   data      - 全部 AIS 数据表
    %   sampleInterval - 采样间隔（小时）

    % 对航段 1
    F1 = runPrediction(catInfo.seg1(2), catInfo.seg1(1), catInfo.seg1(3), ...
        optParams.startRemovalPct, optParams.endRemovalPct, optParams.initialLearnRate,
optParams.maxEpochs, data, sampleInterval);
    % 对航段 2
    F2 = runPrediction(catInfo.seg2(2), catInfo.seg2(1), catInfo.seg2(3), ...
        optParams.startRemovalPct, optParams.endRemovalPct, optParams.initialLearnRate,
optParams.maxEpochs, data, sampleInterval);
    avgF = (F1 + F2) / 2;

    fprintf('【目标函数】航段 1 评分=%.4f, 航段 2 评分=%.4f, 平均评分=%.4f\n', F1, F2, avgF);
    obj = -avgF; % 取负值，因 bayesopt 最小化目标
end

```

```

function F = runPrediction(shipID, tripID, stepLength, startRemovalPct, endRemovalPct,
initialLearnRate, maxEpochs, data, sampleInterval)
    % runPrediction - 对指定船舶(shipID)与航段(tripID)，基于完整航迹（仅取前 stepLength 行）运行
    % 残缺轨迹提取、子序 DTW 匹配、网络训练、预测及后处理，计算最终评分 F。
    % 输入：
    %   shipID, tripID - 指定航段
    %   stepLength     - 期望航迹行数（步长）
    %   startRemovalPct, endRemovalPct - 残缺轨迹截取比例
    %   initialLearnRate, maxEpochs   - 网络训练参数

```

```

% data          - 全部 AIS 数据表
% sampleInterval - 采样间隔（小时）
%
% 输出：
% F - 最终评分

%% 挑选一段完整航迹（仅用于提取残缺轨迹）
trajectory = pickFullTrajectory(data, shipID, tripID);
% 若完整航迹行数超过期望步长，则截取前 stepLength 行；否则按原有行数
if height(trajectory) >= stepLength
    trajectory = trajectory(1:stepLength, :);
end
% 构造完整航迹矩阵，包含 6 个特征变量：lon, lat, sog, cog, sin_phase, cos_phase
fullTrajectory = [trajectory.lon, trajectory.lat, ...
                  trajectory.sog, trajectory.cog, ...
                  trajectory.sin_phase, trajectory.cos_phase];

%% 构建残缺轨迹（查询序列）
n = size(fullTrajectory,1); % 航迹点数目
startIdx = floor(n * startRemovalPct) + 1;
endIdx    = n - floor(n * endRemovalPct);
incompleteTrajectory = fullTrajectory(startIdx:endIdx, :);

%% =====（原有）子序 DTW 匹配：与当前完整航迹进行搜索=====
[minDist, bestStart, bestPathQuery, bestPathRef] = subseq_dtw(incompleteTrajectory,
fullTrajectory);
alignedSegment = fullTrajectory(bestStart : bestStart + size(incompleteTrajectory,1) -
1, :);
if bestStart + size(incompleteTrajectory,1) - 1 < n
    futurePart = fullTrajectory(bestStart + size(incompleteTrajectory,1) : end, :);
else
    futurePart = [];
end

%% ===== 新增：遍历文件中其他所有航段，寻找全局最优且未来部分非空的匹配 =====
bestGlobalDist = inf;
bestGlobalTrajectory = [];
bestGlobalStart = [];
bestGlobalTargetPort = ''; % 保存候选的目标港口（字符型）

uniqueTrips = unique(data(:,{'ship_name','trip_id'}),'rows');
for i = 1:height(uniqueTrips)
    sID = uniqueTrips.ship_name(i);
    tID = uniqueTrips.trip_id(i);
    if sID == shipID && tID == tripID
        continue;
    end
    tempTraj = pickFullTrajectory(data, sID, tID);
    tempFullTrajectory = [tempTraj.lon, tempTraj.lat, ...

```

```

        tempTraj.sog, tempTraj.cog, ...
        tempTraj.sin_phase, tempTraj.cos_phase];
if size(tempFullTrajectory,1) < size(incompleteTrajectory,1)
    continue;
end
[d, tempBestStart, ~, ~] = subseq_dtw(incompleteTrajectory, tempFullTrajectory);
% 检查候选的匹配是否能提供未来部分
if tempBestStart + size(incompleteTrajectory,1) - 1 < size(tempFullTrajectory,1)
    if d < bestGlobalDist
        bestGlobalDist = d;
        bestGlobalTrajectory = tempFullTrajectory;
        bestGlobalStart = tempBestStart;
        % 保存该候选的目标港口，取该候选完整航迹表的最后一行的 leg_end_port_code
        bestGlobalTargetPort = tempTraj.leg_end_port_code{end};
    end
end
end
% 如果找到了满足条件的候选，则使用其匹配结果和目标港口；否则退回使用当前航段匹配结果
if ~isempty(bestGlobalTrajectory)
    alignedSegment = bestGlobalTrajectory(bestGlobalStart : bestGlobalStart +
size(incompleteTrajectory,1) - 1, :);
    futurePart = bestGlobalTrajectory(bestGlobalStart + size(incompleteTrajectory,1) :
end, :);
    usedTargetPort = bestGlobalTargetPort;
else
    usedTargetPort = trajectory.leg_end_port_code{end};
end
%% ===== 全局最优子序 DTW 匹配覆盖结束 =====

%% 保存结果变量（供后续处理）
incompleteTrajectoryOutput = incompleteTrajectory; % 残缺轨迹 Tx6
alignedSegmentOutput      = alignedSegment;        % 对齐部分 Tx6

%% 计算 6 个特征的 Z-score 归一化参数（基于完整航迹）
featureMean = mean(fullTrajectory, 1);
featureStd  = std(fullTrajectory, 0, 1);

%% 对已有变量进行 Z-score 标准化转换
incompleteTrajectoryNorm = (incompleteTrajectoryOutput - featureMean) ./ featureStd;
alignedSegmentNorm      = (alignedSegmentOutput - featureMean) ./ featureStd;
% 定义真实未来轨迹，用于评分：完整航迹中残缺轨迹之后的部分（不改变）
trueFuturePart = fullTrajectory(endIdx+1:end, :);
% 用于训练输入的未来轨迹：匹配到的最佳航迹对齐部分之后的未来轨迹（变量 futurePart）
if ~isempty(futurePart)
    futurePartForTraining = futurePart;
else
    futurePartForTraining = [];
end
if ~isempty(futurePartForTraining)

```

```

        futurePartForTrainingNorm = (futurePartForTraining - featureMean) ./ featureStd;
else
    futurePartForTrainingNorm = [];
end

%% 构造归一化后的引导段（guideSegmentNorm），使行数与残缺轨迹一致
% 改进：guideSegmentNorm 来自匹配到的最佳航迹对齐部分之后的未来轨迹（futurePart）
T1 = size(incompleteTrajectoryNorm, 1);
T2 = size(futurePartForTrainingNorm, 1);
if T2 >= T1
    guideSegmentNorm = futurePartForTrainingNorm(1:T1, :);
else
    guideSegmentNorm = [futurePartForTrainingNorm;
repmat(futurePartForTrainingNorm(end, :), T1 - T2, 1)];
end

%% 构造模型输入（归一化后）：横向拼接归一化后的残缺轨迹与引导段
X_input_norm = [incompleteTrajectoryNorm, guideSegmentNorm];

%% 构造训练样本（自监督训练）
XTrain = {X_input_norm'};
YTrain = {guideSegmentNorm'};

%% 构建轨迹补全预测网络
layers = [ ...
    sequenceInputLayer(12, 'Name', 'input')
    fullyConnectedLayer(64, 'Name', 'fc1')
    reluLayer('Name', 'relu1')
    fullyConnectedLayer(64, 'Name', 'fc2')
    tanhLayer('Name', 'tanh1')
    lstmLayer(64, 'OutputMode', 'sequence', 'Name', 'lstm')
    fullyConnectedLayer(6, 'Name', 'fc3')
    regressionLayer('Name', 'output')
];

%% 训练网络（自监督训练，由于样本较少，仅供演示）
options = trainingOptions('adam', ...
    'MaxEpochs', maxEpochs, ...
    'MiniBatchSize', 1, ...
    'GradientThreshold', 1, ...
    'InitialLearnRate', initialLearnRate, ...
    'Verbose', 0, ...
    'Plots', 'none');

net = trainNetwork(XTrain, YTrain, layers, options);

%% 预测部分（完成补全后反标准化转换）
X_input_cell = {X_input_norm'};
Y_pred_norm = predict(net, X_input_cell);

```

```

Y_pred_norm = Y_pred_norm{1}'; % 预测结果: T1×6 (归一化尺度)
Y_pred = Y_pred_norm .* featureStd + featureMean; % 反标准化

%% ----- 预测结果后处理: 裁剪或插值 -----
currentEndTimeStr = trajectory.slice_time(endIdx);
currentEndTime = datetime(currentEndTimeStr, 'InputFormat', 'yyyy/M/d HH:mm');

nextTripIdx = find((data.ship_name == shipID) & (data.trip_id > tripID));
if isempty(nextTripIdx)
    warning('未找到同一船舶的下一个航段数据! 跳过裁剪, 使用全部预测结果');
    Y_pred_adjusted = Y_pred;
else
    nextTripData = data(nextTripIdx, :);
    nextTripData = sortrows(nextTripData, 'slice_time');
    nextStartTimeStr = nextTripData.slice_time(1);
    nextStartTime = datetime(nextStartTimeStr, 'InputFormat', 'yyyy/M/d HH:mm');

    timeGapHours = hours(nextStartTime - currentEndTime) - 3;

    predictedDuration = size(Y_pred,1) * sampleInterval;

    if predictedDuration > timeGapHours
        numKeep = floor(timeGapHours / sampleInterval);
        numKeep = min(numKeep, size(Y_pred,1));
        Y_pred_adjusted = Y_pred(1:numKeep, :);
    elseif predictedDuration < timeGapHours
        % 使用匹配到的最佳航段对应的目标港口 (usedTargetPort) 作为目标
        targetPortCode = usedTargetPort;
        portData = readtable('C:\Users\幻 15\Desktop\赛题数据 - 副本\港口静态信息数据.xls');
        portIdx = strcmp(portData.port_code, targetPortCode);
        if ~any(portIdx)
            error('未在港口静态信息中找到目标港口 %s', targetPortCode);
        end
        targetLon = portData.lon(portIdx);
        targetLat = portData.lat(portIdx);

        extraHours = timeGapHours - predictedDuration;
        numExtra = ceil(extraHours / sampleInterval);

        t_existing = (0:sampleInterval:(predictedDuration - sampleInterval))';
        t_new = (predictedDuration:sampleInterval:(predictedDuration + (numExtra-
1)*sampleInterval))';

        lastPredLon = Y_pred(end, 1);
        lastPredLat = Y_pred(end, 2);

        maxTime = predictedDuration + sampleInterval;

```

```

        interpLon = interp1([t_existing(end); maxTime], [Y_pred(end,1); targetLon], t_new,
'linear', 'extrap');
        interpLat = interp1([t_existing(end); maxTime], [Y_pred(end,2); targetLat], t_new,
'linear', 'extrap');
        % 对超出插值区间的查询点，直接赋值为目标港口经纬度
        exceedIdx = t_new > maxTime;
        interpLon(exceedIdx) = targetLon;
        interpLat(exceedIdx) = targetLat;

        interpOther = repmat(Y_pred(end, 3:6), numExtra, 1);

        extraPred = [interpLon, interpLat, interpOther];
        disp(extraPred);

        Y_pred_adjusted = [Y_pred; extraPred];
    else
        Y_pred_adjusted = Y_pred;
    end
    fullTrajectoryPrediction = [incompleteTrajectoryOutput; Y_pred_adjusted];

%% ----- 评分计算 -----
predPoints = Y_pred_adjusted(:, 1:2);
truePoints = trueFuturePart(:, 1:2); % 评分依旧使用真实未来轨迹
P = size(predPoints, 1);
Q = size(truePoints, 1);

predTimestamps = currentEndTime + hours(sampleInterval*(1:P));
trueTimestamps = currentEndTime + hours(sampleInterval*(1:Q));

if P > Q
    extraCount = P - Q;
    fprintf('\n 预测点多出 %d 个点\n', extraCount);
elseif Q > P
    missingCount = Q - P;
    fprintf('\n 预测点少了 %d 个点\n', missingCount);
else
    fprintf('\n 预测点数量与真实点数量一致\n');
end

if P == 0
    D1 = 0;
    D2 = 0;
    F = 0;
else
    numMatched = min(P, Q);
    D1 = 0;
    for i = 1:numMatched
        d = norm(truePoints(i,:) - predPoints(i,:));
        D1 = D1 + d;
    end
end

```



```

end

D2 = 0;
if P > Q
    for i = Q+1:P
        d = norm(predPoints(i,:));
        D2 = D2 + d;
    end
elseif Q > P
    for i = P+1:Q
        d = norm(truePoints(i,:));
        D2 = D2 + d;
    end
end

F = 100 / (1 + D1 + D2);
end

fprintf('\n 总匹配距离 D1 = %.4f\n', D1);
fprintf(' 未匹配距离 D2 = %.4f\n', D2);
fprintf(' 最终评分 F = %.4f\n', F);

avgDTW = minDist / size(incompleteTrajectory, 1); % 计算并输出 DTW 平均距离
fprintf(' 匹配航段的 DTW 平均距离: %.4f\n', avgDTW); % 新增输出
end

function trajectory = pickFullTrajectory(data, shipID, tripID)
    % pickFullTrajectory - 根据 shipID 与 tripID 挑选完整航迹
    idx = (data.ship_name == shipID) & (data.trip_id == tripID);
    trajectory = data(idx, :);
    trajectory = sortrows(trajectory, 'slice_time');
end

function [minDist, bestStart, bestPathQuery, bestPathRef] = subseq_dtw(query, reference)
    % 在 reference 中遍历所有连续子序列, 与 query 做 DTW, 寻找最小距离
    M = size(query,1);
    N = size(reference,1);
    minDist = Inf;
    bestStart = -1;
    bestPathQuery = [];
    bestPathRef = [];

    for k = 1:(N - M + 1)
        candidate = reference(k:(k+M-1), :);
        [d, pathQ, pathR] = dtw_full(query, candidate);
        if d < minDist
            minDist = d;
            bestStart = k;
        end
    end
end

```

```

        bestPathQuery = pathQ;
        bestPathRef = pathR;
    end
end
end

function [d, pathQ, pathR] = dtw_full(s, t)
    % 计算序列 s 和 t 的 DTW 距离及对齐路径
    Ns = size(s,1);
    Nt = size(t,1);
    D = inf(Ns+1, Nt+1);
    D(1,1) = 0;
    for i = 2:Ns+1
        for j = 2:Nt+1
            cost = norm(s(i-1,:) - t(j-1,:));
            D(i,j) = cost + min([D(i-1,j), D(i,j-1), D(i-1,j-1)]);
        end
    end
    d = D(Ns+1, Nt+1);

    i = Ns+1;
    j = Nt+1;
    pathQ = [];
    pathR = [];
    while (i > 1 || j > 1)
        pathQ = [i-1; pathQ];
        pathR = [j-1; pathR];
        if i == 1
            j = j - 1;
        elseif j == 1
            i = i - 1;
        else
            [~, idx] = min([D(i-1,j), D(i,j-1), D(i-1,j-1)]);
            switch idx
                case 1
                    i = i - 1;
                case 2
                    j = j - 1;
                case 3
                    i = i - 1;
                    j = j - 1;
            end
        end
    end
end
end
end

```

## 5.代码展示: LSTM\_PAR.mlx

```
%% 主程序
clc; clear; close all; tic;

%% 可调参数定义（均放在代码前面，不影响后续变量传递） trajectory = pickFullTrajectory
% 训练数据文件路径（用于完整航段匹配及后续处理）
filePath = 'C:\Users\幻 15\Desktop\Result\AIS 信息数据训练集_处理完成_TEST.csv';
% 采样间隔（小时），用于后续预测结果时间尺度的确定
sampleInterval = 3;
% 经纬度平移偏移量（已知地理范围：经度 [-180,180]、纬度 [-90,90]）
offsetLon = 180;
offsetLat = 90;

%% 读取训练数据（全局只读一次）
data = readtable(filePath);
% 将训练数据中的经纬度平移，使数值均为正
data.lon = data.lon + offsetLon;
data.lat = data.lat + offsetLat;

%% 预处理：计算相位的 sin 编码与 cos 编码（训练数据）
data.sin_phase = sin(deg2rad(data.hdg));
data.cos_phase = cos(deg2rad(data.hdg));

%% 新增：读取 TEST_1.csv，作为残缺轨迹构造的数据来源，并获取其中所有航段信息
testFilePath = 'C:\Users\幻 15\Desktop\Result\AIS 信息数据测试集_处理完成_TEST.csv';
testData = readtable(testFilePath);
% 对 TEST_1 数据同样平移经纬度
testData.lon = testData.lon + offsetLon;
testData.lat = testData.lat + offsetLat;
testData.sin_phase = sin(deg2rad(testData.hdg));
testData.cos_phase = cos(deg2rad(testData.hdg));

%% 新增：预先读取港口静态信息数据
portData = readtable('C:\Users\幻 15\Desktop\赛题数据 - 副本\港口静态信息数据.xls');
% 对港口数据进行经纬度平移处理
portData.lon = portData.lon + offsetLon;
portData.lat = portData.lat + offsetLat;

% 主循环基于 TEST_1 中的 (ship_name, trip_id) 组合进行遍历
uniqueVoyages = unique(testData(:, {'ship_name', 'trip_id'}), 'rows');
uniqueVoyages = sortrows(uniqueVoyages, 'trip_id');
```

```

% 参数将在每个航段内根据步长范围动态设置
% 初始化输出 cell 数组，用于累积每个航段预测结果
numVoyages = height(uniqueVoyages);
outputCell = cell(numVoyages, 1);

%% 并行循环遍历 TEST_1 中的每个航段（基于 ship_name 与 trip_id）
parfor i = 1:numVoyages
    shipID = uniqueVoyages.ship_name(i);
    tripID = uniqueVoyages.trip_id(i);

    % 从 TEST_1 中获取当前航段完整轨迹（用于步长计算）
    trajectory = pickFullTrajectory(testData, shipID, tripID);
    if isempty(trajectory)
        warning('TEST_1 中船舶 %d 航段 %d 无完整轨迹数据，跳过!', shipID, tripID);
        outputCell{i} = table();
        continue;
    end
    % 以当前航段数据行数作为步长
    stepLength = height(trajectory);

    %% 根据步长范围选择参数
    if stepLength >= 1 && stepLength <= 13
        startRemovalPct = 0.1742;
        endRemovalPct = 0.4527;
        initialLearnRate = 9.6389e-3;
        maxEpochs = 906;
    elseif stepLength >= 14 && stepLength <= 25
        startRemovalPct = 0.0732;
        endRemovalPct = 0.2503;
        initialLearnRate = 2.8266e-4;
        maxEpochs = 1499;
    elseif stepLength >= 26 && stepLength <= 33
        startRemovalPct = 0.2579;
        endRemovalPct = 0.2001;
        initialLearnRate = 2.5843e-4;
        maxEpochs = 745;
    elseif stepLength >= 34 && stepLength <= 41
        startRemovalPct = 0.1270;
        endRemovalPct = 0.2213;
        initialLearnRate = 2.4824e-4;
        maxEpochs = 1498;
    elseif stepLength >= 42 && stepLength <= 51
        startRemovalPct = 0.0503;
        endRemovalPct = 0.2010;
        initialLearnRate = 6.9885e-4;
        maxEpochs = 1241;
    elseif stepLength >= 52 && stepLength <= 66
        startRemovalPct = 0.2980;

```

```

        endRemovalPct = 0.2028;
        initialLearnRate = 1.7381e-4;
        maxEpochs = 1490;
elseif stepLength >= 67 && stepLength <= 83
    startRemovalPct = 0.0718;
    endRemovalPct = 0.2000;
    initialLearnRate = 2.0034e-3;
    maxEpochs = 772;
elseif stepLength >= 84
    startRemovalPct = 0.1995;
    endRemovalPct = 0.2414;
    initialLearnRate = 1.8454e-3;
    maxEpochs = 1531;
else
    error('无效的步长: %d', stepLength);
end

% 调用预测函数, 传入预先加载的 testData 与 portData
[F, incompleteTable, Y_pred_adjusted, currentEndTime] = runPrediction(shipID, tripID,
stepLength, ...
    startRemovalPct, endRemovalPct, initialLearnRate, maxEpochs, data, sampleInterval,
testData, portData);
% 此处评分逻辑已去除, 不输出 F

%% 构造预测部分的时间戳
numPred = size(Y_pred_adjusted,1);
predictedTimes = currentEndTime + hours(sampleInterval*(1:numPred));

%% 构造预测结果的表格 (输出字段保持不变)
predictedTable = table();
predictedTable.ship_name = repmat(shipID, numPred, 1);
predictedTable.slice_time = predictedTimes; % datetime 类型
% 在输出前将预测结果的经纬度转换回来 (减去偏移量)
predictedTable.lon = Y_pred_adjusted(:,1) - offsetLon;
predictedTable.lat = Y_pred_adjusted(:,2) - offsetLat;
predictedTable.hdg = nan(numPred,1);
predictedTable.sog = nan(numPred,1);
predictedTable.cog = nan(numPred,1);
predictedTable.draught = nan(numPred,1);
predictedTable.status = repmat({''}, numPred, 1);
predictedTable.trip_id = repmat(tripID, numPred, 1);

%% 选取残缺航段既有部分 (字段顺序不变), 并转换回原始经纬度
incompleteTableOut = incompleteTable(:, {'ship_name', 'slice_time', 'lon', 'lat',
'hdg', 'sog', 'cog', 'draught', 'status', 'trip_id'});
if ~isdatetime(incompleteTableOut.slice_time)
    incompleteTableOut.slice_time = datetime(incompleteTableOut.slice_time,
'InputFormat', 'yyyy/M/d HH:MM');
end

```

```

incompleteTableOut.lon = incompleteTableOut.lon - offsetLon;
incompleteTableOut.lat = incompleteTableOut.lat - offsetLat;

%% 保证 'status' 字段数据类型一致
if ~iscell(incompleteTableOut.status)
    predictedTable.status = repmat(NaN, numPred, 1);
end

%% 合并残缺航段和预测补全部分，按顺序追加
currentOutputTable = [incompleteTableOut; predictedTable];
outputCell{i} = currentOutputTable;

fprintf('调试信息：完成第 %d 个 TEST 航段的补全（船舶 %d 航段 %d）。\n', i, shipID, tripID);
end

%% 合并所有航段的预测结果
fullOutputTable = vertcat(outputCell{:});

%% 写入 CSV 文件（所有航段结果存入同一文件）
outputFilePath = 'C:\Users\幻 15\Desktop\Result\预测结果.csv';
writetable(fullOutputTable, outputFilePath);
fprintf('\n所有预测结果已写入文件：%s\n', outputFilePath);
toc;

%% ----- 辅助函数部分 -----

```

```

function [F, incompleteTable, Y_pred_adjusted, currentEndTime] = runPrediction(shipID,
tripID, stepLength, startRemovalPct, endRemovalPct, initialLearnRate, maxEpochs, data,
sampleInterval, testData, portData)
    % runPrediction - 对指定船舶(shipID)与航段(tripID)进行预测
    % 此处统一数据来源：残缺轨迹基于传入的 testData 构造，其时间与特征映射到训练数据尺度，
    % 而完整航迹（用于 DTW 匹配、归一化、引导段及其它后续处理）仍取自训练数据。fullTrajectory
    %
    % 新增要求：评分部分已去除。
    %
    % 输出：
    % F - （已去除评分逻辑）返回空数组
    % incompleteTable - 残缺航段对应的原始数据表（来自 TEST_1）
    % Y_pred_adjusted - 预测补全部分（反标准化后的矩阵）
    % currentEndTime - 残缺航段最后一个时间点（从 TEST_1 中提取）

    % 定义平移偏移量（与主程序保持一致）
    offsetLon = 180;
    offsetLat = 90;

```



```

%% 【基于 TEST_1 构造残缺轨迹】
% 直接使用传入的 testData
testFullTrajectory = pickFullTrajectory(testData, shipID, tripID);

% 构造残缺轨迹：若行数超过 stepLength，则截取前 stepLength 行
if height(testFullTrajectory) >= stepLength
    incompleteTrajectoryTable = testFullTrajectory(1:stepLength, :);
else
    incompleteTrajectoryTable = testFullTrajectory;
end
% 构造残缺轨迹矩阵（6 个特征），用于 DTW 匹配（保持原始观测数据不扩展）
incompleteTrajectory = [incompleteTrajectoryTable.lon,
incompleteTrajectoryTable.lat, ...
                        incompleteTrajectoryTable.sog, incompleteTrajectoryTable.cog, ...
                        incompleteTrajectoryTable.sin_phase,
incompleteTrajectoryTable.cos_phase];

% 保存残缺航段对应的原始数据（均来自 TEST_1）
incompleteTable = incompleteTrajectoryTable;

%% 【构建真实未来轨迹（用于其它后续处理）来自 TEST_1】
if height(testFullTrajectory) > height(incompleteTrajectoryTable)
    trueFuturePart = testFullTrajectory(height(incompleteTrajectoryTable)+1:end, :);
else
    trueFuturePart = [];
end

%% ----- 全局匹配方案 -----
disp('采用全局最优匹配方案进行轨迹匹配');
bestGlobalDist = inf;
bestGlobalTrajectory = [];
bestGlobalStart = [];
bestGlobalTargetPort = '';

uniqueTrips = unique(data(:, {'ship_name', 'trip_id'}), 'rows');
for i = 1:height(uniqueTrips)
    sID = uniqueTrips.ship_name(i);
    tID = uniqueTrips.trip_id(i);
    tempTraj = pickFullTrajectory(data, sID, tID);
    tempFullTrajectory = [tempTraj.lon, tempTraj.lat, ...
                        tempTraj.sog, tempTraj.cog, ...
                        tempTraj.sin_phase, tempTraj.cos_phase];
    if size(tempFullTrajectory, 1) < size(incompleteTrajectory, 1)
        continue;
    end
    [d, tempBestStart, ~, ~] = subsequ_dtw(incompleteTrajectory, tempFullTrajectory);
    % 如果候选航段属于当前船舶，调整距离因子为 0.8
    if sID == shipID

```

```

        d_adjusted = d * 0.8;
    else
        d_adjusted = d;
    end
    if tempBestStart + size(incompleteTrajectory,1) - 1 < size(tempFullTrajectory,1)
        if d_adjusted < bestGlobalDist
            bestGlobalDist = d_adjusted;
            bestGlobalTrajectory = tempFullTrajectory;
            bestGlobalStart = tempBestStart;
            bestGlobalTargetPort = tempTraj.leg_end_port_code{end};
        end
    end
end

if isempty(bestGlobalTrajectory)
    error('全局匹配失败，无法完成轨迹补全。');
else
    alignedSegment = bestGlobalTrajectory(bestGlobalStart : bestGlobalStart +
size(incompleteTrajectory,1) - 1, :);
    if bestGlobalStart + size(incompleteTrajectory,1) - 1 <
size(bestGlobalTrajectory,1)
        futurePart = bestGlobalTrajectory(bestGlobalStart +
size(incompleteTrajectory,1) : end, :);
    else
        futurePart = [];
    end
    usedTargetPort = bestGlobalTargetPort;
    fullTrajectory = bestGlobalTrajectory;
end

%% ----- 再次全局最优 DTW 匹配 -----
bestGlobalDist = inf;
bestGlobalTrajectory = [];
bestGlobalStart = [];
bestGlobalTargetPort = '';

uniqueTrips = unique(data(:,{'ship_name','trip_id'}),'rows');
for i = 1:height(uniqueTrips)
    sID = uniqueTrips.ship_name(i);
    tID = uniqueTrips.trip_id(i);
    tempTraj = pickFullTrajectory(data, sID, tID);
    tempFullTrajectory = [tempTraj.lon, tempTraj.lat, ...
        tempTraj.sog, tempTraj.cog, ...
        tempTraj.sin_phase, tempTraj.cos_phase];
    if size(tempFullTrajectory,1) < size(incompleteTrajectory,1)
        continue;
    end
    [d, tempBestStart, ~, ~] = subseq_dtw(incompleteTrajectory, tempFullTrajectory);
    if tempBestStart + size(incompleteTrajectory,1) - 1 < size(tempFullTrajectory,1)

```

```

% 候选加权：如果属于当前船舶，距离乘 0.8
if sID == shipID
    d_adjusted = d * 0.8;
else
    d_adjusted = d;
end
if d_adjusted < bestGlobalDist
    bestGlobalDist = d_adjusted;
    bestGlobalTrajectory = tempFullTrajectory;
    bestGlobalStart = tempBestStart;
    bestGlobalTargetPort = tempTraj.leg_end_port_code{end};
end
end
end
if ~isempty(bestGlobalTrajectory)
    alignedSegment = bestGlobalTrajectory(bestGlobalStart : bestGlobalStart +
size(incompleteTrajectory,1) - 1, :);
    futurePart = bestGlobalTrajectory(bestGlobalStart + size(incompleteTrajectory,1) :
end, :);
    usedTargetPort = bestGlobalTargetPort;
else
    usedTargetPort = trajectory.leg_end_port_code{end};
end

%% ----- 特征映射（归一化） -----
% 使用训练数据统计量对 TEST_1 构造的残缺轨迹进行映射
featureMean = mean(fullTrajectory, 1);
featureStd = std(fullTrajectory, 0, 1);

% 对残缺轨迹进行归一化（原始观测，用于匹配与后续扩展）
incompleteTrajectoryNorm = (incompleteTrajectory - featureMean) ./ featureStd;
alignedSegmentNorm = (alignedSegment - featureMean) ./ featureStd;
% “真实未来轨迹”采用的是 TEST_1 中的数据
if ~isempty(trueFuturePart)
    trueFuturePartMat = table2array(trueFuturePart(:, {'lon', 'lat'}));
else
    trueFuturePartMat = [];
end

%% ----- 构造归一化后的引导段（保留完整引导段信息） -----
% 对未来部分数据（用于引导）进行归一化处理
if ~isempty(futurePart)
    futurePartForTraining = futurePart;
    futurePartForTrainingNorm = (futurePartForTraining - featureMean) ./ featureStd;
else
    futurePartForTrainingNorm = [];
end
% 保留完整引导段，不进行裁剪
guideSegmentNorm = futurePartForTrainingNorm;

```

```

%% ----- 扩展残缺轨迹与引导段（要求 1） -----
% 使得残缺轨迹与引导段行数一致
T_orig = size(incompleteTrajectoryNorm,1);
T_target = size(guideSegmentNorm,1);
if T_orig < T_target
    % 扩展残缺轨迹
    numExtend = T_target - T_orig;
    extensionMatrix = repmat(incompleteTrajectoryNorm(end, :), numExtend, 1);
    incompleteTrajectoryNorm_extended = [incompleteTrajectoryNorm; extensionMatrix];
    indicator = [ones(T_orig,1); zeros(numExtend,1)];
elseif T_orig > T_target
    % 扩展引导段
    numExtend = T_orig - T_target;
    extensionGuide = repmat(guideSegmentNorm(end, :), numExtend, 1);
    guideSegmentNorm = [guideSegmentNorm; extensionGuide];
    incompleteTrajectoryNorm_extended = incompleteTrajectoryNorm;
    indicator = ones(T_orig,1);
else
    incompleteTrajectoryNorm_extended = incompleteTrajectoryNorm;
    indicator = ones(T_orig,1);
end

%% ----- 构造模型输入 -----
% 将扩展后的残缺轨迹、完整引导段以及扩展标识拼接成最终输入
X_input_norm = [incompleteTrajectoryNorm_extended, guideSegmentNorm, indicator];

%% 构造训练样本（自监督训练）
XTrain = {X_input_norm'};
YTrain = {guideSegmentNorm'};

%% 构建轨迹补全预测网络（输入维度由原来的 12 调整为 13）
layers = [ ...
    sequenceInputLayer(13, 'Name', 'input')
    fullyConnectedLayer(64, 'Name', 'fc1')
    reluLayer('Name', 'relu1')
    fullyConnectedLayer(64, 'Name', 'fc2')
    tanhLayer('Name', 'tanh1')
    lstmLayer(64, 'OutputMode', 'sequence', 'Name', 'lstm')
    fullyConnectedLayer(6, 'Name', 'fc3')
    regressionLayer('Name', 'output')
];

%% 训练网络（自监督训练，样本较少仅供演示）
options = trainingOptions('adam', ...
    'MaxEpochs', maxEpochs, ...
    'MiniBatchSize', 1, ...
    'GradientThreshold', 1, ...
    'InitialLearnRate', initialLearnRate, ...

```

```

    'Verbose', 0, ...
    'Plots', 'none');

net = trainNetwork(XTrain, YTrain, layers, options);

%% 预测（完成补全后反标准化转换）
X_input_cell = {X_input_norm};
Y_pred_norm = predict(net, X_input_cell);
Y_pred_norm = Y_pred_norm{1}';
Y_pred = Y_pred_norm .* featureStd + featureMean;

%% ----- 预测结果后处理：裁剪或插值 -----
% 当前残缺航段末尾时间从 TEST_1 中提取
if iscell(incompleteTrajectoryTable.slice_time)
    currentEndTimeStr = incompleteTrajectoryTable.slice_time{end};
else
    currentEndTimeStr = incompleteTrajectoryTable.slice_time(end);
end
if ~isdatetime(currentEndTimeStr)
    currentEndTime = datetime(currentEndTimeStr, 'InputFormat', 'yyyy/M/d HH:mm');
else
    currentEndTime = currentEndTimeStr;
end

% 使用 TEST_1 中的数据计算下一航段（nextTripIdx）
nextTripIdx = find((testData.ship_name == shipID) & (testData.trip_id > tripID));
if isempty(nextTripIdx)
    warning('船舶 %d 在 TEST_1 中无下一个航段数据！跳过裁剪，使用全部预测结果', shipID);
    Y_pred_adjusted = Y_pred;
else
    nextTripData = testData(nextTripIdx, :);
    nextTripData = sortrows(nextTripData, 'slice_time');
    nextStartTimeStr = nextTripData.slice_time(1);
    nextStartTime = datetime(nextStartTimeStr, 'InputFormat', 'yyyy/M/d HH:mm');

    timeGapHours = hours(nextStartTime - currentEndTime) - sampleInterval;
    predictedDuration = size(Y_pred,1) * sampleInterval;
    if predictedDuration > timeGapHours
        numKeep = floor(timeGapHours / sampleInterval);
        numKeep = min(numKeep, size(Y_pred,1));
        Y_pred_adjusted = Y_pred(1:numKeep, :);
    elseif predictedDuration < timeGapHours
        targetPortCode = usedTargetPort;
        % 使用传入的 portData 而非重新读取文件
        portIdx = strcmp(portData.port_code, targetPortCode);
        if ~any(portIdx)
            error('未在港口静态信息中找到目标港口 %s', targetPortCode);
        end
        targetLon = portData.lon(portIdx);
    end
end

```

```

        targetLat = portData.lat(portIdx);
        extraHours = timeGapHours - predictedDuration;
        numExtra = ceil(extraHours / sampleInterval);
        t_existing = (0:sampleInterval:(predictedDuration - sampleInterval));
        t_new = (predictedDuration:sampleInterval:(predictedDuration + (numExtra-
1)*sampleInterval));
        lastPredLon = Y_pred(end, 1);
        lastPredLat = Y_pred(end, 2);
        maxTime = predictedDuration + sampleInterval;
        interpLon = interp1([t_existing(end); maxTime], [Y_pred(end,1); targetLon],
t_new, 'linear', 'extrap');
        interpLat = interp1([t_existing(end); maxTime], [Y_pred(end,2); targetLat],
t_new, 'linear', 'extrap');
        exceedIdx = t_new > maxTime;
        interpLon(exceedIdx) = targetLon;
        interpLat(exceedIdx) = targetLat;
        interpOther = repmat(Y_pred(end, 3:6), numExtra, 1);
        extraPred = [interpLon, interpLat, interpOther];
        Y_pred_adjusted = [Y_pred; extraPred];
    else
        Y_pred_adjusted = Y_pred;
    end
end

%% ----- 去除评分计算 -----
F = [];
end

function trajectory = pickFullTrajectory(data, shipID, tripID)
    % 根据 ship_name 与 trip_id 挑选完整航迹
    idx = (data.ship_name == shipID) & (data.trip_id == tripID);
    trajectory = data(idx, :);
    trajectory = sortrows(trajectory, 'slice_time');
end

function [minDist, bestStart, bestPathQuery, bestPathRef] = subseq_dtw(query, reference)
    % 在 reference 中遍历所有连续子序列, 与 query 做 DTW, 寻找最小距离
    M = size(query,1);
    N = size(reference,1);
    minDist = Inf;
    bestStart = -1;
    bestPathQuery = [];
    bestPathRef = [];

    for k = 1:(N - M + 1)
        candidate = reference(k:(k+M-1), :);
        [d, pathQ, pathR] = dtw_full(query, candidate);
    end
end

```

```

        if d < minDist
            minDist = d;
            bestStart = k;
            bestPathQuery = pathQ;
            bestPathRef = pathR;
        end
    end
end

function [d, pathQ, pathR] = dtw_full(s, t)
    % 计算序列 s 和 t 的 DTW 距离及对齐路径
    Ns = size(s,1);
    Nt = size(t,1);
    D = inf(Ns+1, Nt+1);
    D(1,1) = 0;
    for i = 2:Ns+1
        for j = 2:Nt+1
            cost = norm(s(i-1,:) - t(j-1,:));
            D(i,j) = cost + min([D(i-1,j), D(i,j-1), D(i-1,j-1)]);
        end
    end
    d = D(Ns+1, Nt+1);

    i = Ns+1;
    j = Nt+1;
    pathQ = [];
    pathR = [];
    while (i > 1 || j > 1)
        pathQ = [i-1; pathQ];
        pathR = [j-1; pathR];
        if i == 1
            j = j - 1;
        elseif j == 1
            i = i - 1;
        else
            [~, idx] = min([D(i-1,j), D(i,j-1), D(i-1,j-1)]);
            switch idx
                case 1
                    i = i - 1;
                case 2
                    j = j - 1;
                case 3
                    i = i - 1;
                    j = j - 1;
            end
        end
    end
end
end
end

```



## 6.代码展示: LSTM\_ZHEN.mlx

```
%% 主程序
clc; clear; close all; tic;

%% 可调参数定义
filePath = 'C:\Users\幻 15\Desktop\Result\AIS 信息数据训练集_处理完成_TEST.csv';
sampleInterval = 3;
offsetLon = 180;
offsetLat = 90;

%% 读取训练数据
data = readtable(filePath);
data.lon = data.lon + offsetLon;
data.lat = data.lat + offsetLat;
data.sin_phase = sin(deg2rad(data.hdg));
data.cos_phase = cos(deg2rad(data.hdg));

%% 读取 TEST 数据
testFilePath = "C:\Users\幻 15\Desktop\Result\AIS 信息数据测试集_处理完成_TEST.csv";
testData = readtable(testFilePath);
testData.lon = testData.lon + offsetLon;
testData.lat = testData.lat + offsetLat;
testData.sin_phase = sin(deg2rad(testData.hdg));
testData.cos_phase = cos(deg2rad(testData.hdg));

%% 读取港口静态信息
portData = readtable('C:\Users\幻 15\Desktop\赛题数据 - 副本\港口静态信息数据.xls');
portData.lon = portData.lon + offsetLon;
portData.lat = portData.lat + offsetLat;

%% >>>> 只处理船舶 1 航段 1
shipID = 1;
tripID = 1;

trajectory = pickFullTrajectory(testData, shipID, tripID);
if isempty(trajectory)
    error('TEST 中船舶 1 航段 1 无数据');
end
```

```

stepLength = height(trajjectory);

%% 根据步长范围选择参数（原样保留）
if stepLength >= 1 && stepLength <= 13
    startRemovalPct = 0.1742; endRemovalPct = 0.4527;
    initialLearnRate = 9.6389e-3; maxEpochs = 906;
elseif stepLength >= 14 && stepLength <= 25
    startRemovalPct = 0.0732; endRemovalPct = 0.2503;
    initialLearnRate = 2.8266e-4; maxEpochs = 1499;
% ...其余分支不变
elseif stepLength >= 84
    startRemovalPct = 0.1995; endRemovalPct = 0.2414;
    initialLearnRate = 1.8454e-3; maxEpochs = 1531;
else
    error('无效的步长: %d', stepLength);
end

%% 执行补全预测
[F, incompleteTable, Y_pred_adjusted, currentEndTime] = runPrediction(shipID, tripID,
stepLength, ...
    startRemovalPct, endRemovalPct, initialLearnRate, maxEpochs, data, sampleInterval,
testData, portData);

numPred = size(Y_pred_adjusted,1);
predictedTimes = currentEndTime + hours(sampleInterval*(1:numPred));

%% 构造预测结果表
predictedTable = table();
predictedTable.ship_name = repmat(shipID, numPred, 1);
predictedTable.slice_time = predictedTimes;
predictedTable.lon = Y_pred_adjusted(:,1) - offsetLon;
predictedTable.lat = Y_pred_adjusted(:,2) - offsetLat;
predictedTable.hdg = nan(numPred,1);
predictedTable.sog = nan(numPred,1);
predictedTable.cog = nan(numPred,1);
predictedTable.draught = nan(numPred,1);
predictedTable.status = repmat({''}, numPred, 1);
predictedTable.trip_id = repmat(tripID, numPred, 1);

incompleteTableOut = incompleteTable(:, {'ship_name', 'slice_time', 'lon', 'lat', 'hdg',
'sog', 'cog', 'draught', 'status', 'trip_id'});
if ~isdatetime(incompleteTableOut.slice_time)
    incompleteTableOut.slice_time = datetime(incompleteTableOut.slice_time, 'InputFormat',
'yyyy/M/d HH:MM');
end

```

```
incompleteTableOut.lon = incompleteTableOut.lon - offsetLon;
incompleteTableOut.lat = incompleteTableOut.lat - offsetLat;
```

%% >>>> 调用绘图函数：关键帧图像生成

```
drawCompletionFrames(incompleteTableOut, predictedTable, sampleInterval);
```

```
toc;
```

```
function [F, incompleteTable, Y_pred_adjusted, currentEndTime] = runPrediction(shipID,
tripID, stepLength, startRemovalPct, endRemovalPct, initialLearnRate, maxEpochs, data,
sampleInterval, testData, portData)
    % runPrediction - 对指定船舶(shipID)与航段(tripID)进行预测
    % 此处统一数据来源：残缺轨迹基于传入的 testData 构造，其时间与特征映射到训练数据尺度，
    % 而完整航迹（用于 DTW 匹配、归一化、引导段及其它后续处理）仍取自训练数据。fullTrajectory
    %
    % 新增要求：评分部分已去除。
    %
    % 输出：
    %   F - （已去除评分逻辑）返回空数组
    %   incompleteTable - 残缺航段对应的原始数据表（来自 TEST_1）
    %   Y_pred_adjusted - 预测补全部分（反标准化后的矩阵）
    %   currentEndTime - 残缺航段最后一个时间点（从 TEST_1 中提取）

    % 定义平移偏移量（与主程序保持一致）
    offsetLon = 180;
    offsetLat = 90;

    %% 【基于 TEST_1 构造残缺轨迹】
    % 直接使用传入的 testData
    testFullTrajectory = pickFullTrajectory(testData, shipID, tripID);

    % 构造残缺轨迹：若行数超过 stepLength，则截取前 stepLength 行
    if height(testFullTrajectory) >= stepLength
        incompleteTrajectoryTable = testFullTrajectory(1:stepLength, :);
    else
        incompleteTrajectoryTable = testFullTrajectory;
    end
    % 构造残缺轨迹矩阵（6 个特征），用于 DTW 匹配（保持原始观测数据不扩展）
    incompleteTrajectory = [incompleteTrajectoryTable.lon,
incompleteTrajectoryTable.lat, ...
                           incompleteTrajectoryTable.sog, incompleteTrajectoryTable.cog, ...
                           incompleteTrajectoryTable.sin_phase,
incompleteTrajectoryTable.cos_phase];
```

```

% 保存残缺航段对应的原始数据（均来自 TEST_1）
incompleteTable = incompleteTrajectoryTable;

%% 【构建真实未来轨迹（用于其它后续处理）来自 TEST_1】
if height(testFullTrajectory) > height(incompleteTrajectoryTable)
    trueFuturePart = testFullTrajectory(height(incompleteTrajectoryTable)+1:end, :);
else
    trueFuturePart = [];
end

%% ----- 全局匹配方案 -----
disp('采用全局最优匹配方案进行轨迹匹配');
bestGlobalDist = inf;
bestGlobalTrajectory = [];
bestGlobalStart = [];
bestGlobalTargetPort = '';

uniqueTrips = unique(data(:, {'ship_name', 'trip_id'}), 'rows');
for i = 1:height(uniqueTrips)
    sID = uniqueTrips.ship_name(i);
    tID = uniqueTrips.trip_id(i);
    tempTraj = pickFullTrajectory(data, sID, tID);
    tempFullTrajectory = [tempTraj.lon, tempTraj.lat, ...
        tempTraj.sog, tempTraj.cog, ...
        tempTraj.sin_phase, tempTraj.cos_phase];
    if size(tempFullTrajectory,1) < size(incompleteTrajectory,1)
        continue;
    end
    [d, tempBestStart, ~, ~] = subsequ_dtw(incompleteTrajectory, tempFullTrajectory);
    % 如果候选航段属于当前船舶，调整距离因子为 0.8
    if sID == shipID
        d_adjusted = d * 0.8;
    else
        d_adjusted = d;
    end
    if tempBestStart + size(incompleteTrajectory,1) - 1 < size(tempFullTrajectory,1)
        if d_adjusted < bestGlobalDist
            bestGlobalDist = d_adjusted;
            bestGlobalTrajectory = tempFullTrajectory;
            bestGlobalStart = tempBestStart;
            bestGlobalTargetPort = tempTraj.leg_end_port_code{end};
        end
    end
end

if isempty(bestGlobalTrajectory)
    error('全局匹配失败，无法完成轨迹补全。');
else

```

```

        alignedSegment = bestGlobalTrajectory(bestGlobalStart : bestGlobalStart +
size(incompleteTrajectory,1) - 1, :);
        if bestGlobalStart + size(incompleteTrajectory,1) - 1 <
size(bestGlobalTrajectory,1)
            futurePart = bestGlobalTrajectory(bestGlobalStart +
size(incompleteTrajectory,1) : end, :);
        else
            futurePart = [];
        end
        usedTargetPort = bestGlobalTargetPort;
        fullTrajectory = bestGlobalTrajectory;
    end

%% ----- 再次全局最优 DTW 匹配 -----
bestGlobalDist = inf;
bestGlobalTrajectory = [];
bestGlobalStart = [];
bestGlobalTargetPort = '';

uniqueTrips = unique(data(:,{'ship_name','trip_id'}),'rows');
for i = 1:height(uniqueTrips)
    sID = uniqueTrips.ship_name(i);
    tID = uniqueTrips.trip_id(i);
    tempTraj = pickFullTrajectory(data, sID, tID);
    tempFullTrajectory = [tempTraj.lon, tempTraj.lat, ...
        tempTraj.sog, tempTraj.cog, ...
        tempTraj.sin_phase, tempTraj.cos_phase];
    if size(tempFullTrajectory,1) < size(incompleteTrajectory,1)
        continue;
    end
    [d, tempBestStart, ~, ~] = subseq_dtw(incompleteTrajectory, tempFullTrajectory);
    if tempBestStart + size(incompleteTrajectory,1) - 1 < size(tempFullTrajectory,1)
        % 候选加权: 如果属于当前船舶, 距离乘 0.8
        if sID == shipID
            d_adjusted = d * 0.8;
        else
            d_adjusted = d;
        end
        if d_adjusted < bestGlobalDist
            bestGlobalDist = d_adjusted;
            bestGlobalTrajectory = tempFullTrajectory;
            bestGlobalStart = tempBestStart;
            bestGlobalTargetPort = tempTraj.leg_end_port_code{end};
        end
    end
end
if ~isempty(bestGlobalTrajectory)
    alignedSegment = bestGlobalTrajectory(bestGlobalStart : bestGlobalStart +
size(incompleteTrajectory,1) - 1, :);

```

```

        futurePart = bestGlobalTrajectory(bestGlobalStart + size(incompleteTrajectory,1) :
end, :);
        usedTargetPort = bestGlobalTargetPort;
    else
        usedTargetPort = trajectory.leg_end_port_code{end};
    end

%% ----- 特征映射（归一化） -----
% 使用训练数据统计量对 TEST_1 构造的残缺轨迹进行映射
featureMean = mean(fullTrajectory, 1);
featureStd = std(fullTrajectory, 0, 1);

% 对残缺轨迹进行归一化（原始观测，用于匹配与后续扩展）
incompleteTrajectoryNorm = (incompleteTrajectory - featureMean) ./ featureStd;
alignedSegmentNorm = (alignedSegment - featureMean) ./ featureStd;
% “真实未来轨迹”采用的是 TEST_1 中的数据
if ~isempty(trueFuturePart)
    trueFuturePartMat = table2array(trueFuturePart(:, {'lon', 'lat'}));
else
    trueFuturePartMat = [];
end

%% ----- 构造归一化后的引导段（保留完整引导段信息） -----
% 对未来部分数据（用于引导）进行归一化处理
if ~isempty(futurePart)
    futurePartForTraining = futurePart;
    futurePartForTrainingNorm = (futurePartForTraining - featureMean) ./ featureStd;
else
    futurePartForTrainingNorm = [];
end

% 保留完整引导段，不进行裁剪
guideSegmentNorm = futurePartForTrainingNorm;

%% ----- 扩展残缺轨迹与引导段（要求 1） -----
% 使得残缺轨迹与引导段行数一致
T_orig = size(incompleteTrajectoryNorm,1);
T_target = size(guideSegmentNorm,1);
if T_orig < T_target
    % 扩展残缺轨迹
    numExtend = T_target - T_orig;
    extensionMatrix = repmat(incompleteTrajectoryNorm(end, :), numExtend, 1);
    incompleteTrajectoryNorm_extended = [incompleteTrajectoryNorm; extensionMatrix];
    indicator = [ones(T_orig,1); zeros(numExtend,1)];
elseif T_orig > T_target
    % 扩展引导段
    numExtend = T_orig - T_target;
    extensionGuide = repmat(guideSegmentNorm(end, :), numExtend, 1);
    guideSegmentNorm = [guideSegmentNorm; extensionGuide];
    incompleteTrajectoryNorm_extended = incompleteTrajectoryNorm;

```

```

        indicator = ones(T_orig,1);
else
    incompleteTrajectoryNorm_extended = incompleteTrajectoryNorm;
    indicator = ones(T_orig,1);
end

%% ----- 构造模型输入 -----
% 将扩展后的残缺轨迹、完整引导段以及扩展标识拼接成最终输入
X_input_norm = [incompleteTrajectoryNorm_extended, guideSegmentNorm, indicator];

%% 构造训练样本（自监督训练）
XTrain = {X_input_norm'};
YTrain = {guideSegmentNorm'};

%% 构建轨迹补全预测网络（输入维度由原来的 12 调整为 13）
layers = [ ...
    sequenceInputLayer(13, 'Name', 'input')
    fullyConnectedLayer(64, 'Name', 'fc1')
    reluLayer('Name', 'relu1')
    fullyConnectedLayer(64, 'Name', 'fc2')
    tanhLayer('Name', 'tanh1')
    lstmLayer(64, 'OutputMode', 'sequence', 'Name', 'lstm')
    fullyConnectedLayer(6, 'Name', 'fc3')
    regressionLayer('Name', 'output')
];

%% 训练网络（自监督训练，样本较少仅供演示）
options = trainingOptions('adam', ...
    'MaxEpochs', maxEpochs, ...
    'MiniBatchSize', 1, ...
    'GradientThreshold', 1, ...
    'InitialLearnRate', initialLearnRate, ...
    'Verbose', 0, ...
    'Plots', 'none');

net = trainNetwork(XTrain, YTrain, layers, options);

%% 预测（完成补全后反标准化转换）
X_input_cell = {X_input_norm'};
Y_pred_norm = predict(net, X_input_cell);
Y_pred_norm = Y_pred_norm{1}';
Y_pred = Y_pred_norm .* featureStd + featureMean;

%% ----- 预测结果后处理：裁剪或插值 -----
% 当前残缺航段末尾时间从 TEST_1 中提取
if iscell(incompleteTrajectoryTable.slice_time)
    currentEndTimeStr = incompleteTrajectoryTable.slice_time{end};
else
    currentEndTimeStr = incompleteTrajectoryTable.slice_time(end);

```



```

end
if ~isdatetime(currentEndTimeStr)
    currentEndTime = datetime(currentEndTimeStr, 'InputFormat', 'yyyy/M/d HH:mm');
else
    currentEndTime = currentEndTimeStr;
end

% 使用 TEST_1 中的数据计算下一航段 (nextTripIdx)
nextTripIdx = find((testData.ship_name == shipID) & (testData.trip_id > tripID));
if isempty(nextTripIdx)
    warning('船舶 %d 在 TEST_1 中无下一个航段数据! 跳过裁剪, 使用全部预测结果', shipID);
    Y_pred_adjusted = Y_pred;
else
    nextTripData = testData(nextTripIdx, :);
    nextTripData = sortrows(nextTripData, 'slice_time');
    nextStartTimeStr = nextTripData.slice_time(1);
    nextStartTime = datetime(nextStartTimeStr, 'InputFormat', 'yyyy/M/d HH:mm');

    timeGapHours = hours(nextStartTime - currentEndTime) - sampleInterval;
    predictedDuration = size(Y_pred,1) * sampleInterval;
    if predictedDuration > timeGapHours
        numKeep = floor(timeGapHours / sampleInterval);
        numKeep = min(numKeep, size(Y_pred,1));
        Y_pred_adjusted = Y_pred(1:numKeep, :);
    elseif predictedDuration < timeGapHours
        targetPortCode = usedTargetPort;
        % 使用传入的 portData 而非重新读取文件
        portIdx = strcmp(portData.port_code, targetPortCode);
        if ~any(portIdx)
            error('未在港口静态信息中找到目标港口 %s', targetPortCode);
        end
        targetLon = portData.lon(portIdx);
        targetLat = portData.lat(portIdx);
        extraHours = timeGapHours - predictedDuration;
        numExtra = ceil(extraHours / sampleInterval);
        t_existing = (0:sampleInterval:(predictedDuration - sampleInterval));
        t_new = (predictedDuration:sampleInterval:(predictedDuration + (numExtra-1)*sampleInterval));
        lastPredLon = Y_pred(end, 1);
        lastPredLat = Y_pred(end, 2);
        maxTime = predictedDuration + sampleInterval;
        interpLon = interp1([t_existing(end); maxTime], [Y_pred(end,1); targetLon],
t_new, 'linear', 'extrap');
        interpLat = interp1([t_existing(end); maxTime], [Y_pred(end,2); targetLat],
t_new, 'linear', 'extrap');
        exceedIdx = t_new > maxTime;
        interpLon(exceedIdx) = targetLon;
        interpLat(exceedIdx) = targetLat;
        interpOther = repmat(Y_pred(end, 3:6), numExtra, 1);
    end
end

```

```

        extraPred = [interpLon, interpLat, interpOther];
        Y_pred_adjusted = [Y_pred; extraPred];
    else
        Y_pred_adjusted = Y_pred;
    end
end

%% ----- 去除评分计算 -----
F = [];
end

function trajectory = pickFullTrajectory(data, shipID, tripID)
    % 根据 ship_name 与 trip_id 挑选完整航迹
    idx = (data.ship_name == shipID) & (data.trip_id == tripID);
    trajectory = data(idx, :);
    trajectory = sortrows(trajectory, 'slice_time');
end

function [minDist, bestStart, bestPathQuery, bestPathRef] = subseq_dtw(query, reference)
    % 在 reference 中遍历所有连续子序列, 与 query 做 DTW, 寻找最小距离
    M = size(query,1);
    N = size(reference,1);
    minDist = Inf;
    bestStart = -1;
    bestPathQuery = [];
    bestPathRef = [];

    for k = 1:(N - M + 1)
        candidate = reference(k:(k+M-1), :);
        [d, pathQ, pathR] = dtw_full(query, candidate);
        if d < minDist
            minDist = d;
            bestStart = k;
            bestPathQuery = pathQ;
            bestPathRef = pathR;
        end
    end
end

function [d, pathQ, pathR] = dtw_full(s, t)
    % 计算序列 s 和 t 的 DTW 距离及对齐路径
    Ns = size(s,1);
    Nt = size(t,1);
    D = inf(Ns+1, Nt+1);
    D(1,1) = 0;
    for i = 2:Ns+1
        for j = 2:Nt+1

```

```

        cost = norm(s(i-1,:) - t(j-1,:));
        D(i,j) = cost + min([D(i-1,j), D(i,j-1), D(i-1,j-1)]);
    end
end
d = D(Ns+1, Nt+1);

i = Ns+1;
j = Nt+1;
pathQ = [];
pathR = [];
while (i > 1 || j > 1)
    pathQ = [i-1; pathQ];
    pathR = [j-1; pathR];
    if i == 1
        j = j - 1;
    elseif j == 1
        i = i - 1;
    else
        [~, idx] = min([D(i-1,j), D(i,j-1), D(i-1,j-1)]);
        switch idx
            case 1
                i = i - 1;
            case 2
                j = j - 1;
            case 3
                i = i - 1;
                j = j - 1;
        end
    end
end
end
end

```

```

function drawCompletionFrames(incompleteTable, predictedTable, sampleInterval)
    all_lons = [incompleteTable.lon; predictedTable.lon];
    all_lats = [incompleteTable.lat; predictedTable.lat];

    totalFrames = 4;
    steps = round(linspace(1, height(predictedTable), totalFrames));
    time_labels = cellstr(predictedTable.slice_time(steps));

    % 用最后一帧决定统一坐标范围
    lon_ref = [incompleteTable.lon; predictedTable.lon(1:steps(end))];
    lat_ref = [incompleteTable.lat; predictedTable.lat(1:steps(end))];
    margin = 0.05;
    x_lim = [min(lon_ref)-margin, max(lon_ref)+margin];
    y_lim = [min(lat_ref)-margin, max(lat_ref)+margin];

```

```

for i = 1:totalFrames
    % 创建窗口，支持拉伸缩放
    fig = figure('Name', sprintf('关键帧 %d', i), ...
        'Units', 'normalized', ...
        'Position', [0.2+(i-1)*0.05, 0.2-(i-1)*0.05, 0.5, 0.5], ...
        'Resize', 'on'); % 允许缩放

    hold on;

    % 橙色残缺轨迹
    plot(incompleteTable.lon, incompleteTable.lat, '-', ...
        'LineWidth', 2, 'Color', [1, 0.5, 0]);
    plot(incompleteTable.lon, incompleteTable.lat, 'o', ...
        'MarkerFaceColor', [1, 0.5, 0], 'MarkerEdgeColor', [1, 0.5, 0]);

    % 补全轨迹
    plot(predictedTable.lon(1:steps(i)), predictedTable.lat(1:steps(i)), ...
        'g--o', 'LineWidth', 1.5);

    % 红色箭头
    if steps(i) > 1
        quiver(predictedTable.lon(steps(i)-1), predictedTable.lat(steps(i)-1), ...
            predictedTable.lon(steps(i)) - predictedTable.lon(steps(i)-1), ...
            predictedTable.lat(steps(i)) - predictedTable.lat(steps(i)-1), ...
            0, 'r', 'MaxHeadSize', 1, 'LineWidth', 1.5);
    end

    % 图像视觉设置（无 axis equal，让其随窗口缩放）
    xlabel('Longitude'); ylabel('Latitude');
    title(sprintf('关键帧 %d\n时间: %s', i, time_labels{i}));
    legend('原始残缺段', '预测补全轨迹', '时间方向', 'Location', 'best');
    set(gcf, 'Color', 'w'); % 或者 [1 1 1]

    axis([x_lim y_lim]); % 固定坐标范围
    axis normal; % 允许纵横比自由伸缩
    grid on;
    box on;

    hold off;
end
end

```

## 7.代码展示: Geo\_5.mlx

```

clc; clear;
tic;

%% 数据读取
aisData = readtable("C:\Users\幻 15\Desktop\Result\AIS 信息数据测试集_处理完成_TEST.csv");
aisData.slice_time = datetime(aisData.slice_time, 'InputFormat', 'yyyy/M/d H:mm');
aisData = sortrows(aisData, {'ship_name', 'slice_time'});

% 新增：读取港口静态信息数据
portData = readtable("C:\Users\幻 15\Desktop\赛题数据 - 副本\港口静态信息数据.xls");

%% 半径 3km 范围内的经纬度去除
% 计算地球表面两点之间的距离，单位为 km
% Haversine 公式
haversine = @(lat1, lon1, lat2, lon2) ...
    2 * 6371 * asin(sqrt(sin(deg2rad(lat2-lat1)/2).^2 + cos(deg2rad(lat1)) .*
    cos(deg2rad(lat2)) .* sin(deg2rad(lon2-lon1)/2).^2));

% 目标经纬度
target_lat = -13.2305;
target_lon = -76.7341;

% 按 (ship_name, trip_id) 分组，找出每个分组的首行坐标
[G, ~] = findgroups(aisData.ship_name, aisData.trip_id);
firstRowIndex = splitapply(@(x) x(1), (1:height(aisData))', G);

% 计算每个分组第一个点与目标点的距离
badGroupMask = arrayfun(@(idx) haversine(aisData.lat(idx), aisData.lon(idx), target_lat,
target_lon) < 3, firstRowIndex);

% 找出需要删除的分组，将其整段删除
badGroups = find(badGroupMask);
for i = 1:length(badGroups)
    aisData(G == badGroups(i), :) = [];
end

%% 地图样式选择：类似图中效果
basemapType = 'colorterrain'; % 可试 grayland、

%% 经纬度调整函数
adjustLon = @(lon) arrayfun(@(x) x+360*(x<-160), lon);

```

```

%% 自定义配色（避免蓝/青）
baseColors = lines(20);
excluded = [1, 2, 3]; % 代表蓝、青
baseColors(excluded,:) = [];
customColors = repmat(baseColors, 20, 1); % 足量配色

%% 地理轨迹图（所有舰船及港口）
figure('Name','所有舰船轨迹图', 'Color', 'w');
geobasemap(basemapType);
hold on;

% 绘制所有港口（沿用代码 1 中的黑圈黄色点，无文字信息）
ports_adj_lon = adjustLon(portData.lon);
geoscatter(portData.lat, ports_adj_lon, 60, 'y', 'o', 'filled', 'MarkerEdgeColor', 'k',
'MarkerFaceAlpha', 0.7);

% 船舶轨迹图：基于 trip_id 划分航段
ship_ids = unique(aisData.ship_name);
color_idx = 1;

for s = 1:length(ship_ids)
    shipData = aisData(aisData.ship_name == ship_ids(s), :);
    trip_ids = unique(shipData.trip_id);

    for i = 1:length(trip_ids)
        segment = shipData(shipData.trip_id == trip_ids(i), :);
        adj_lon = adjustLon(segment.lon);

        % 安全索引颜色（避免越界）
        cidx = mod(color_idx - 1, size(customColors,1)) + 1;
        geoplot(segment.lat, adj_lon, '-', 'LineWidth', 1.5, 'Color',
customColors(cidx,:));
        color_idx = color_idx + 1;
    end
end

title('舰船轨迹图');
hold off;
toc;

```