**bump\_choose.py** 这个文件总的来说是对于从数据库或 CSV 文件中读取输油泵数据，并对泵的状态进行处理和计算。

**各个组成函数的说明：**

**open\_bump\_num() 函数说明文档**

**概述**

该函数用于查询当前各站的开泵情况，并返回一个DataFrame对象，其中行表示站点，列表示泵，值为0表示关闭，值为1表示开启。

**使用方法**

调用该函数即可获取当前各站的开泵情况DataFrame对象。

open\_bump\_data = open\_bump\_num()

print(open\_bump\_data)

**参数**

该函数无需传入任何参数。

**返回值**

返回一个DataFrame对象，表示当前各站的开泵情况。

**实现逻辑**

从数据库中获取泵数据。

对每个泵的数据进行处理，计算其平均值。

根据一定条件（泵压差和变频器频率）判断每个泵是否处于开启状态。

将开启状态的泵转换为权值，存储在DataFrame中返回。

**示例**

python

import pandas as pd

# 调用函数获取开泵情况数据

open\_bump\_data = open\_bump\_num()

# 打印结果

print(open\_bump\_data)

**get\_bumps\_data\_mysql() 函数说明文档**

概述

该函数用于从 MySQL 数据库中获取泵数据，并返回一个列表，其中包含每个站点的泵数据。

使用方法

调用该函数即可获取从 MySQL 数据库中读取的泵数据列表。

**示例**

bump\_data = get\_bumps\_data\_mysql()

print(bump\_data)

**参数**

该函数无需传入任何参数。

**返回值**

返回一个列表，其中包含每个站点的泵数据。每个站点的泵数据以列表形式存储在返回的列表中。

**实现逻辑**

连接到 MySQL 数据库。

执行 SQL 查询语句，从数据库中获取泵数据。

将查询结果解析为列表形式，并返回。

**示例：**

# 调用函数获取泵数据

bump\_data = get\_bumps\_data\_mysql()

# 打印结果

print(bump\_data)

get\_bumps\_data\_csv() 函数说明：

功能：从 CSV 文件中读取输油泵的数据。

输入参数：无。

返回值：一个二维列表，表示每个泵的数据。

示例用法：

bumps\_data = get\_bumps\_data\_csv()

get\_bumps\_open\_mysql() 函数说明：

功能：从 MySQL 数据库中获取每个站点泵的状态、编号。

输入参数：无。

返回值：一个 DataFrame 对象，包含站点编号、泵编号、泵名称和泵状态。

示例用法：

mysql\_data = get\_bumps\_open\_mysql()

get\_bumps\_open\_csv() 函数说明：

功能：从 CSV 文件中获取每个站点泵的状态和编号。

输入参数：无。

返回值：一个二维列表，表示每个站点泵的状态和编号。

示例用法：

bumps\_setting = get\_bumps\_open\_csv()

**bump\_power.py**

这个文件的目的是获取泵的实际功率。

下面是对这段代码的分析和生成的代码说明文档：

real\_bumpp\_power() 函数说明：

功能：获取泵的实际功率。

返回值：一个字典，包含每个泵的编号、名称、状态、是否开启、所属站点、线路类型。

依赖库：依赖于 numpy、pandas、math 和 pyomo.environ 库，以及 base.mysql\_conn 中的 get\_conn 函数。

细节：

定义了内部函数 get\_bump\_state()，用于从 MySQL 数据库中获取泵的状态信息。

在 get\_bump\_state() 函数中执行 SQL 查询，获取泵的编号、名称、状态、是否开启、所属站点和线路类型。将查询结果保存到一个字典中，并返回该字典作为实际功率数据。

real\_bumpp\_power() 函数说明：

功能：获取泵的实际功率。

输入参数：无。

返回值：一个字典，包含每个泵的编号、名称、状态、是否开启、所属站点、线路类型。

示例用法：

python

real\_power\_data = real\_bumpp\_power()

内部函数 get\_bump\_state() 说明：

功能：从 MySQL 数据库中获取泵的状态信息。

输入参数：无。

返回值：一个字典，包含每个泵的编号、名称、状态、是否开启、所属站点、线路类型。

示例用法：不直接调用，作为 real\_bumpp\_power() 函数的内部函数使用。

bump\_power.py主要用于处理和预测泵的压力数据，包括从数据库获取数据、应用不同的模型进行预测、计算修正值以及获取站点的调节系数和初始压力。

这个文件包含了多个函数，以下是对每个函数的详细说明，以及它们的作用和输入输出：

1. do\_predict\_1(flow)

功能: 一个简单的线性回归模型，用于根据流量 flow 预测压力。

输入: flow (float) - 流量值。

输出: (float) - 预测的压力值。

2. do\_predict(flow)

功能: 一个多元线性回归模型，用于根据不同站点和流量 flow 预测压力。

输入: flow (float) - 流量值。

输出: (dict) - 包含不同站点预测压力的字典。

3. predict\_fixed\_frequency\_bump\_pressure\_v1(flow, regularization, index)

功能: 预测工频泵的压力值，并应用正则化项。

输入: flow (float) - 流量值, regularization (float) - 正则化项, index (int) - 站点索引。

输出: (float) - 预测的压力值。

4. var\_do\_predict(flow, freq)

功能: 预测变频泵的压力值，考虑频率 freq 的影响。

输入: flow (float) - 流量值, freq (float) - 频率值。

输出: (dict) - 包含不同站点预测压力的字典。

5. predict\_var\_frequency\_bump\_pressure\_v1(flow, freq, regularization, index)

功能: 预测变频泵的压力值，并应用正则化项。

输入: flow (float) - 流量值, freq (float) - 频率值, regularization (float) - 正则化项, index (int) - 站点索引。

输出: (float) - 预测的压力值。

6. real\_pump\_press()

功能: 从数据库获取泵的实际压力数据，并进行处理。

输入: 无。

输出: (DataFrame) - 包含实际开启泵口压力数据的DataFrame。

7. fit\_corrected\_value(gpbeng\_nihe\_press)

功能: 计算工频泵的实际功率与拟合功率之间的差值均值，作为拟合修正值。

输入: gpbeng\_nihe\_press (DataFrame) - 包含拟合功率数据的DataFrame。

输出: (float) - 拟合修正值。

8. predict\_fixed\_frequency\_bump\_pressure\_v2(flow, regularization)

功能: 使用新的线性回归模型预测工频泵的压力值，并应用正则化项。

输入: flow (float) - 流量值, regularization (float) - 正则化项。

输出: (float) - 预测的压力值。

9. predict\_var\_frequency\_bump\_pressure\_v2(flow, freq, regularization)

功能: 使用新的线性回归模型预测变频泵的压力值，并应用正则化项。

输入: flow (float) - 流量值, freq (float) - 频率值, regularization (float) - 正则化项。

输出: (float) - 预测的压力值。

10. predict\_fixed\_frequency\_bump\_pressure\_v3(flow, in\_pressure, index)

功能: 使用 riyi\_bump\_pressure\_model 模型预测工频泵的压力值。

输入: flow (float) - 流量值, in\_pressure (float) - 入口压力值, index (int) - 站点索引。

输出: (float) - 预测的压力值。

11. predict\_var\_frequency\_bump\_pressure\_v3(flow, freq, in\_pressure, index)

功能: 使用 riyi\_bump\_pressure\_model 模型预测变频泵的压力值。

输入: flow (float) - 流量值, freq (float) - 频率值, in\_pressure (float) - 入口压力值, index (int) - 站点索引。

输出: (float) - 预测的压力值。

12. regularize(flow\_capacity)

功能: 获取每个站点的负反馈调节系数和初始压力。

输入: flow\_capacity (float) - 流量容量。

输出: (dict) - 包含每个站点负反馈调节系数的字典, init\_pressure (float) - 初始压力值。

13. calc\_reg(bump\_data, flow\_capacity, start\_pos, num\_bumps)

功能: 计算修正项。

输入: bump\_data (DataFrame) - 泵数据, flow\_capacity (float) - 流量容量, start\_pos (int) - 开始计算的位置, num\_bumps (int) - 泵的数量。

输出: (float) - 修正项, minimal\_init\_pressure (float) - 最小初始压力值。

**bump\_pressure.py 这个文件主要是对泵压力的预测。**

文件各个函数的说明

do\_predict\_1(flow)

目的: 根据给定的流量 flow 计算预测的压力值。

输入: flow (float) - 泵的流量。

输出: (float) - 预测的压力值。

do\_predict(flow)

目的: 提供一个字典，包含不同站点基于流量 flow 的预测压力值。

输入: flow (float) - 泵的流量。

输出: (dict) - 以站点索引为键，预测压力值为值的字典。

predict\_fixed\_frequency\_bump\_pressure\_v1(flow, regularization, index)

目的: 预测工频泵的压力值，并根据正则化项进行调整。

输入: flow (float) - 泵的流量，regularization (float) - 正则化项，index (int) - 站点索引。

输出: (float) - 调整后的预测压力值。

var\_do\_predict(flow, freq)

目的: 预测变频泵的压力值，并考虑频率 freq 的影响。

输入: flow (float) - 泵的流量，freq (float) - 泵的频率。

输出: (dict) - 以站点索引为键，预测压力值为值的字典。

predict\_var\_frequency\_bump\_pressure\_v1(flow, freq, regularization, index)

目的: 预测变频泵的压力值，并根据正则化项进行调整。

输入: flow (float) - 泵的流量，freq (float) - 泵的频率，regularization (float) - 正则化项，index (int) - 站点索引。

输出: (float) - 调整后的预测压力值。

real\_pump\_press()

目的: 从数据库获取实际泵压力数据，并进行处理。

输入: 无。

输出: (DataFrame) - 包含实际泵压力数据的DataFrame。

fit\_corrected\_value(gpbeng\_nihe\_press)

目的: 计算工频泵的实际功率与拟合功率之间的差值均值，作为拟合修正值。

输入: gpbeng\_nihe\_press (DataFrame) - 包含拟合功率数据的DataFrame。

输出: (float) - 拟合修正值。

predict\_fixed\_frequency\_bump\_pressure\_v2(flow, regularization)

目的: 使用新的线性回归模型预测工频泵的压力值，并应用正则化项。

输入: flow (float) - 泵的流量，regularization (float) - 正则化项。

输出: (float) - 调整后的预测压力值。

predict\_var\_frequency\_bump\_pressure\_v2(flow, freq, regularization)

目的: 使用新的线性回归模型预测变频泵的压力值，并应用正则化项。

输入: flow (float) - 泵的流量，freq (float) - 泵的频率，regularization (float) - 正则化项。

输出: (float) - 调整后的预测压力值。

predict\_fixed\_frequency\_bump\_pressure\_v3(flow, in\_pressure, index)

目的: 使用 riyi\_bump\_pressure\_model 模型预测工频泵的压力值。

输入: flow (float) - 泵的流量，in\_pressure (float) - 泵的入口压力，index (int) - 站点索引。

输出: (float) - 预测的压力值。

predict\_var\_frequency\_bump\_pressure\_v3(flow, freq, in\_pressure, index)

目的: 使用 riyi\_bump\_pressure\_model 模型预测变频泵的压力值。

输入: flow (float) - 泵的流量，freq (float) - 泵的频率，in\_pressure (float) - 泵的入口压力，index (int) - 站点索引。

输出: (float) - 预测的压力值。

regularize(flow\_capacity)

目的: 获取每个站点的负反馈调节系数和初始压力。

输入: flow\_capacity (float) - 泵的流量容量。

输出: (dict) - 包含每个站点负反馈调节系数的字典，init\_pressure (float) - 初始压力值。

calc\_reg(bump\_data, flow\_capacity, start\_pos, num\_bumps)

目的: 计算修正项。

输入: bump\_data (DataFrame) - 泵数据，flow\_capacity (float) - 流量容量，start\_pos (int) - 开始计算的位置，num\_bumps (int) - 泵的数量。

输出: (float) - 修正项，minimal\_init\_pressure (float) - 最小初始压力值。

**bump\_settings.py 这个文件提供对泵站数据进行管理。**

文件的各个函数说明

get\_max\_num\_bumps()

目的: 从数据库查询每个站点的泵数量。

输入: 无。

输出: (dict) - 以站点索引为键，泵数量为值的字典。

get\_started\_num\_bumps(threshold=0.3)

目的: 根据给定的阈值，从数据库查询每个站点开启的泵数量。

输入: threshold (float, 可选) - 泵开启状态的判定阈值，默认值为0.3。

输出: (dict) - 以站点索引为键，开启的泵数量为值的字典。

get\_bump\_power\_tag()

目的: 提供站点和泵序号对应的泵功率点位标签。

输入: 无。

输出: (dict) - 以站点和泵序号的元组为键，泵功率点位标签为值的字典。

get\_bump\_freq\_tag()

目的: 提供站点和泵序号对应的泵变频频率点位标签。

输入: 无。

输出: (dict) - 以站点和泵序号的元组为键，泵变频频率点位标签为值的字典。

get\_bump\_tag()

目的: 获取泵的信息，包括站id，泵id，功率点位，频率点位，入口压力点位，出口压力点位。

输入: 无。

输出: (dict) - 以泵id为键，包含泵相关信息的字典为值的字典。

get\_max\_num\_bumps\_from\_csv()

目的: 从CSV文件中获取每个站点的泵数量。

输入: 无。

输出: (dict) - 以站点索引为键，泵数量为值的字典。

get\_started\_num\_bumps\_from\_csv()

目的: 从CSV文件中获取每个站点开启的泵数量。

输入: 无。

输出: (dict) - 以站点索引为键，开启的泵数量为值的字典。

？？？？？？？？？

**Flow\_power\_fit.py 这个文件主要提供对泵站压力与流量工作数据分析与拟合程序。**

文件的函数说明

do\_predict()

目的: 使用预设的系数进行工频泵压力与流量的关系拟合。

输入: 无。

输出: (dict) - 以泵索引为键，包含拟合系数的列表为值的字典。

var\_do\_predict()

目的: 使用预设的系数进行变频泵功率与流量和频率的关系拟合。

输入: 无。

输出: (dict) - 以泵索引为键，包含拟合系数的列表为值的字典。

get\_conn()

目的: 获取数据库连接。

输入: 无。

输出: 数据库连接对象。

主要流程

数据获取: 程序首先定义了工频泵和变频泵的功率和压力标签列表，然后通过数据库连接获取相应标签的数据。

数据处理: 获取的数据通过时间戳进行排序和筛选，以确保数据的连续性和有效性。对于变频泵，程序还进行了额外的筛选，以确保只使用开启状态的泵的数据。

数据拟合: 对于工频泵，程序使用二次多项式对压力和流量的关系进行拟合。对于变频泵，程序使用一个更复杂的公式对功率、流量和频率的关系进行拟合。

结果展示: 拟合参数和拟合函数将被打印输出，同时使用matplotlib库绘制原始数据点和拟合曲线/曲面。

？？？？？？？？？？？？？？

**Flow\_press\_fit.py 这个文件是基于泵站的流量数据预测压力分析与拟合的程序。**

函数说明

do\_predict(flow)

目的: 定义一个用于拟合工频泵的函数，该函数基于流量 flow 预测压力。

输入: flow (float) - 泵的流量。

输出: (float) - 预测的压力值。

target\_function(xy, a, b, c, w)

目的: 定义一个用于拟合变频泵的函数，该函数基于流量 x 和频率 y 预测功率 w。

输入: xy (tuple) - 流量和频率的元组，a, b, c (float) - 拟合系数，w (float) - 变频泵的功率。

输出: (float) - 预测的功率值。

function\_to\_plot(flow, freq)

目的: 定义一个用于生成原始拟合函数数据的函数。

输入: flow (float) - 泵的流量，freq (float) - 泵的频率。

输出: (float) - 基于流量和频率计算的值。

主要流程

数据获取: 程序首先定义了工频泵和变频泵的压力标签列表，然后通过数据库连接获取相应标签的数据。

数据处理: 获取的数据通过时间戳进行排序和筛选，以确保数据的连续性和有效性。对于变频泵，程序还进行了额外的筛选，以确保只使用开启状态的泵的数据。

数据拟合: 对于工频泵，程序使用二次多项式对压力和流量的关系进行拟合。对于变频泵，程序使用一个更复杂的公式对功率、流量和频率的关系进行拟合。

结果展示: 拟合参数和拟合函数将被打印输出，同时使用matplotlib库绘制原始数据点和拟合曲线/曲面。

**Info\_to\_json.py 这个文件是将泵的启停状态、站点的电费信息、实际运行中的泵的数据整合为json格式。**

文件函数说明：

get\_bump\_state()

目的: 获取泵的启停状态。

输入: 无。

输出: (dict) - 以泵的ID为键，泵的启停状态（0表示关闭，1表示开启）为值的字典。

get\_station\_df()

目的: 获取站点的电费信息。

输入: 无。

输出: (DataFrame) - 包含站点ID、名称、电费和电费总额的DataFrame。

get\_real\_bump\_data()

目的: 获取实际运行中的泵的数据。

输入: 无。

输出: (DataFrame) - 包含泵的ID、名称、状态、功率和站点的DataFrame。

get\_info\_json()

目的: 将泵站数据和站点数据整合为JSON格式。

输入: 无。

输出: (str) - JSON格式的字符串，包含泵站和站点的相关信息。

主要流程

数据库连接: 程序通过get\_conn函数建立与数据库的连接

泵状态获取: 使用get\_bump\_state函数查询并返回泵的启停状态。

站点信息获取: 使用get\_station\_df函数查询站点的电费信息，并创建DataFrame。

实际泵数据获取: 使用get\_real\_bump\_data函数查询并返回实际运行中的泵的数据，并创建DataFrame。

数据整合与调整: 程序将泵数据和站点数据整合，并对数据进行必要的调整，例如修正东海站的电费总额

JSON格式输出: 使用get\_info\_json函数将整合后的数据转换为JSON格式，并去除字符串中的空格。

**Input\_to\_json.py 这个文件获取了管道流量与压力数据，最后转换成json数据。**

函数说明

get\_real\_in\_out\_press(line\_id=1)

目的: 获取实际的管道流量和压力数据。

输入: line\_id (int, 可选) - 指定获取数据的管线ID，默认为1。

输出: (DataFrame) - 包含流量、入口压力、管线类型、优化ID、出口压力、管道ID和管道名称的DataFrame。

get\_input\_json()

目的: 将管道流量和压力数据转换为JSON格式。

输入: 无。

输出: (str) - JSON格式的字符串，包含管道流量和压力的相关信息。

主要流程

数据库连接: 程序通过get\_conn函数建立与数据库的连接。

数据获取: 使用get\_real\_in\_out\_press函数查询并返回指定管线的流量和压力数据。

数据框架创建: 将查询结果创建为DataFrame，便于后续处理和分析。

JSON格式输出: 使用get\_input\_json函数将DataFrame中的数据转换为JSON格式，并去除字符串中的空格。

**inter\_station\_pressure\_difference\_fitting.py 这个文件主要是对站点出站压力、站点进站压力、流量数据进行站间压差公式拟合。**

函数说明

get\_conn()

目的: 获取数据库连接。

输入: 无。

输出: 数据库连接对象。

nonlinear\_func(xy, a, b, c, d)

目的: 定义非线性拟合函数。

输入: xy (tuple) - 自变量数据对，a, b, c, d (float) - 拟合系数。

输出: (float) - 根据拟合函数计算得到的值。

主要流程

数据库连接: 程序通过get\_conn函数建立与数据库的连接。

数据获取: 程序从数据库中获取指定管线在2022年和2023年的出站压力、进站压力和流量数据。

数据处理: 程序对获取的数据进行清洗和筛选，以确保数据的准确性和可用性。

数据拟合: 对每个管段的数据进行非线性拟合，得到压降计算公式。

结果展示: 程序打印出拟合后的系数和函数表达式，并通过3D图形展示拟合平面。

交互式视图: 使用ipywidgets库提供交互式3D图形视图

Output\_to\_json.py 这个文件

函数说明

get\_station\_bumps\_index()

目的: 获取站点与泵的索引映射。

输入: 无。

输出: (dict) - 以站点ID为键，泵ID列表为值的字典。

get\_json\_from\_file(file)

目的: 从文件中读取数据并转换为JSON格式。

输入: file (str) - 数据文件的路径。

输出: (str) - JSON格式的字符串。

get\_json\_from\_records(bump\_choice\_result, elec\_price)

目的: 从记录结果和电费价格列表中生成DataFrame并转换为JSON格式。

输入: bump\_choice\_result (list) - 泵的选择结果列表，elec\_price (list) - 电费价格列表。

输出: (str) - JSON格式的字符串。

get\_json\_from\_df(choice\_df, elec\_price)

目的: 从DataFrame中提取数据并转换为JSON格式。

输入: choice\_df (DataFrame) - 包含泵选择结果的DataFrame，elec\_price (list) - 电费价格列表。

输出: (str) - JSON格式的字符串。