盈德气体换热网络分析平台使用说明

1. 软件简介

本软件是为盈德气体荆门工厂煤制甲醇装置的合成工段及精馏工段所开发的专用软件，主要实现以下四个功能：

1) 夹点分析

通过数据模块接收装置流股信息，对现行网络匹配绘制温焓图，并进行夹点分析，得到换热网络最优匹配下的公用工程用量、节能潜力以及最大碳减排量。

2) 不合理换热分析

主要实现查找现行换热网络内的不合理换热器。

3) 负荷转移分析

由于催化剂失活等导致的生产条件变化会使换热器负荷发生变动，为了使换热流股能达到目标温度，我们增加了负荷转移分析模块。

当流股数据发生变动时，该模块可以检测负荷转移的所有可能路径，并计算可能发生的温度及负荷变化。

4) 换热网络设计

该模块通过热力学与随机方法相结合，对现行流股数据的换热网络自动设计，设计目标为公用工程用量小于现行换热网络公用工程用量。

2. 安装及运行环境

运行环境：Windows xp 系统及以上；CPU 600MHz 及以上；内存 512M 及以上；硬盘可用空间 1.5G 及以上。

安装：解压“盈德气体换热网络分析平台.zip”，在文件夹中找到“盈德气体换热网络分析平台.exe”，选择该文件添加到桌面快捷方式，即安装成功。

3. 使用说明

3.1 流程图页面

单击左侧“流程图”按钮即可切换至流程图页面。

页面内置相关工艺流程图，包括：甲醇合成装置流程图、甲醇精馏装置流程图、现行换热网络图像及现行换热网络有向图。通过图片两侧按钮可切换图片。

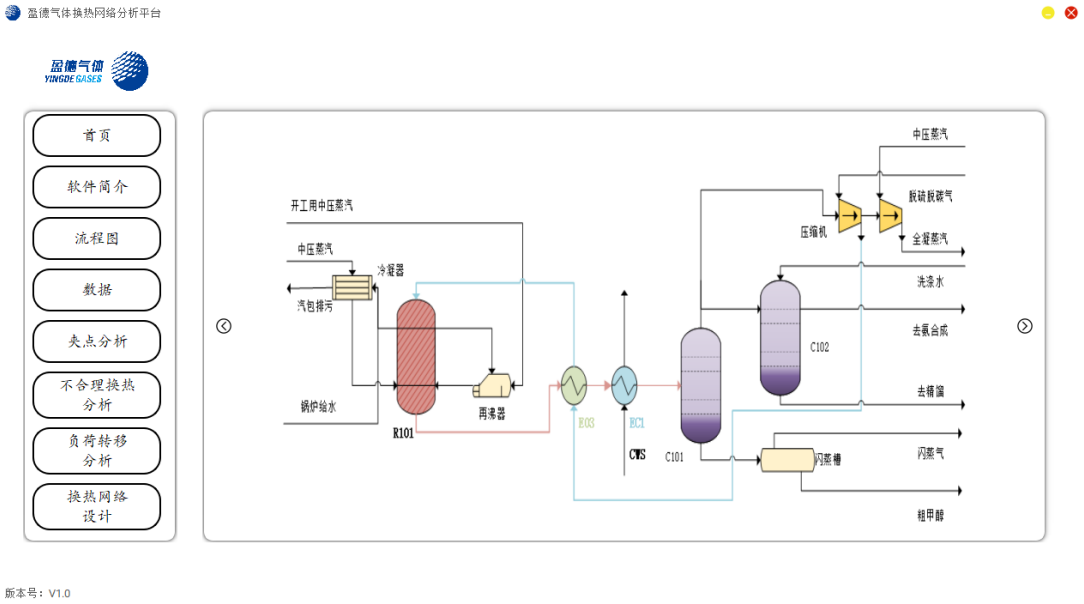


图 3-1 甲醇合成装置流程图页面

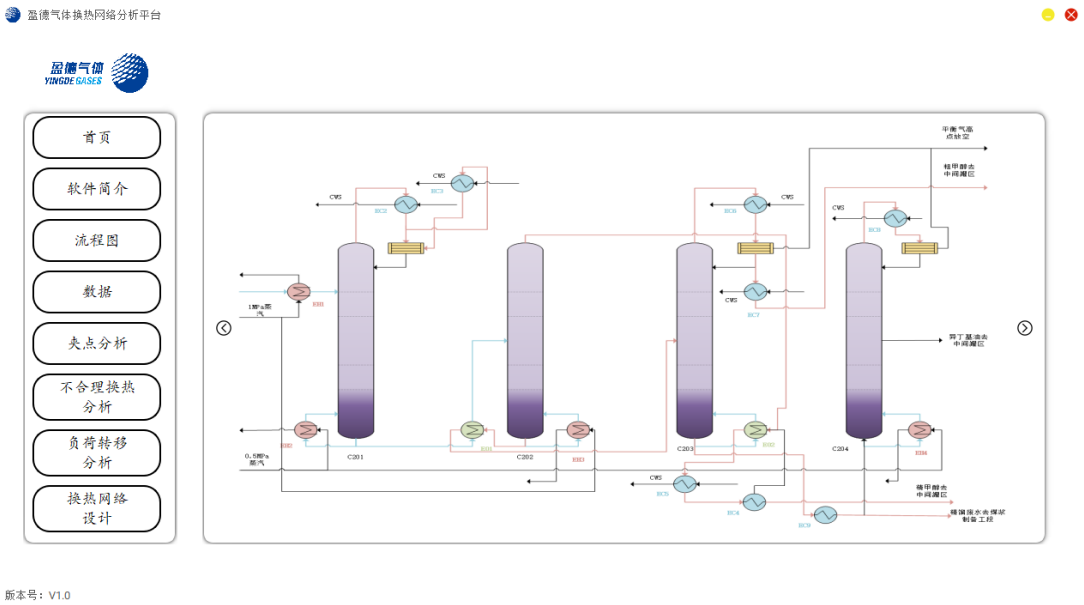


图 3-2 甲醇精馏装置流程图页面

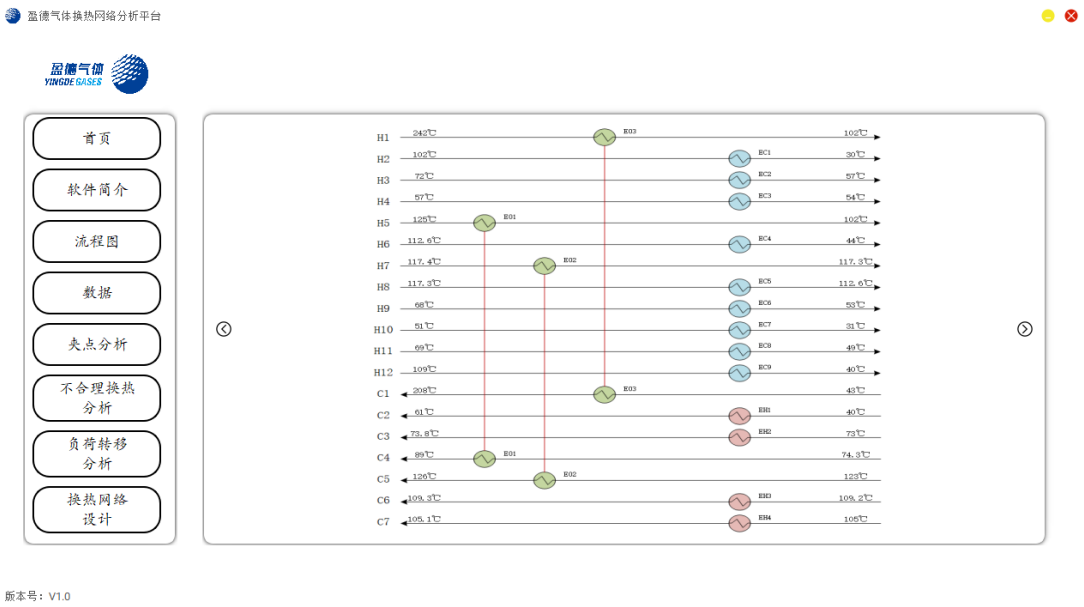


图 3-3 现行换热网络页面

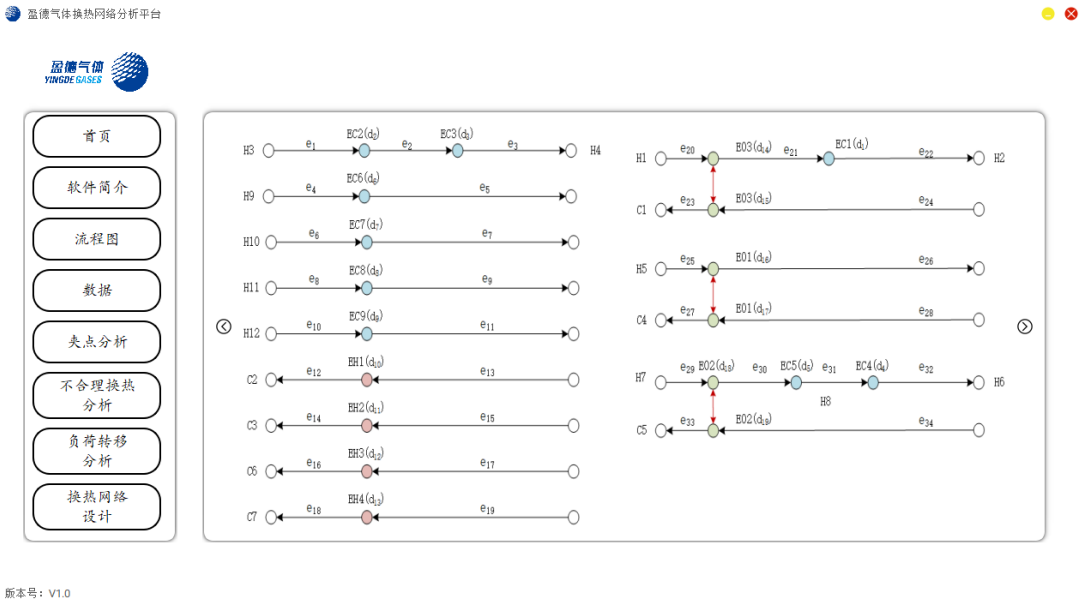


图 3-4 现行换热网络有向图页面

3.2 数据页面

软件内置现行甲醇合成及精馏装置流股数据，可对数据进行修改、上传、下载及恢复功能。

3.2.1 数据直接修改

1. 双击单元格，修改数据；

2. 回车，单元格变红，表示修改成功。

图 3-5 数据直接修改

3.2.2 清空数据

点击清空数据按钮，即可清空数据。

图 3-6 清空数据

3.2.3 上传文件

1. 清空数据；

2. 点击上传文件按钮；

3. 选择 json 类型文件上传。

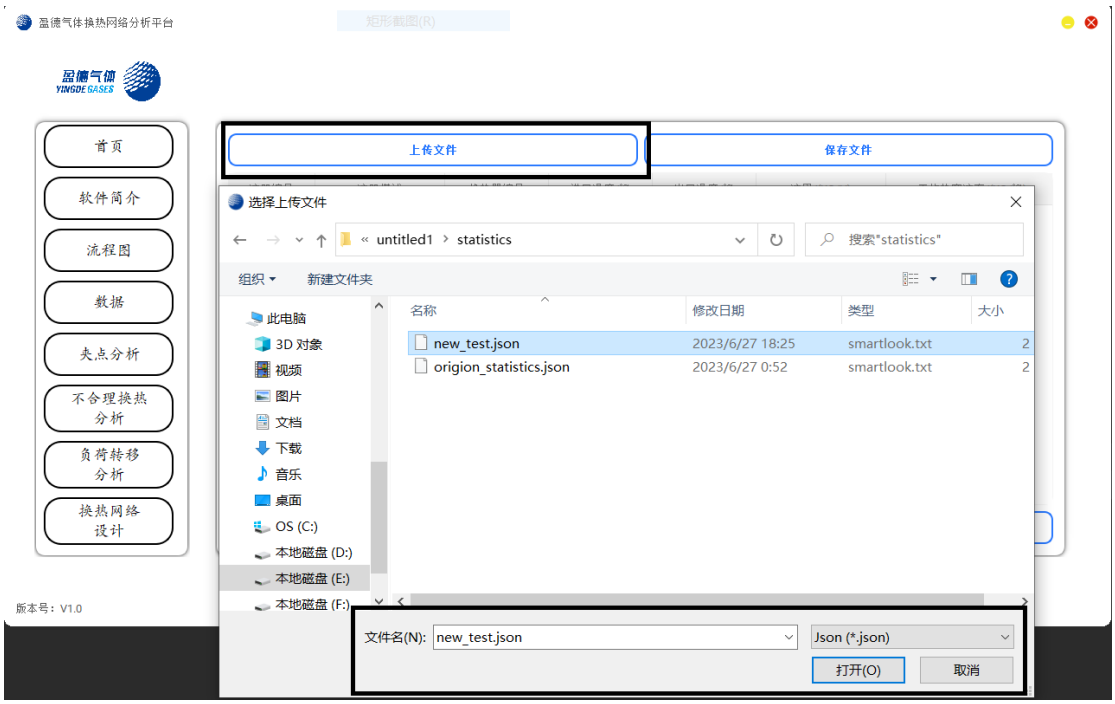


图 3-7 上传 json 类型文件

图 3-8 上传文件结果展示

Json 文件内部格式说明：

1. 手动输入 json 文件。

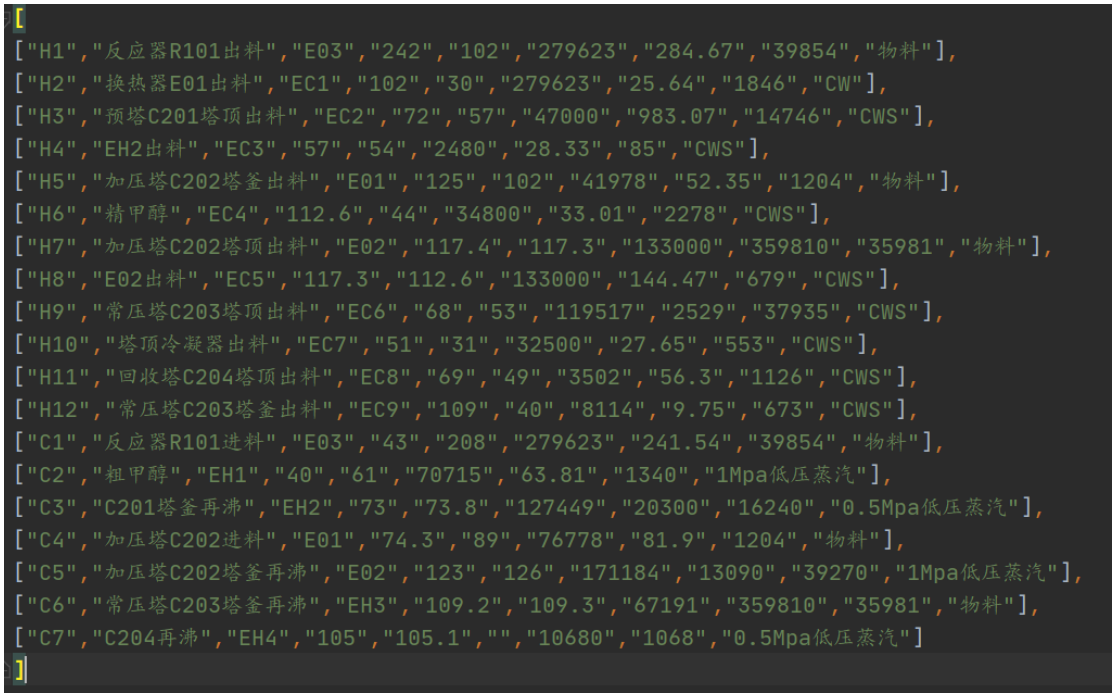
   1) 新建记事本文件（.txt）；

2) 每一行按照["流股编号","流股描述","换热器编号","进口温度/℃","出口温度/℃","流量/(kg/h)","平均热容流率/(kg/℃)","热负荷/kW","换热介质"]的格式输入；

3) 每行之间用“,”连接，最外面用[ ]包括，即[[第一行内容],[第二行内容],[第三行内容],……]；

4) 所有标点符号全为英文标点；

5) 重命名修改文件类型（.txt）为（.json）。



```
[
["H1","反应器R101出料","E03","242","102","279623","284.67","39854","物料"],
["H2","换热器E01出料","EC1","102","30","279623","25.64","1846","CW"],
["H3","预塔C201塔顶出料","EC2","72","57","47000","983.07","14746","CWS"],
["H4","EH2出料","EC3","57","54","2480","28.33","85","CWS"],
["H5","加压塔C202塔釜出料","E01","125","102","41978","52.35","1204","物料"],
["H6","精甲醇","EC4","112.6","44","34800","33.01","2278","CWS"],
["H7","加压塔C202塔顶出料","E02","117.4","117.3","133000","359810","35981","物料"],
["H8","E02出料","EC5","117.3","112.6","133000","144.47","679","CWS"],
["H9","常压塔C203塔顶出料","EC6","68","53","119517","2529","37935","CWS"],
["H10","塔顶冷凝器出料","EC7","51","31","32500","27.65","553","CWS"],
["H11","回收塔C204塔顶出料","EC8","69","49","3502","56.3","1126","CWS"],
["H12","常压塔C203塔釜出料","EC9","109","40","8114","9.75","673","CWS"],
["C1","反应器R101进料","E03","43","208","279623","241.54","39854","物料"],
["C2","粗甲醇","EH1","40","61","70715","63.81","1340","1Mpa低压蒸汽"],
["C3","C201塔釜再沸","EH2","73","73.8","127449","20300","16240","0.5Mpa低压蒸汽"],
["C4","加压塔C202进料","E01","74.3","89","76778","81.9","1204","物料"],
["C5","加压塔C202塔釜再沸","E02","123","126","171184","13090","39270","1Mpa低压蒸汽"],
["C6","常压塔C203塔釜再沸","EH3","109.2","109.3","67191","359810","35981","物料"],
["C7","C204再沸","EH4","105","105.1","","10680","1068","0.5Mpa低压蒸汽"]
]
```

图 3-9 手动输入 json 格式文件示意图

2. 由 excel 文件转 json 文件。

1) 按照["流股编号","流股描述","换热器编号","进口温度/℃","出口温度/℃","流量/(kg/h)","平均热容流率/(kg/℃)","热负荷/kW","换热介质"]的顺序，在 Excel 中输入每一行的内容；

2) 从百度中搜索 Excel 转 Json，推荐网址：https://wejson.cn/txt2json/。进入网址，选择 Excel 文本转 json 功能，选择 json 数组，是否格式化选择是；

3) 复制 Excel 到文本框内并粘贴，进行 json 数组转换；

4) 新建记事本文件（.txt），复制并粘贴转换结果至记事本内，重命名修改文件类型（.txt）为（.json）。

注：上传的新数据仅可于夹点分析及换热网络设计模块。

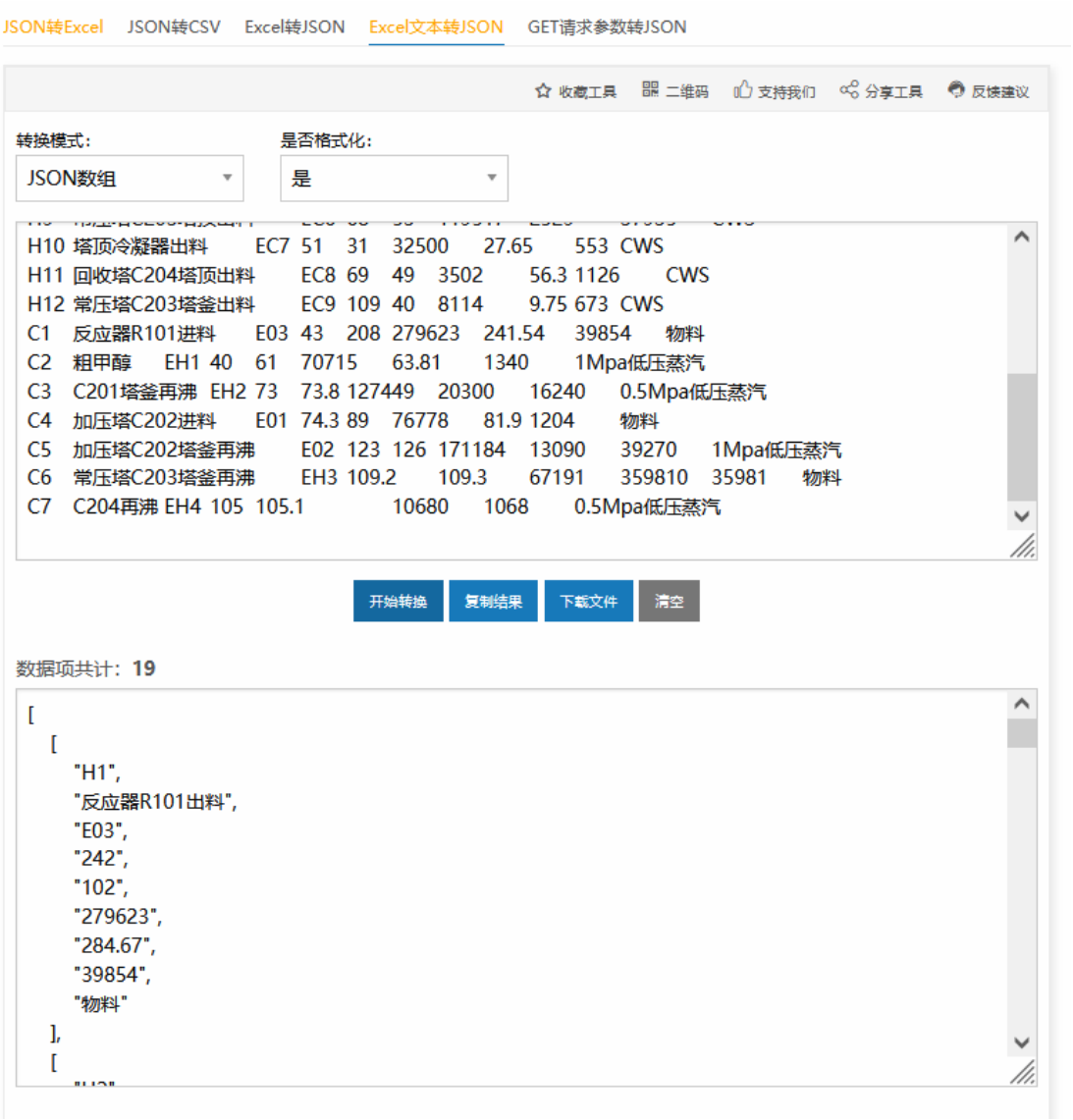


图 3-10Excel 转 Json 文件示例

3.2.4 恢复数据

点击恢复数据按钮，即可恢复数据。

图 3-11 恢复数据

3.2.5 保存文件

保存文件是将当前数据页面展示的数据保存为 Excel 文件的功能。

1. 点击保存文件按钮；
2. 输入文件名，点击保存（保存类型为 Excel 文件）。

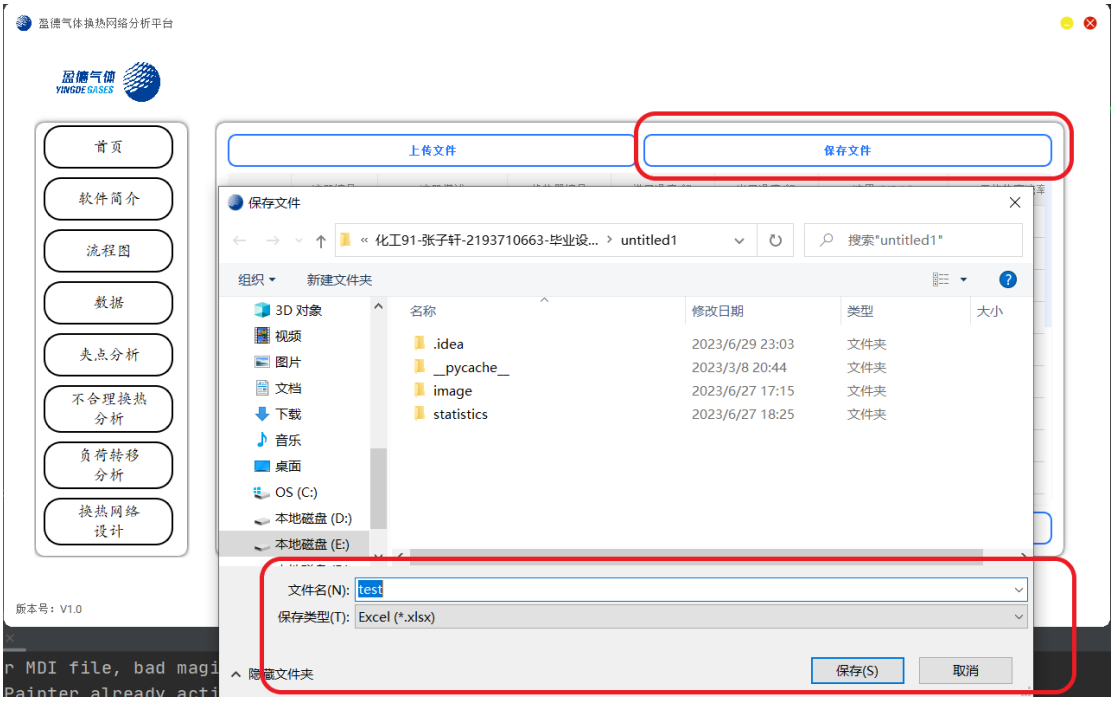


图 3-12 保存文件功能

3.3 夹点分析

夹点分析是根据问题表法计算换热系统的夹点温度、最小公用工程用量等。本模块可输出当前流股数据的冷热复合曲线、总复合曲线、冷却公用工程用量、加热公用工程用量为、节能潜力为(仅针对内置数据)、预计减少碳排放(仅针对内置数据)以及夹点温度。

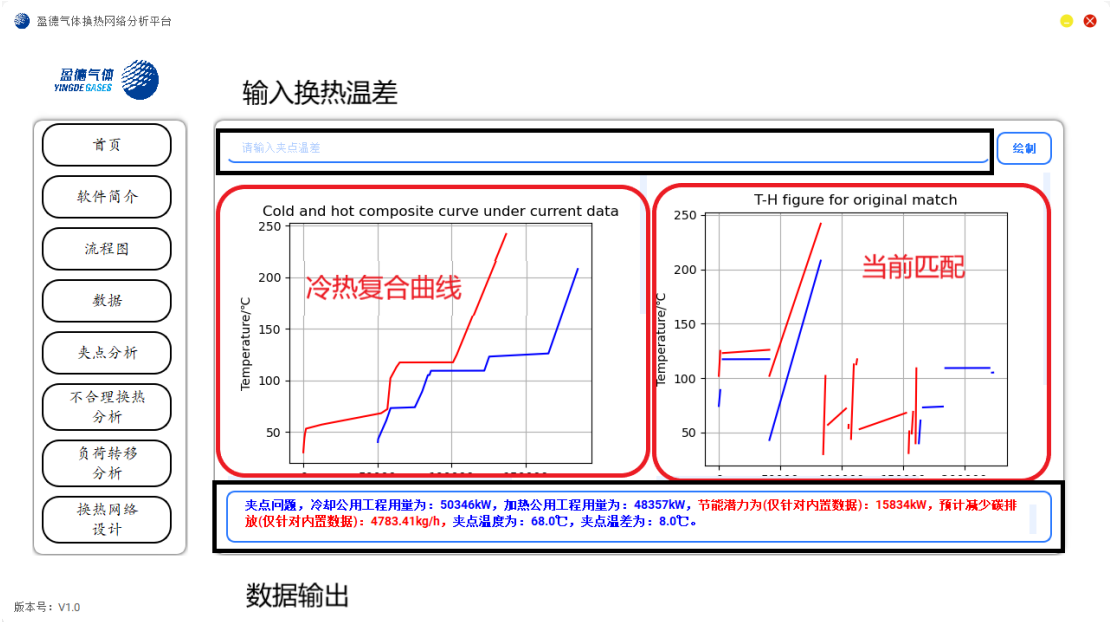


图 3-13 夹点分析模块结果输出

1. 针对内置数据，默认夹点温差为 8℃的结果已默认展示；
2. 针对内置数据，需改变换热温差的，在文本框输入夹点温差（单位℃），点击绘制按钮；
3. 针对新上传数据，上传数据后，在文本框输入夹点温差（单位℃），点击绘制按钮。

3.4 不合理换热分析

主要实现查找现行换热网络内的不合理换热器。主要不合理换热器表现形式有：**跨越夹点的冷热流股换热器、夹点之上的冷却公用工程换热器及夹点之下的加热公用工程换热器。**

单击查找不合理换热器按钮，即可对现行换热网络进行不合理换热分析。

图 3-14 不合理换热分析结果图

3.5 负荷转移分析

由于催化剂失活等导致的生产条件变化会使换热器负荷发生变动；当流股数据发生变动时，该模块可以检测负荷转移的所有可能路径，并计算可能发生的温度及负荷变化。

检索模式有：关键换热器（KHE）负荷不变、关键换热器出口温度不变。本软件选择负荷转移路径的第一个换热器作为关键换热器。

1. 数据页面更改流股数据（波动）；
2. 选择检索模式：KHE 负荷不变或 KHE 出口温度不变；
3. 点击检索，得到结果；
4. 回到数据页面，负荷迁移路径上的相应数据变化已自动更改，单元格背景色变为绿色；
5. 点击恢复数据，即可对其他流股波动进行分析。

图 3-15 负荷转移分析结果

图 3-16 数据页面响应负荷转移分析结果

3.6 换热网络设计

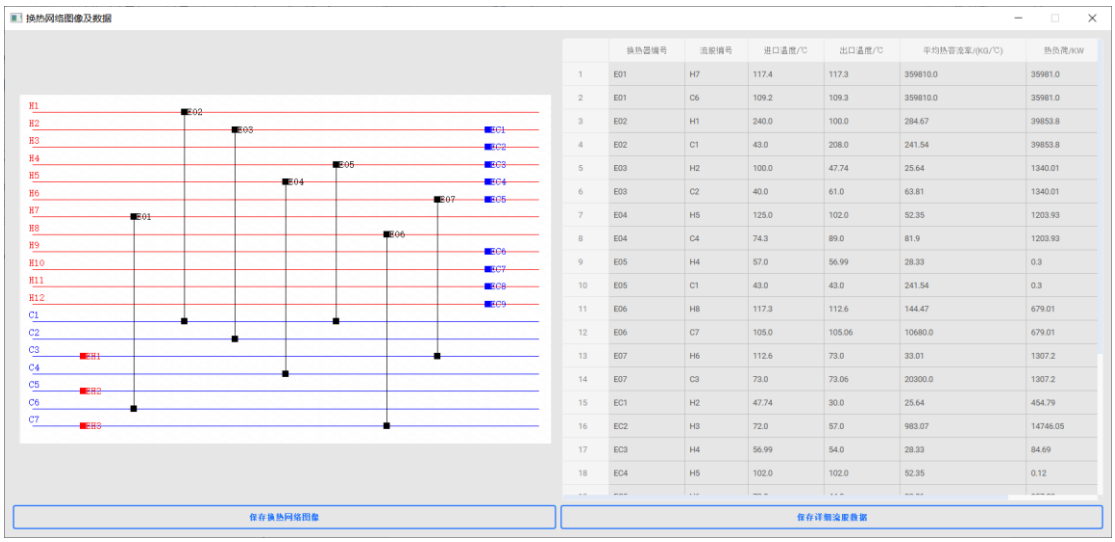
该模块可以实现以小于现行换热网络公用工程用量为目标，对流股数据进行换热网络自动设计。该模型属于通用优化模型，既可以实现使换热网络的改造设计，又可以针对新换热网络进行设计。

寻优规则：结合热力学方法随机生成可行解。终止条件：生成 10 个公用工程用量小于现行换热网络公用工程用量的解。目标解的选择：所有解中公用工程用量最小的解。

该模块将换热网络图像可视化，并配备保存图像及详细流股数据的功能。同时给出：该换热系统理论最大公用工程用量、理论最小公用工程用量、共生成可行解个数、目标解的实际公用工程用量、对于现行换热网络节省公用工程用量、预计减少碳排放以及求解耗时。

图 3-17 换热网络设计结果

1. 对于内置数据，可直接点击求解按钮进行求解；
2. 对于新上传数据，需输入现行换热网络公用工程用量（单位 kW），或输入预期最大公用工程用量，再点击求解按钮进行求解；
3. 点击查看换热网络图像及数据按钮，从子窗口查看可视化图像及详细流股数据；
4. 点击保存换热网络图像按钮，即可保存图像（.png 格式）；
5. 点击保存详细流股数据按钮，即可保存流股数据（Excel 文件）。



	换热器编号	流股编号	进口温度/℃	出口温度/℃	平均热容流率/(KG/℃)	热负荷/KW
1	E01	H7	117.4	117.3	359810.0	35981.0
2	E01	C6	109.2	109.3	359810.0	35981.0
3	E02	H1	240.0	100.0	284.67	39853.8
4	E02	C1	43.0	208.0	241.54	39853.8
5	E03	H2	100.0	47.74	25.64	1340.01
6	E03	C2	40.0	61.0	63.81	1340.01
7	E04	H5	125.0	102.0	52.35	1203.93
8	E04	C4	74.3	89.0	81.9	1203.93
9	E05	H4	57.0	56.99	28.33	0.3
10	E05	C1	43.0	43.0	241.54	0.3
11	E06	H8	117.3	112.6	144.47	679.01
12	E06	C7	105.0	105.06	10680.0	679.01
13	E07	H6	112.6	73.0	33.01	1307.2
14	E07	C3	73.0	73.06	20300.0	1307.2
15	EC1	H2	47.74	30.0	25.64	454.79
16	EC2	H3	72.0	57.0	983.07	14746.05
17	EC3	H4	56.99	54.0	28.33	84.69
18	EC4	H5	102.0	102.0	52.35	0.12

图 3-18 换热网络设计结果可视化及详细流股数据展示

4. 源码

```
1.  import os
2. from PyQt5 import QtGui, QtWidgets
3. from PyQt5.QtGui import *
4. from PyQt5.Qt import *
5. import sys
6. from PyQt5.QtWidgets import (QWidget, QHBoxLayout,  QPushButton, QLabel, QFileDialog, QVBoxLayout, QLineEdit)
7. # from PyQt5.QtGui import QPixmap
8. import numpy as np
9. import matplotlib
10.matplotlib.use("Qt5Agg")
11.from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
12.from matplotlib.figure import Figure
13.from PyQt5.QtWidgets import QApplication, QDesktopWidget
14.from PyQt5.QtCore import Qt
15.from qt_material import apply_stylesheet
16.from PyQt5.QtWidgets import *
17.import math
18.import random
19.import copy
20.import numpy as np
21.from scipy import linalg
22.import time
23.
```

```
24.BASE_DIR = os.path.dirname(sys.argv[0])
25.
26.class MyFigureCanvas(FigureCanvas):
27.    def __init__(self):
28.        fig = Figure()
29.        FigureCanvas.__init__(self, fig)
30.        self.axes = fig.add_subplot(111)
31.
32.
33.class builtPaintWidget(QWidget):
34.    def __init__(self,information,table,parent=None,):
35.        super(builtPaintWidget, self).__init__(parent)
36.        self.setMinimumSize(840, 600)
37.        self.infor = information
38.
39.        pic_show_label = QLabel(self)
40.        # 设置窗口尺寸
41.        pic_show_label.resize(840, 600)
42.        # 加载图片,并自定义图片展示尺寸
43.        self.p = QtGui.QPixmap('./image/HEN_BG.jpeg').scaled(840, 550)
44.        # 显示图片
45.        self.table_widget = table
46.        self.p = self.draw()
47.        pic_show_label.setPixmap(self.p)
48.        pic_layout = QHBoxLayout()
49.        pic_layout.addWidget(pic_show_label)
```

```
        self.setLayout(pic_layout)

    def draw(self):
        # 在当前窗口创建绘制类
        painter = QPainter(self.p)
        # 开始绘制
        painter.begin(self.p)
        ## 实线
        H = []
        C = []
        # print(self.table_widget.rowCount())
        for i in [0]:
            for j in range(self.table_widget.rowCount()):
                in_t = float(self.table_widget.item(j, 3).text())
                out_t = float(self.table_widget.item(j, 4).text())

                if in_t >= out_t and i == 0:
                    H.append(self.table_widget.item(j, i).text())
                if in_t <= out_t and i == 0:
                    C.append(self.table_widget.item(j, i).text())
        pen = QPen(Qt.red, Qt.SolidLine)
        painter.setPen(pen)
        DH = 550/(len(H)+len(C)+1)
        for i in range(len(H)):
            painter.drawLine(20, DH*(i+1), 820, DH*(i+1))
            painter.drawText(15, DH*(i+1)-5,H[i])
```

```
76.
77.         pen = QPen(Qt.blue, Qt.SolidLine)
78.         painter.setPen(pen)
79.         for j in range(len(C)):
80.             painter.drawLine(20, DH*(i+j+2), 820, DH*(i+j+2))
81.             painter.drawText(15, DH*(i+j+2) - 5, C[j])
82.
83.
84.
85.         x=[]
86.
87.         lable = [[],[],[]]
88.         for i in range(100):
89.             lable[0].append('EC%d'%(i+1))
90.             lable[1].append('E0%d'%(i+1))
91.             lable[2].append('EH%d'%(i+1))
92.         count = 0
93.         Z_kji = self.infor['structure'][1]
94.         ZHU = self.infor['structure'][0]
95.         ZCU = self.infor['structure'][2]
96.
97.         for k in range(len(H)):
98.             for j in range(len(C)):
99.                 for i in range(len(H)):
100.                     if Z_kji[k][j][i] == 1:
101.                         count = count+1
```

```
        count = count + 2
        gap = 800/(count+1)
        for t in range(count):
            x.append(int(20+gap*(t+1)))

        num = 0
        for p in range(len(C)):
            if ZHU[p] == 1:
                pen = QPen(Qt.red, 10, Qt.SolidLine)
                painter.setPen(pen)
                painter.drawPoint(x[0], DH*(p+1+len(H)))
                pen = QPen(Qt.red, 3, Qt.SolidLine)
                painter.setPen(pen)
                painter.drawText(x[0]+5, DH * (p + 1+len(H))+5,lable[2][num])
                num = num + 1

        x.pop(0)
        num = 0
        for k in range(len(H)):
            for j in range(len(C)):
                for i in range(len(H)):
                    if Z_kji[k][j][i] == 1:
                        pen = QPen(Qt.black, 10, Qt.SolidLine)
                        painter.setPen(pen)
                        painter.drawPoint(x[0], DH * (j + 1+len(H)))
                        painter.drawPoint(x[0], DH * (i + 1))
```

```
                        pen = QPen(Qt.black, 1, Qt.SolidLine)
                        painter.setPen(pen)
                        painter.drawLine(x[0], DH * (j + 1+len(H)), x[0], DH * (i + 1))
                        pen = QPen(Qt.black, 3, Qt.SolidLine)
                        painter.setPen(pen)
                        painter.drawText( x[0]+5, DH * (i + 1)+5,lable[1][num])
                        num = num + 1
                        x.pop(0)

        num = 0
        for p in range(len(H)):
            if ZCU[p] == 1:
                pen = QPen(Qt.blue, 10, Qt.SolidLine)
                painter.setPen(pen)
                painter.drawPoint(x[0], DH*(p+1))
                pen = QPen(Qt.blue, 3, Qt.SolidLine)
                painter.setPen(pen)
                painter.drawText(x[0]+5, DH * (p + 1)+5,lable[0][num])
                num = num + 1

        painter.end()
        self.p.save('./image/a.png')
        return self.p

        # ## 虚线
        # pen.setStyle(Qt.DashLine)
```

```
        # painter.setPen(pen)
        # painter.drawLine(20, 40, 200, 40)
        #
        # ## 点划线
        # pen.setStyle(Qt.DashDotLine)
        # painter.setPen(pen)
        # painter.drawLine(20, 60, 200, 60)
        #
        # ## 自定义
        # pen.setStyle(Qt.CustomDashLine)
        # pen.setDashPattern([1, 4, 5, 4])  # 设置一段线的长度和间隔
        # painter.setPen(pen)
        # painter.drawLine(20, 80, 200, 80)
        # 结束绘制


class child(QDialog):
    def __init__(self,information,table):
        super().__init__()
        self.setWindowFlag(Qt.WindowMinMaxButtonsHint)
        self.setFixedHeight(800)
        self.setFixedWidth(1750)
        self.setWindowTitle("换热网络图像及数据")
        painter_layout = QHBoxLayout()
        wigglyWidget = builtPaintWidget(information,table)

```

```
        layout = QHBoxLayout()

        painter_layout.addWidget(wigglyWidget)
        layout.addLayout(painter_layout)
        self.table_widget = table
        infor_table = QTableWidget(0,6)
        TCIN = []
        TCOUT = []
        FCPC = []
        THIN = []
        THOUT = []
        FCPH = []
        H=[]
        C=[]
        # print(self.table_widget.rowCount())
        for i in [3, 4, 6, 0]:
            for j in range(self.table_widget.rowCount()):
                in_t = float(self.table_widget.item(j, 3).text())
                out_t = float(self.table_widget.item(j, 4).text())
                if in_t >= out_t and i == 3:
                    THIN.append(float(self.table_widget.item(j, i).text()))
                    # print(TH_in)

                if in_t <= out_t and i == 3:
                    TCIN.append(float(self.table_widget.item(j, i).text()))
                if in_t >= out_t and i == 4:
```

```
                    THOUT.append(float(self.table_widget.item(j, i).text()))
                if in_t <= out_t and i == 4:
                    TCOUT.append(float(self.table_widget.item(j, i).text()))
                if in_t >= out_t and i == 6:
                    FCPH.append(float(self.table_widget.item(j, i).text()))
                if in_t <= out_t and i == 6:
                    FCPC.append(float(self.table_widget.item(j, i).text()))
                if in_t >= out_t and i == 0:
                    H.append(self.table_widget.item(j, i).text())
                if in_t <= out_t and i == 0:
                    C.append(self.table_widget.item(j, i).text())
        # self.table_widget = table_widget = QTableWidget(0, 9)
        table_header = [
            {"field": "HE_index", "text": "换热器编号", 'width': 120},
            {"field": "stream_index", "text": "流股编号", 'width': 100},
            {"field": "T_input", "text": "进口温度/℃", 'width': 120},
            {"field": "T_output", "text": "出口温度/℃", 'width': 120},
            {"field": "Cp", "text": "平均热容流率/(kg/℃)", 'width': 200},
            {"field": "heat_load", "text": "热负荷/kW", 'width': 120},
        ]
        for idx, info in enumerate(table_header):
            item = QTableWidgetItem()
            item.setText(info['text'])
            infor_table.setHorizontalHeaderItem(idx, item)
            infor_table.setColumnWidth(idx, info['width'])
        Z_kji = information['structure'][1]
```

```
        ZHU = information['structure'][0]
        ZCU = information['structure'][2]
        len_HU = 0
        len_CU = 0
        len_kji = 0


        H_dic = dict()
        C_dic = dict()
        for i in range(len(H)):
            H_dic[i] = H[i]
        for j in range(len(C)):
            C_dic[j] = C[j]
        data_list = []
        lable = [[],[],[]]
        for i in range(100):
            lable[0].append('EC%d'%(i+1))
            lable[1].append('E0%d'%(i+1))
            lable[2].append('EH%d'%(i+1))

        for k in range(len(H)):
            for j in range(len(C)):
                for i in range(len(H)):
                    if Z_kji[k][j][i] == 1:
                        temp_H = []
                        temp_C = []
```

```
                        temp_H.append(lable[1][len_kji])
                        temp_H.append(H_dic[i])
                        temp_C.append(lable[1][len_kji])
                        temp_C.append(C_dic[j])
                        temp_H.append(round(information['temputer'][0][k][i],2))
                        temp_H.append(round(information['temputer'][1][k][i],2))
                        temp_C.append(round(information['temputer'][5][k][j],2))
                        temp_C.append(round(information['temputer'][3][k][j],2))
                        temp_H.append(FCPH[i])
                        temp_C.append(FCPC[j])
                        temp_H.append(round(information['energy'][1][k][j][i],2))
                        temp_C.append(round(information['energy'][1][k][j][i],2))
                        len_kji = len_kji + 1
                        data_list.append(temp_H)
                        data_list.append(temp_C)
        for i in range(len(H)):
            if ZCU[i] == 1:
                temp = []
                temp.append(lable[0][len_CU])
                temp.append(H_dic[i])
                temp.append(round(information['temputer'][2][i],2))
                temp.append(THOUT[i])
                temp.append(FCPH[i])
                temp.append(round(information['energy'][2][i],2))
                len_CU = len_CU + 1
                data_list.append(temp)
```

```
        for j in range(len(C)):
            if ZHU[j] == 1:
                temp = []
                temp.append(lable[2][len_HU])
                temp.append(C_dic[j])
                temp.append(round(information['temputer'][4][j],2))
                temp.append(TCOUT[j])
                temp.append(FCPC[j])
                temp.append(round(information['energy'][0][j],2))
                len_HU = len_HU + 1
                data_list.append(temp)


        #
        # import json
        # file_path = os.path.join(BASE_DIR, "statistics", "origion_statistics.json")
        # with open(file_path, mode='r', encoding='utf-8') as f:
        #     data = f.read()
        # self.data_list = data_list = json.loads(data)
        current_row_count = infor_table.rowCount()
        for row_list in data_list:
            infor_table.insertRow(current_row_count)
            for i, statistic in enumerate(row_list):
                cell = QTableWidgetItem(str(statistic))
                infor_table.setItem(current_row_count, i, cell)
```

```
            current_row_count += 1
        # self.changed_index = []
        # self.change_signal = 0
        # self.color_signal = 1
        # table_widget.cellChanged.connect(self.event_cellchanged)
        infor_table.setEditTriggers(QtWidgets.QAbstractItemView.EditTrigger.NoEditTriggers)
        layout.addWidget(infor_table)
        self.table_widget = infor_table

        out_layout = QVBoxLayout()
        btn_layout = QHBoxLayout()
        btn_pic = QPushButton()
        btn_pic.setText('保存换热网络图像')
        btn_pic.clicked.connect(self.pic_save)
        btn_excel = QPushButton()
        btn_excel.setText('保存详细流股数据')
        btn_excel.clicked.connect(self.excel_save)
        btn_layout.addWidget(btn_pic)
        btn_layout.addWidget(btn_excel)

        out_layout.addLayout(layout)
        out_layout.addLayout(btn_layout)
        self.setLayout(out_layout)

    def pic_save(self):
        p = QPixmap('./image/a.png')
```

```
        file_path, file_type = QFileDialog.getSaveFileName(self, '保存文件', os.getcwd(), 'PNG (*.png)')
        if file_path and file_type:
            p.save(file_path)
    def excel_save(self):
        row_count = self.table_widget.rowCount()
        column_count = self.table_widget.columnCount()
        data = [[] for _ in range(column_count)]
        for i in range(column_count):
            for j in range(row_count):
                data[i].append(self.table_widget.item(j, i).text())
        import pandas as pd
        data_set = dict()
        table_header = ["换热器编号", "流股编号","进口温度/℃", "出口温度/℃",  "平均热容流率/(kg/℃)", "热负荷/kW"]
        for l in range(column_count):
            data_set[table_header[l]] = data[l]
        df = pd.DataFrame(data_set)
        """文件另存对话框"""
        file_path, file_type = QFileDialog.getSaveFileName(self, '保存文件', os.getcwd(), 'Excel (*.xlsx)')
        if file_path and file_type:
            writer = pd.ExcelWriter(file_path, engine='xlsxwriter')
            df.to_excel(writer)
            writer.save()

# class CallHandler(QObject):
#
#     def __init__(self):
```

```
#         super(CallHandler, self).__init__()
#
#     @pyqtSlot(str, result=str)  # 第一个参数即为回调时携带的参数类型
#     def init_home(self, str_args):
#         print('resolving......init home..')
#         print(str_args)  # 查看参数
#         # #####
#         # 这里写对应的处理逻辑比如:
#         # msg = '收到来自 python 的消息'
#         msg = self.getInfo()
#         print(msg)
#         view.page().runJavaScript("alert('%s')" % msg)
#         # MainWindow.right_frame_nonsense.nonsense_text_view.page().runJavaScript("window.say_hello('%s')" % ms
    g)
#         print(1)
#         return 'hello, Python'
#
#     def getInfo(self):
#         import socket, platform
#         hostname = socket.gethostname()
#         ip = socket.gethostbyname(hostname)
#         list_info = platform.uname()
#         sys_name = list_info[0] + list_info[2]
#         cpu_name = list_info[5]
#         dic_info = {"hostname": hostname, "ip": ip, "sys_name": sys_name, \
#                     "cpu_name": cpu_name}
```

```
#         # # 调用js函数，实现回调
#         # self.mainFrame.evaluateJavaScript('%s(%s)' % ('onGetInfo', json.dumps(dic_info)))
#         return json.dumps(dic_info)
#
#
# class WebEngine(QWebEngineView):
#     def __init__(self):
#         super(WebEngine, self).__init__()
#         self.setContextMenuPolicy(Qt.NoContextMenu)  # 设置右键菜单规则为自定义右键菜单
#         # self.customContextMenuRequested.connect(self.showRightMenu)  # 这里加载并显示自定义右键菜单，我们重点不在这里略去了详细带吗
#
#         self.setWindowTitle('QWebChannel 与前端交互')
#         # self.resize(1100, 650)
#         cp = QDesktopWidget().availableGeometry().center()


#然后创建一个主窗口类:


class MainWindow(QtWidgets.QWidget):
    def __init__(self):
        super().__init__()
```

```
412.         self.width = 1280
413.         self.height = 720
414.
415.     # 设置软件图标（可省略）
416.         self.setWindowIcon(QtGui.QIcon("image/logo"))
417.     # 设置主界面标题
418.         self.setWindowTitle("盈德气体甲醇合成换热网络扰动分析平台")
419.     # 设置固定尺寸
420.         self.setFixedSize(self.width, self.height)
421.     # 设置主界面背景色
422.         self.setStyleSheet('''MainWindow{background-color:white}''')
423.
424.     #无边框
425.         self.setWindowFlags(Qt.FramelessWindowHint)
426.
427.     #设置圆角
428.         bitmap = QBitmap(self.size())
429.         bitmap.fill()
430.         painter = QPainter(bitmap)
431.         painter.begin(self)
432.         painter.setPen(Qt.NoPen)
433.         painter.setBrush(Qt.black)
434.         painter.setRenderHint(QPainter.Antialiasing)
435.         painter.drawRoundedRect(bitmap.rect(), 10, 10)
436.         painter.end()
437.         self.setMask(bitmap)
```

```
438.
439.     #居中显示
440.         # qr = self.frameGeometry()
441.         # cp = QDesktopWidget().availableGeometry().center()
442.         # qr.moveCenter(cp)
443.     #上下两个布局
444.         self.title_layout = QVBoxLayout()
445.
446.         self.upper_layout = QHBoxLayout()
447.
448.         logo_small = QLabel()
449.         logo_small.setPixmap(QPixmap("image/logo.tif").scaled(20, 20))
450.         self.upper_layout.addWidget(logo_small)
451.
452.         self.title = QLabel("盈德气体换热网络分析平台")
453.         self.title.setFixedWidth(1300)
454.         self.upper_layout.addWidget(self.title)
455.
456.         self.upper_layout.addStretch()
457.
458.         btn_min = QPushButton()
459.         btn_min.setIcon(QIcon(QPixmap("image/最小化.svg")))
460.         btn_min.setStyleSheet('''
461.                               QPushButton
462.                               {text-align : center;
463.                               color : black;
```

```
                                background-color : white;
                                font: bold;
                                border-color: white;
                                border-width: 0px;
                                border-radius:10px;
                                padding: 6px;
                                height : 14px;
                                border-style: outset;
                                font : 14px;}
                                QPushButton:pressed
                                {text-align : center;
                                background-color : black;
                                font: bold;
                                border-color: white;
                                border-width: 0px;
                                border-radius: 10px;
                                padding: 6px;
                                height : 14px;
                                border-style: outset;
                                font : 14px;}
                                ''')
        btn_min.setFixedHeight(20)
        btn_min.setFixedWidth(20)
        btn_min.clicked.connect(self.showMinimized)
        self.upper_layout.addWidget(btn_min)
```

```
        btn_exit = QPushButton()
        btn_exit.setIcon(QIcon(QPixmap("image/关闭.svg")))
        btn_exit.setStyleSheet('''
                                QPushButton
                                {text-align : center;
                                color : black;
                                background-color : white;
                                font: bold;
                                border-color: white;
                                border-width: 0px;
                                border-radius: 10px;
                                padding: 6px;
                                height : 14px;
                                border-style: outset;
                                font : 14px;}
                                QPushButton:pressed
                                {text-align : center;
                                background-color : black;
                                font: bold;
                                border-color: white;
                                border-width: 0px;
                                border-radius: 10px;
                                padding: 6px;
                                height : 14px;
                                border-style: outset;
                                font : 14px;}
```

```
                                ''')
        btn_exit.setFixedHeight(20)
        btn_exit.setFixedWidth(20)
        btn_exit.clicked.connect(self.close)
        self.upper_layout.addWidget(btn_exit)


        self.title_layout.addLayout(self.upper_layout)
    #左右两个布局

        self.layout = QHBoxLayout()
        self.layout.addWidget(self.init_left())
        self.layout.addWidget(self.init_right())

        self.title_layout.addLayout(self.layout)

        self.title_layout.addStretch()

        version = QLabel("版本号：V1.0")
        self.title_layout.addWidget(version)

        self.setLayout(self.title_layout)


    def mousePressEvent(self, event):
        if event.button() == Qt.LeftButton:
```

```
            # 标记是否按下
            self.m_flag = True
            # 获取鼠标相对窗口的位置
            self.m_Position = event.globalPos() - self.pos()
            event.accept()

    def mouseMoveEvent(self, QMouseEvent):
        try:
            # 仅监听标题栏
            if Qt.LeftButton and self.m_flag and self.title.underMouse():
                # 更改鼠标图标
                self.setCursor(QCursor(Qt.OpenHandCursor))
                # 更改窗口位置
                self.move(QMouseEvent.globalPos() - self.m_Position)
                QMouseEvent.accept()
        except Exception as e:
            print("报错信息=", e)

    def mouseReleaseEvent(self, QMouseEvent):
        self.m_flag = False
        # 恢复鼠标形状
        self.setCursor(QCursor(Qt.ArrowCursor))


    def init_left(self):

```

```
568.
569.         left_widget = QWidget(parent = self)
570.         left_widget.setFixedSize(180, 600)
571.         left_widget.setStyleSheet('''QWidget{background-color: white;
572.                                             border-radius: 10px;
573.                                             }''')
574.
575.
576.         left_frame = QFrame(left_widget)
577.         left_frame.setStyleSheet('''QFrame{background-color: white;
578.                                             border-radius: 10px;
579.                                             }''')
580.         left_frame.setFrameShadow(QFrame.Raised)
581.         left_frame.setFrameShape(QFrame.Box)
582.         left_frame.setLineWidth(3)
583.         left_frame.setMidLineWidth(3)
584.         left_frame.resize(175, 500)
585.         left_frame.move(2,98)
586.         left_frame.shadow = QtWidgets.QGraphicsDropShadowEffect()
587.         left_frame.shadow.setOffset(0,0)  # 偏移
588.         left_frame.shadow.setBlurRadius(10)  # 阴影半径
589.         left_frame.shadow.setColor(QColor('black'))  # 阴影颜色
590.         left_frame.setGraphicsEffect(left_frame.shadow)  # 将设置套用到widget 窗口中
591.
592.         left_layout = QVBoxLayout()
593.         left_widget.setLayout(left_layout)
```

```
594.
595.
596.
597.
598.         lable_logo = QLabel()
599.         lable_logo.setPixmap(QPixmap("image/logofull.png").scaled(150,80))
600.         lable_logo.setStyleSheet("border: 1px white")
601.         left_layout.addWidget(lable_logo)
602.         left_layout.addStretch()
603.
604.         btn_homepage = QPushButton("首页")
605.         btn_homepage.setFixedWidth(150)
606.         btn_homepage.setFixedHeight(50)
607.         btn_homepage.setStyleSheet('''
608.                                 QPushButton
609.                                 {text-align : center;
610.                                 color : black;
611.                                 background-color : white;
612.                                 font: 800;
613.                                 border-color: black;
614.                                 border-width: 1.5px;
615.                                 border-radius: 20px;
616.                                 padding: 6px;
617.                                 height : 14px;
618.                                 border-style: outset;
619.                                 font : 18px;
```

```
                                font-family : STKaiti}
                                QPushButton:pressed
                                {text-align : center;
                                background-color : #F5F5F5;
                                font: bold;
                                border-color: black;
                                border-width: 2px;
                                border-radius: 20px;
                                padding: 6px;
                                height : 14px;
                                border-style: outset;
                                font : 18px;
                                font-family : STKaiti}
                                ''')
        btn_homepage.clicked.connect(self.switch_homepage)
        left_layout.addWidget(btn_homepage)
        left_layout.addStretch()

        btn_introduction = QPushButton("软件简介")
        btn_introduction.setFixedWidth(150)
        btn_introduction.setFixedHeight(50)
        btn_introduction.setStyleSheet('''
                                    QPushButton
                                    {text-align : center;
                                    color : black;
                                    background-color : white;
```

```
                                    font: 800;
                                    border-color: black;
                                    border-width: 1.5px;
                                    border-radius: 20px;
                                    padding: 6px;
                                    height : 14px;
                                    border-style: outset;
                                    font : 18px;
                                    font-family : STKaiti}
                                    QPushButton:pressed
                                    {text-align : center;
                                    background-color : #F5F5F5;
                                    font: bold;
                                    border-color: black;
                                    border-width: 2px;
                                    border-radius: 20px;
                                    padding: 6px;
                                    height : 14px;
                                    border-style: outset;
                                    font : 18px;
                                    font-family : STKaiti}
                                    ''')
        btn_introduction.clicked.connect(self.switch_introduction)
        left_layout.addWidget(btn_introduction)
        left_layout.addStretch()

```

```
672.         btn_pics = QPushButton("流程图")
673.         btn_pics.setFixedWidth(150)
674.         btn_pics.setFixedHeight(50)
675.         btn_pics.setStyleSheet('''QPushButton
676.                                     {text-align : center;
677.                                     color : black;
678.                                     background-color : white;
679.                                     font: bold;
680.                                     border-color: black;
681.                                     border-width: 1.5px;
682.                                     border-radius: 20px;
683.                                     padding: 6px;
684.                                     height : 14px;
685.                                     border-style: outset;
686.                                     font : 18px;
687.                                     font-family : STKaiti}
688.                                     QPushButton:pressed
689.                                     {text-align : center;
690.                                     background-color : #F5F5F5;
691.                                     font: bold;
692.                                     border-color: black;
693.                                     border-width: 1.5px;
694.                                     border-radius: 20px;
695.                                     padding: 6px;
696.                                     height : 14px;
697.                                     border-style: outset;
```

```
                                    font : 18px;
                                    font-family : STKaiti}
                                    ''')
        btn_pics.clicked.connect(self.switch_pics)
        left_layout.addWidget(btn_pics)
        left_layout.addStretch()

        btn_statistics = QPushButton("数据")
        btn_statistics.setFixedWidth(150)
        btn_statistics.setFixedHeight(50)
        btn_statistics.setStyleSheet('''QPushButton
                                    {text-align : center;
                                    color : black;
                                    background-color : white;
                                    font: bold;
                                    border-color: black;
                                    border-width: 1.5px;
                                    border-radius: 20px;
                                    padding: 6px;
                                    height : 14px;
                                    border-style: outset;
                                    font : 18px;
                                    font-family : STKaiti}
                                    QPushButton:pressed
                                    {text-align : center;
                                    background-color : #F5F5F5;
```

```
                                    font: bold;
                                    border-color: black;
                                    border-width: 2px;
                                    border-radius: 20px;
                                    padding: 6px;
                                    height : 14px;
                                    border-style: outset;
                                    font : 18px;
                                    font-family : STKaiti}
                                    ''')
        btn_statistics.clicked.connect(self.switch_statistic)
        left_layout.addWidget(btn_statistics)
        left_layout.addStretch()

        btn_pinch = QPushButton("夹点分析")
        btn_pinch.setFixedWidth(150)
        btn_pinch.setFixedHeight(50)
        btn_pinch.setStyleSheet('''QPushButton
                                    {text-align : center;
                                    color : black;
                                    background-color : white;
                                    font: bold;
                                    border-color: black;
                                    border-width: 1.5px;
                                    border-radius: 20px;
                                    padding: 6px;
```

```
                                    height : 14px;
                                    border-style: outset;
                                    font : 18px;
                                    font-family : STKaiti}
                                    QPushButton:pressed
                                    {text-align : center;
                                    background-color : #F5F5F5;
                                    font: bold;
                                    border-color: black;
                                    border-width: 2px;
                                    border-radius: 20px;
                                    padding: 6px;
                                    height : 14px;
                                    border-style: outset;
                                    font : 18px;
                                    font-family : STKaiti}
                                    ''')
        btn_pinch.clicked.connect(self.switch_pinch)
        left_layout.addWidget(btn_pinch)
        left_layout.addStretch()

        btn_nonsense = QPushButton("不合理换热\n 分析")
        btn_nonsense.setFixedWidth(150)
        btn_nonsense.setFixedHeight(55)
        btn_nonsense.setStyleSheet('''QPushButton
                                    {text-align : center;
```

```
                                        color : black;
                                        background-color : white;
                                        font: bold;
                                        border-color: black;
                                        border-width: 1.5px;
                                        border-radius: 20px;
                                        padding: 6px;
                                        height : 14px;
                                        border-style: outset;
                                        font : 18px;
                                        font-family : STKaiti}
                                        QPushButton:pressed
                                        {text-align : center;
                                        background-color : #F5F5F5;
                                        font: bold;
                                        border-color: black;
                                        border-width: 2px;
                                        border-radius: 20px;
                                        padding: 6px;
                                        height : 14px;
                                        border-style: outset;
                                        font : 18px;
                                        font-family : STKaiti}
                                        ''')
        btn_nonsense.clicked.connect(self.switch_nonsense)
        left_layout.addWidget(btn_nonsense)
```

```
        left_layout.addStretch()

        btn_loadshift = QPushButton("负荷转移\n 分析")
        btn_loadshift.setFixedWidth(150)
        btn_loadshift.setFixedHeight(55)
        btn_loadshift.setStyleSheet('''QPushButton
                                       {text-align : center;
                                       color : black;
                                       background-color : white;
                                       font: bold;
                                       border-color: black;
                                       border-width: 1.5px;
                                       border-radius: 20px;
                                       padding: 6px;
                                       height : 14px;
                                       border-style: outset;
                                       font : 18px;
                                       font-family : STKaiti}
                                       QPushButton:pressed
                                       {text-align : center;
                                       background-color : #F5F5F5;
                                       font: bold;
                                       border-color: black;
                                       border-width: 2px;
                                       border-radius: 20px;
                                       padding: 6px;
```

```
                                    height : 14px;
                                    border-style: outset;
                                    font : 18px;
                                    font-family : STKaiti}
                                    ''')
        btn_loadshift.clicked.connect(self.switch_load)
        left_layout.addWidget(btn_loadshift)
        left_layout.addStretch()

        btn_HENdesign = QPushButton("换热网络\n 设计")
        btn_HENdesign.setFixedWidth(150)
        btn_HENdesign.setFixedHeight(55)
        btn_HENdesign.setStyleSheet('''QPushButton
                                    {text-align : center;
                                    color : black;
                                    background-color : white;
                                    font: bold;
                                    border-color: black;
                                    border-width: 1.5px;
                                    border-radius: 20px;
                                    padding: 6px;
                                    height : 14px;
                                    border-style: outset;
                                    font : 18px;
                                    font-family : STKaiti}
                                    QPushButton:pressed
```

```
                                    {text-align : center;
                                    background-color : #F5F5F5;
                                    font: bold;
                                    border-color: black;
                                    border-width: 2px;
                                    border-radius: 20px;
                                    padding: 6px;
                                    height : 14px;
                                    border-style: outset;
                                    font : 18px;
                                    font-family : STKaiti}
                                    ''')
        btn_HENdesign.clicked.connect(self.switch_design)
        left_layout.addWidget(btn_HENdesign)
        left_layout.addStretch()

        return left_widget


    def init_right(self):
        # right_layout = QVBoxLayout()
        # btn_mainpage = QPushButton("首页")
        # right_layout.addWidget(btn_mainpage)
        # btn_statistics = QPushButton("数据")
        # right_layout.addWidget(btn_statistics)
        # right_layout.addStretch()
```

```
880.         self.right_widget = QWidget(parent=self)
881.         self.right_widget.setFixedSize(1000, 600)
882.         self.right_widget.setStyleSheet('''QWidget{background-color: white;
883.                                                    border-radius: 10px;
884.                                                    }''')
885.         #图片 frame
886.         self.right_frame_pics = QFrame(self.right_widget)
887.         self.right_frame_pics.setStyleSheet('''QFrame{background-color: white;
888.                                                    border-radius: 10px;
889.                                                    }''')
890.         self.right_frame_pics.setFrameShadow(QFrame.Raised)
891.         self.right_frame_pics.setFrameShape(QFrame.Box)
892.         self.right_frame_pics.setLineWidth(3)
893.         self.right_frame_pics.setMidLineWidth(3)
894.         self.right_frame_pics.resize(980, 500)
895.         self.right_frame_pics.move(2, 98)
896.         self.right_frame_pics.shadow = QtWidgets.QGraphicsDropShadowEffect()
897.         self.right_frame_pics.shadow.setOffset(0, 0)  # 偏移
898.         self.right_frame_pics.shadow.setBlurRadius(10)  # 阴影半径
899.         self.right_frame_pics.shadow.setColor(QColor('black'))  # 阴影颜色
900.         self.right_frame_pics.setGraphicsEffect(self.right_frame_pics.shadow)  # 将设置套用到widget 窗口中
901.
902.         pics = QHBoxLayout()
903.         self.right_frame_pics.setLayout(pics)
904.         self.image_index = 1
905.
```

```
906.         btn_last = QPushButton()
907.         btn_last.setIcon(QIcon(QPixmap('image/last.png')))
908.         btn_last.setStyleSheet('''
909.                                QPushButton
910.                                {text-align : center;
911.                                color : black;
912.                                background-color : white;
913.                                font: bold;
914.                                border-color: white;
915.                                border-width: 0px;
916.                                border-radius: 10px;
917.                                padding: 6px;
918.                                height : 14px;
919.                                border-style: outset;
920.                                font : 14px;}
921.                                QPushButton:pressed
922.                                {text-align : center;
923.                                background-color : black;
924.                                font: bold;
925.                                border-color: white;
926.                                border-width: 0px;
927.                                border-radius: 10px;
928.                                padding: 6px;
929.                                height : 14px;
930.                                border-style: outset;
931.                                font : 14px;}
```

```
                            ''')
        btn_last.setFixedHeight(20)
        btn_last.setFixedWidth(20)
        btn_last.clicked.connect(self.switch_last)
        pics.addWidget(btn_last)
        pics.addStretch()


        self.img1 = QLabel()
        # img = QImageReader()
        # scale = 800 / img.size().width()
        # height = int(img.size().height() * scale)
        # img.setScaledSize(QSize(800, height))
        # fname, _ = QFileDialog.getOpenFileName(self, 'Open File','C://', "Image files (*.jpg *.png)")
        # img = img.read("jingliu.tif")
        # # 打开设置好的图片
        # pixmap = QPixmap(img)
        self.img1.setPixmap(QPixmap('image/hecheng.tif'))
        self.img1.setScaledContents(True)
        self.img1.setFixedWidth(800)
        pics.addWidget(self.img1)

        self.img2 = QLabel()
        self.img2.setPixmap(QPixmap('image/jingliu.tif'))
        self.img2.setScaledContents(True)
        pics.addWidget(self.img2)
```

```
958.         self.img2.hide()
959.
960.         self.img3 = QLabel()
961.         self.img3.setPixmap(QPixmap('image/wangluo.tif'))
962.         self.img3.setScaledContents(True)
963.         self.img3.setFixedWidth(600)
964.         pics.addWidget(self.img3)
965.         self.img3.hide()
966.
967.         self.img4 = QLabel()
968.         self.img4.setPixmap(QPixmap('image/youxiang.tif'))
969.         self.img4.setScaledContents(True)
970.         pics.addWidget(self.img4)
971.         self.img4.hide()
972.
973.         btn_next = QPushButton()
974.         btn_next.setIcon(QIcon(QPixmap('image/next.png')))
975.         btn_next.setStyleSheet('''
976.                                     QPushButton
977.                                     {text-align : center;
978.                                     color : black;
979.                                     background-color : white;
980.                                     font: bold;
981.                                     border-color: white;
982.                                     border-width: 0px;
983.                                     border-radius: 10px;
```

```
                                    padding: 6px;
                                    height : 14px;
                                    border-style: outset;
                                    font : 14px;}
                                    QPushButton:pressed
                                    {text-align : center;
                                    background-color : black;
                                    font: bold;
                                    border-color: white;
                                    border-width: 0px;
                                    border-radius: 10px;
                                    padding: 6px;
                                    height : 14px;
                                    border-style: outset;
                                    font : 14px;}
                                    ''')
            btn_next.setFixedHeight(20)
            btn_next.setFixedWidth(20)
            btn_next.clicked.connect(self.switch_next)
            pics.addStretch()
            pics.addWidget(btn_next)
            self.right_frame_pics.hide()
            #数据页面
            self.forbidden_color=1
            self.right_frame_statistic = QFrame(self.right_widget)
            self.right_frame_statistic.setStyleSheet('''QFrame{background-color: white;
```

```
                                                        border-radius: 10px;
                                                        }''')
        self.right_frame_statistic.setFrameShadow(QFrame.Raised)
        self.right_frame_statistic.setFrameShape(QFrame.Box)
        self.right_frame_statistic.setLineWidth(3)
        self.right_frame_statistic.setMidLineWidth(3)
        self.right_frame_statistic.resize(980, 500)
        self.right_frame_statistic.move(2, 98)
        self.right_frame_statistic.shadow = QtWidgets.QGraphicsDropShadowEffect()
        self.right_frame_statistic.shadow.setOffset(0, 0)  # 偏移
        self.right_frame_statistic.shadow.setBlurRadius(10)  # 阴影半径
        self.right_frame_statistic.shadow.setColor(QColor('black'))  # 阴影颜色
        self.right_frame_statistic.setGraphicsEffect(self.right_frame_statistic.shadow)  # 将设置套用到widget
    窗口中
        statistic_layout = QVBoxLayout()

        statistic_add_layout = QHBoxLayout()
        self.upload_btn = QPushButton("上传文件", self)
        self.upload_btn.clicked.connect(self.select_file)  # 为button 绑定消息对话框
        self.download_btn = QPushButton("保存文件", self)
        self.download_btn.clicked.connect(self.save_file)  # 为button 绑定消息对话框
        # txt_pinch = QLineEdit()
        # txt_pinch.text()
        # statistic_add_layout.addWidget(txt_pinch)
        # self.txt_add = txt_pinch
```

```
1034.           # txt_pinch.setPlaceholderText("请按照“流股编号，流股描述，换热器编号，进口温度，出口温度，流量，平均热容流率，
    热负荷，换热介质”的顺序及格式输入")
1035.           # btn_add = QPushButton("ADD")
1036.           # btn_add.clicked.connect(self.event_add_click)
1037.           statistic_add_layout.addWidget(self.upload_btn)
1038.           statistic_add_layout.addWidget(self.download_btn)
1039.           statistic_layout.addLayout(statistic_add_layout)
1040.
1041.           self.table_widget = table_widget = QTableWidget(0, 9)
1042.           table_header = [
1043.               {"field": "stream_index", "text": "流股编号", 'width': 100},
1044.               {"field": "stream_discribe", "text": "流股描述", 'width': 150},
1045.               {"field": "HE_index", "text": "换热器编号", 'width': 120},
1046.               {"field": "T_input", "text": "进口温度/℃", 'width': 120},
1047.               {"field": "T_output", "text": "出口温度/℃", 'width': 120},
1048.               {"field": "flow_rate", "text": "流量/(kg/h)", 'width': 150},
1049.               {"field": "Cp", "text": "平均热容流率/(kg/℃)", 'width': 200},
1050.               {"field": "heat_load", "text": "热负荷/kW", 'width': 120},
1051.               {"field": "medium", "text": "换热介质", 'width': 100},
1052.           ]
1053.           for idx, info in enumerate(table_header):
1054.               item = QTableWidgetItem()
1055.               item.setText(info['text'])
1056.               table_widget.setHorizontalHeaderItem(idx, item)
1057.               table_widget.setColumnWidth(idx, info['width'])
1058.
```

```
1059.           import json
1060.           file_path = os.path.join(BASE_DIR, "statistics", "origion_statistics.json")
1061.           with open(file_path, mode='r', encoding='utf-8') as f:
1062.               data = f.read()
1063.           self.data_list = data_list = json.loads(data)
1064.           current_row_count = table_widget.rowCount()
1065.           for row_list in data_list:
1066.               table_widget.insertRow(current_row_count)
1067.               for i, statistic in enumerate(row_list):
1068.                   cell = QTableWidgetItem(str(statistic))
1069.                   table_widget.setItem(current_row_count, i, cell)
1070.               current_row_count += 1
1071.           self.changed_index = []
1072.           self.change_signal = 0
1073.           self.color_signal = 1
1074.           table_widget.cellChanged.connect(self.event_cellchanged)
1075.           # self.table_widget.setStyleSheet('''QTableView::Item{background-color: pink}''')
1076.
1077.
1078.
1079.
1080.
1081.           statistic_layout.addWidget(table_widget)
1082.
1083.           statistic_btn_layout = QHBoxLayout()
1084.
```

```
1085.           btn_delete = QPushButton()
1086.           btn_delete.setText('清空数据')
1087.           btn_delete.clicked.connect(self.event_statistic_delete)
1088.           statistic_btn_layout.addWidget(btn_delete)
1089.
1090.           btn_recover = QPushButton()
1091.           btn_recover.setText('恢复数据')
1092.           btn_recover.clicked.connect(self.event_statistic_recover)
1093.           statistic_btn_layout.addWidget(btn_recover)
1094.
1095.           # btn_newline = QPushButton()
1096.           # btn_newline.setText('添加数据')
1097.           # statistic_btn_layout.addWidget(btn_newline)
1098.           # btn_newline.clicked.connect(self.event_btn_newline)
1099.
1100.           statistic_layout.addLayout(statistic_btn_layout)
1101.
1102.           self.right_frame_statistic.setLayout(statistic_layout)
1103.
1104.
1105.
1106.
1107.
1108.
1109.           self.right_frame_statistic.hide()
1110.           #夹点
```

```
1111.           self.right_frame_pinch = QFrame(self.right_widget)
1112.           self.right_frame_pinch.setStyleSheet('''QFrame{background-color: white;
1113.                                                   border-radius: 10px;
1114.                                                   }''')
1115.           self.right_frame_pinch.setFrameShadow(QFrame.Raised)
1116.           self.right_frame_pinch.setFrameShape(QFrame.Box)
1117.           self.right_frame_pinch.setLineWidth(3)
1118.           self.right_frame_pinch.setMidLineWidth(3)
1119.           self.right_frame_pinch.resize(980, 500)
1120.           self.right_frame_pinch.move(2, 98)
1121.           self.right_frame_pinch.shadow = QtWidgets.QGraphicsDropShadowEffect()
1122.           self.right_frame_pinch.shadow.setOffset(0, 0)  # 偏移
1123.           self.right_frame_pinch.shadow.setBlurRadius(10)  # 阴影半径
1124.           self.right_frame_pinch.shadow.setColor(QColor('black'))  # 阴影颜色
1125.           self.right_frame_pinch.setGraphicsEffect(self.right_frame_pinch.shadow)  # 将设置套用到widget窗口中
1126.
1127.           pinch_layout = QVBoxLayout()
1128.
1129.           pinch_add_layout = QHBoxLayout()
1130.
1131.           self.pinch_print = QTextEdit()
1132.           self.pinch_print.setFixedHeight(60)
1133.
1134.           txt_pinch = QLineEdit()
1135.           txt_pinch.text()
1136.           pinch_add_layout.addWidget(txt_pinch)
```

```
1137.           self.txt_pinch = txt_pinch
1138.           txt_pinch.setPlaceholderText("请输入夹点温差")
1139.           btn_pinch = QPushButton("绘制")
1140.           self.DTmin = 8
1141.           btn_pinch.clicked.connect(self.event_pinch_click)
1142.           pinch_add_layout.addWidget(btn_pinch)
1143.           pinch_layout.addLayout(pinch_add_layout)
1144.
1145.           graphic_layout = QHBoxLayout()
1146.
1147.           self.graphicView = QGraphicsView()
1148.           # self.graphicView.verticalScrollBar().setSliderPosition(0)
1149.           # self.graphicView.horizontalScrollBar().setSliderPosition(0)
1150.           self.fig = MyFigureCanvas()
1151.           self.fig3 = MyFigureCanvas()
1152.           self.graphicScene = QGraphicsScene()
1153.           # self.graphicScene.addWidget(self.fig)
1154.           # self.graphicView.setScene(self.graphicScene)
1155.
1156.           curve_frame = QFrame()
1157.           curve_frame.setStyleSheet('''QFrame{background-color: white;}''')
1158.           curve_layout = QVBoxLayout()
1159.           curve_layout.addWidget(self.fig)
1160.           curve_layout.addWidget(self.fig3)
1161.
1162.           curve_frame.setLayout(curve_layout)
```

```
1163.             self.graphicScene.addWidget(curve_frame)
1164.             self.graphicView.setScene(self.graphicScene)
1165.             graphic_layout.addWidget(self.graphicView)
1166.             self.slot_plot()
1167.             self.slot_plot3()
1168.             self.graphicView.centerOn(0,0)
1169.             graphic_layout.addStretch()
1170.             self.graphicView2 = QGraphicsView()
1171.             self.fig2 = MyFigureCanvas()
1172.             self.graphicScene2 = QGraphicsScene()
1173.             self.graphicScene2.addWidget(self.fig2)
1174.             self.graphicView2.setScene(self.graphicScene2)
1175.             self.slot_plot2()
1176.             self.graphicView2.centerOn(0,0)
1177.             graphic_layout.addWidget(self.graphicView2)
1178.             pinch_layout.addLayout(graphic_layout)
1179.
1180.             self.pinch_print.setStyleSheet('''QTextEdit{font: bold 14px;
1181.                                                         }''')
1182.             pinch_layout.addWidget(self.pinch_print)
1183.
1184.             self.right_frame_pinch.setLayout(pinch_layout)
1185.             self.right_frame_pinch.hide()
1186.             #负荷
1187.             self.right_frame_load = QFrame(self.right_widget)
1188.             self.right_frame_load.setStyleSheet('''QFrame{background-color: white;
```

```
                                                        border-radius: 10px;
                                                        }''')
        self.right_frame_load.setFrameShadow(QFrame.Raised)
        self.right_frame_load.setFrameShape(QFrame.Box)
        self.right_frame_load.setLineWidth(3)
        self.right_frame_load.setMidLineWidth(3)
        self.right_frame_load.resize(980, 500)
        self.right_frame_load.move(2, 98)
        self.right_frame_load.shadow = QtWidgets.QGraphicsDropShadowEffect()
        self.right_frame_load.shadow.setOffset(0, 0)  # 偏移
        self.right_frame_load.shadow.setBlurRadius(10)  # 阴影半径
        self.right_frame_load.shadow.setColor(QColor('black'))  # 阴影颜色
        self.right_frame_load.setGraphicsEffect(self.right_frame_load.shadow)  # 将设置套用到widget窗口中

        load_layout = QVBoxLayout()
        load_btn_layout = QHBoxLayout()
        load_shift_chose = QComboBox()
        # load_shift_chose_index = [0, 1]
        # load_shift_chose_text = ['出口温度不变', 'KHE热负荷不变']
        # for i in range(len(load_shift_chose_index) - 1):
        #     load_shift_chose.setItemText(load_shift_chose_index[i], load_shift_chose_text[i])
        load_shift_chose.addItems(['KHE出口温度不变', 'KHE热负荷不变'])
        self.load_shift_chose = load_shift_chose
        load_btn_layout.addWidget(load_shift_chose)
        btn_load_shift = QPushButton()
        btn_load_shift.setText('检索')
```

```
        btn_load_shift.clicked.connect(self.event_load_shift_search)
        load_btn_layout.addWidget(btn_load_shift)
        load_layout.addLayout(load_btn_layout)
        load_print = QTextEdit()
        self.load_print = load_print
        load_layout.addWidget(load_print)
        load_btn_bottom_layout = QHBoxLayout()
        btn_remove_load_data = QPushButton('清空')
        # load_btn_bottom_layout.addWidget(btn_remove_load_data)
        btn_refresh_load_data = QPushButton('更新负荷数据')
        # load_btn_bottom_layout.addWidget(btn_refresh_load_data)
        load_layout.addLayout(load_btn_bottom_layout)
        self.right_frame_load.setLayout(load_layout)
        self.right_frame_load.hide()
        #设计
        self.right_frame_design = QFrame(self.right_widget)
        self.right_frame_design.setStyleSheet('''QFrame{background-color: white;
                                                         border-radius: 10px;
                                                         }''')
        self.right_frame_design.setFrameShadow(QFrame.Raised)
        self.right_frame_design.setFrameShape(QFrame.Box)
        self.right_frame_design.setLineWidth(3)
        self.right_frame_design.setMidLineWidth(3)
        self.right_frame_design.resize(980, 500)
        self.right_frame_design.move(2, 98)
        self.right_frame_design.shadow = QtWidgets.QGraphicsDropShadowEffect()
```

```
1241.           self.right_frame_design.shadow.setOffset(0, 0)  # 偏移
1242.           self.right_frame_design.shadow.setBlurRadius(10)  # 阴影半径
1243.           self.right_frame_design.shadow.setColor(QColor('black'))  # 阴影颜色
1244.           self.right_frame_design.setGraphicsEffect(self.right_frame_design.shadow)  # 将设置套用到widget窗口中
1245.
1246.
1247.           design_layout = QVBoxLayout()
1248.
1249.           notice_layout = QVBoxLayout()
1250.           # notice_layout.addStretch()
1251.           note = QLabel()
1252.           note.setText('说明')
1253.           note.setStyleSheet("font: 25px")
1254.           notice_layout.addWidget(note)
1255.           parameters = QLabel()
1256.           parameters.setText("①设计目标：小于现行公用工程用量（默认 114536.37kW）；②求解过程中请勿点击软件，否则可能造
      成闪退")
1257.           parameters.setStyleSheet("font: 18px")
1258.           notice_layout.addWidget(parameters)
1259.
1260.           run_layout = QHBoxLayout()
1261.
1262.           self.text_max_energy = QLineEdit()
1263.           self.text_max_energy.setPlaceholderText('请输入现行公用工程用量，单位：kW')
1264.
1265.           btn_run_solve = QPushButton()
```

```
1266.           btn_run_solve.setText('求解')
1267.           btn_run_solve.clicked.connect(self.HEN_solve)
1268.
1269.           run_layout.addWidget(self.text_max_energy)
1270.           run_layout.addWidget(btn_run_solve)
1271.           notice_layout.addLayout(run_layout)
1272.           # notice_layout.addStretch()
1273.
1274.
1275.
1276.
1277.           # self.HEN = QWidget()
1278.           #
1279.           # self.HEN_draw = QPainter(self.HEN)
1280.           # self.HEN_draw.begin(self.HEN)
1281.           # self.pen = QPen(Qt.red, Qt.SolidLine)
1282.           # self.HEN_draw.setPen(self.pen)
1283.           # self.HEN_draw.drawLine(20,20,200,20)
1284.           # self.HEN_draw.end()
1285.           design_layout.addLayout(notice_layout)
1286.           # design_layout.addStretch()
1287.           self.design_console = QTextEdit()
1288.           self.design_console.setReadOnly(True)
1289.           design_layout.addWidget(self.design_console)
1290.           # design_layout.addStretch()
1291.           # wigglyWidget = builtPaintWidget()
```

```
1292.           # design_layout.addWidget(wigglyWidget)
1293.           design_btn_layout = QHBoxLayout()
1294.           btn_HEN_draw = QPushButton()
1295.           btn_HEN_draw.setText('查看换热网络图像及数据')
1296.           btn_HEN_draw.clicked.connect(self.show_child)
1297.           # btn_HEN_table = QPushButton()
1298.           # btn_HEN_table.setText('查看换热数据')
1299.           design_btn_layout.addWidget(btn_HEN_draw)
1300.           # design_btn_layout.addWidget(btn_HEN_table)
1301.           design_layout.addLayout(design_btn_layout)
1302.           self.right_frame_design.setLayout(design_layout)
1303.
1304.           self.right_frame_design.hide()
1305.           #不合理换热
1306.           self.right_frame_nonsense = QFrame(self.right_widget)
1307.           self.right_frame_nonsense.setStyleSheet('''QFrame{background-color: white;
1308.                                                             border-radius: 10px;
1309.                                                             }''')
1310.           self.right_frame_nonsense.setFrameShadow(QFrame.Raised)
1311.           self.right_frame_nonsense.setFrameShape(QFrame.Box)
1312.           self.right_frame_nonsense.setLineWidth(3)
1313.           self.right_frame_nonsense.setMidLineWidth(3)
1314.           self.right_frame_nonsense.resize(980, 500)
1315.           self.right_frame_nonsense.move(2, 98)
1316.           self.right_frame_nonsense.shadow = QtWidgets.QGraphicsDropShadowEffect()
1317.           self.right_frame_nonsense.shadow.setOffset(0, 0)  # 偏移
```

```
1318.           self.right_frame_nonsense.shadow.setBlurRadius(10)  # 阴影半径
1319.           self.right_frame_nonsense.shadow.setColor(QColor('black'))  # 阴影颜色
1320.           self.right_frame_nonsense.setGraphicsEffect(self.right_frame_nonsense.shadow)  # 将设置套用到widget 窗
    口中
1321.           # nonsense_text = QVBoxLayout()
1322.           # nonsense_text_view = WebEngine()
1323.           # nonsense_text_channel = QWebChannel()
1324.           # nonsense_text_handler = CallHandler()
1325.           # nonsense_text_channel.registerObject('PyHandler', nonsense_text_handler)
1326.           # nonsense_text_view.page().setWebChannel(nonsense_text_channel)
1327.           # nonsense_url_string = urllib.request.pathname2url(os.path.join(os.getcwd(), "js/index.html"))
1328.           # nonsense_text_view.load(QUrl(nonsense_url_string))
1329.           # nonsense_text.addWidget(nonsense_text_view)
1330.           # # nonsense_text_view.show()
1331.           # self.nonsense_text_view = nonsense_text_view
1332.           # self.right_frame_nonsense.setLayout(nonsense_text)
1333.           nonsense_layout = QVBoxLayout()
1334.
1335.           btn_find_nonsense = QPushButton()
1336.           btn_find_nonsense.setText('查找不合理换热器')
1337.           btn_find_nonsense.clicked.connect(self.event_find_nonsense)
1338.           nonsense_layout.addWidget(btn_find_nonsense)
1339.
1340.           self.nonsense_print = QTextEdit()
1341.           self.nonsense_print.setFixedHeight(400)
1342.           nonsense_layout.addWidget(self.nonsense_print)
```

```
1343.
1344.
1345.           # HE_add_layout = QVBoxLayout()
1346.           # CHE_layout = QHBoxLayout()
1347.           # CHE_text = QLabel()
1348.           # CHE_text.setFixedWidth(120)
1349.           # CHE_text.setText('冷公用工程换热器')
1350.           # CHE_layout.addWidget(CHE_text)
1351.           # CHE_new = QLineEdit()
1352.           # CHE_new.setPlaceholderText('输入新换热器编号')
1353.           # CHE_layout.addWidget(CHE_new)
1354.           # CHE_left = QLineEdit()
1355.           # CHE_left.setPlaceholderText('输入左侧换热器编号')
1356.           # CHE_layout.addWidget(CHE_left)
1357.           # CHE_right = QLineEdit()
1358.           # CHE_right.setPlaceholderText('输入右侧换热器编号')
1359.           # CHE_layout.addWidget(CHE_right)
1360.           # CHE_add_btn = QPushButton()
1361.           # CHE_add_btn.setFixedWidth(180)
1362.           # CHE_add_btn.setText('添加冷公用工程换热器')
1363.           # CHE_layout.addWidget(CHE_add_btn)
1364.           # HE_add_layout.addLayout(CHE_layout)
1365.           # HE_add_layout.addStretch()
1366.           #
1367.           # HHE_layout = QHBoxLayout()
1368.           # HHE_text = QLabel()
```

```
1369.           # HHE_text.setFixedWidth(120)
1370.           # HHE_text.setText('热公用工程换热器')
1371.           # HHE_layout.addWidget(HHE_text)
1372.           # HHE_new = QLineEdit()
1373.           # HHE_new.setPlaceholderText('输入新换热器编号')
1374.           # HHE_layout.addWidget(HHE_new)
1375.           # HHE_left = QLineEdit()
1376.           # HHE_left.setPlaceholderText('输入左侧换热器编号')
1377.           # HHE_layout.addWidget(HHE_left)
1378.           # HHE_right = QLineEdit()
1379.           # HHE_right.setPlaceholderText('输入右侧换热器编号')
1380.           # HHE_layout.addWidget(HHE_right)
1381.           # HHE_add_btn = QPushButton()
1382.           # HHE_add_btn.setFixedWidth(180)
1383.           # HHE_add_btn.setText('添加热公用工程换热器')
1384.           # HHE_layout.addWidget(HHE_add_btn)
1385.           # HE_add_layout.addLayout(HHE_layout)
1386.           # HE_add_layout.addStretch()
1387.
1388.           # HE_layout = QHBoxLayout()
1389.           # HE_text = QLabel()
1390.           # HE_text.setFixedWidth(120)
1391.           # HE_text.setText('冷热流股换热器')
1392.           # HE_layout.addWidget(HE_text)
1393.           # HE_new = QLineEdit()
1394.           # HE_new.setPlaceholderText('输入新换热器编号')
```

```
1395.           # HE_new.setFixedWidth(210)
1396.           # HE_layout.addWidget(HE_new)
1397.           # HE_text_layout=QVBoxLayout()
1398.           # HEH_text_layout = QHBoxLayout()
1399.           # HE_H_left = QLineEdit()
1400.           # HE_H_left.setPlaceholderText('输入热流股左侧换热器编号')
1401.           # HEH_text_layout.addWidget(HE_H_left)
1402.           # HE_H_right = QLineEdit()
1403.           # HE_H_right.setPlaceholderText('输入热流股右侧换热器编号')
1404.           # HEH_text_layout.addWidget(HE_H_right)
1405.           #
1406.           # HEC_text_layout = QHBoxLayout()
1407.           # HE_C_left = QLineEdit()
1408.           # HE_C_left.setPlaceholderText('输入冷流股左侧换热器编号')
1409.           # HEC_text_layout.addWidget(HE_C_left)
1410.           # HE_C_right = QLineEdit()
1411.           # HE_C_right.setPlaceholderText('输入冷流股右侧换热器编号')
1412.           # HEC_text_layout.addWidget(HE_C_right)
1413.           #
1414.           # HE_text_layout.addLayout(HEH_text_layout)
1415.           # HE_text_layout.addLayout(HEC_text_layout)
1416.           # HE_layout.addLayout(HE_text_layout)
1417.           # HE_add_btn = QPushButton()
1418.           # HE_add_btn.setFixedWidth(180)
1419.           # HE_add_btn.setText('添加冷热流股换热器')
1420.           # HE_layout.addWidget(HE_add_btn)
```

```
1421.           # HE_add_layout.addLayout(HE_layout)
1422.           #
1423.           # nonsense_layout.addStretch()
1424.           # nonsense_layout.addLayout(HE_add_layout)
1425.
1426.           self.right_frame_nonsense.setLayout(nonsense_layout)
1427.           self.right_frame_nonsense.hide()
1428.           #软件简介
1429.           self.right_frame_introduction = QFrame(self.right_widget)
1430.           self.right_frame_introduction.setStyleSheet('''QFrame{background-color: white;
1431.                                                               background-image:url(./image/bg);
1432.                                                               border-radius: 10px;
1433.                                                               }''')
1434.           self.right_frame_introduction.setFrameShadow(QFrame.Raised)
1435.           self.right_frame_introduction.setFrameShape(QFrame.Box)
1436.           self.right_frame_introduction.setLineWidth(3)
1437.           self.right_frame_introduction.setMidLineWidth(3)
1438.           self.right_frame_introduction.resize(980, 500)
1439.           self.right_frame_introduction.move(2, 98)
1440.           self.right_frame_introduction.shadow = QtWidgets.QGraphicsDropShadowEffect()
1441.           self.right_frame_introduction.shadow.setOffset(0, 0)  # 偏移
1442.           self.right_frame_introduction.shadow.setBlurRadius(10)  # 阴影半径
1443.           self.right_frame_introduction.shadow.setColor(QColor('black'))  # 阴影颜色
1444.           self.right_frame_introduction.setGraphicsEffect(self.right_frame_introduction.shadow)  # 将设置套用到
    widget 窗口中
1445.
```

```
            introduction_layout = QVBoxLayout()
            introdaction_text = QTextEdit()
            # introdaction_text.insertPlainText('onlyread')
            introdaction_text.setReadOnly(True)
            introdaction_text.setStyleSheet('''QTextEdit{color:black;
                                                          font: bold 20px;}''')
            # backgroung_pic = QtGui.QPalette()
            # backgroung_pic.setBrush(introdaction_text.backgroundRole(), QtGui.QBrush(QtGui.QPixmap("./image/logofull.png")))
            # introdaction_text.setPalette(backgroung_pic)
            introdaction_text.setMarkdown('''# 欢迎使用!
         本软件是为盈德气体荆门工厂煤制甲醇装置的合成工段及精馏工段所开发的专用软件，主要实现以下四个功能：
  <table>
            <td bgcolor=#FFFF00>1. 夹点分析
            </td>
        </table>
  <table>
            <td bgcolor=#FFFF00>2. 不合理换热分析
            </td>
        </table>
  <table>
            <td bgcolor=#FFFF00>3. 负荷转移分析
            </td>
        </table>
  <table>
```

```
            <td bgcolor=#FFFF00>4. 换热网络设计
            </td>
        </table>

    ### 1. 夹点分析
           通过数据模块接收装置流股信息，对现行网络匹配绘制温焓图，并进行夹点分析，得到换热网络最优匹配下的公用工程用量、节能潜力以及最大碳减排量。以最小换热温差 8℃为例：
    + 最小冷却公用工程用量为：50346kW
    + 最小加热公用工程用量为：48357kW
    + <font color="#dd0000">节能潜力为：15834kW</font>
    + <font color="#dd0000">最大碳减排量：15786.35kg/h</font>
    + 夹点温度为：68.00℃
    ### 2. 不合理换热分析
           主要实现<font color="#dd0000">查找现行换热网络内的不合理换热器。</font><br />
           主要不合理换热器表现形式有：
    + 跨越夹点的冷热流股换热器
    + 夹点之上的冷却公用工程换热器
    + 夹点之下的加热公用工程换热器
    ### 3. 负荷转移分析
           由于催化剂失活等导致的生产条件变化会使换热器负荷发生变动，为了使换热流股能达到目标温度，我们增加了负荷转移分析模块。<br />
           当流股数据发生变动时，该模块可以 <font color="#dd0000">检测负荷转移的所有可能路径，并计算可能发生的温度及负荷变化。</font>
    ### 4. 换热网络设计
           该模块可以实现<font color="#dd0000">以小于现行换热网络公用工程用量为目标，对流股数据进行换热网络自动设计。</font><br />
```

```
       该模型属于通用优化模型，既可以实现使换热网络的改造设计，又可以针对新换热网络进行设计，对现实生产具有重要意义。''')

            introduction_layout.addWidget(introdaction_text)

            self.right_frame_introduction.setLayout(introduction_layout)
            self.right_frame_introduction.hide()


            #主页
            self.right_frame_homepage = QFrame(self.right_widget)
            self.right_frame_homepage.setStyleSheet('''QFrame{background-color: white;
                                                               border-radius: 10px;
                                                               }''')
            self.right_frame_homepage.setFrameShadow(QFrame.Raised)
            self.right_frame_homepage.setFrameShape(QFrame.Box)
            self.right_frame_homepage.setLineWidth(3)
            self.right_frame_homepage.setMidLineWidth(3)
            self.right_frame_homepage.resize(980, 500)
            self.right_frame_homepage.move(2, 98)
            self.right_frame_homepage.shadow = QtWidgets.QGraphicsDropShadowEffect()
            self.right_frame_homepage.shadow.setOffset(0, 0)  # 偏移
            self.right_frame_homepage.shadow.setBlurRadius(10)  # 阴影半径
            self.right_frame_homepage.shadow.setColor(QColor('black'))  # 阴影颜色
            self.right_frame_homepage.setGraphicsEffect(self.right_frame_homepage.shadow)  # 将设置套用到widget窗口中
```

```
1516.
1517.
1518.           return self.right_widget
1519.
1520.       def HEN_solve(self):
1521.
1522.           start_time = time.time()
1523.           self.design_console.clear()
1524.           ###1.初始化数据
1525.
1526.           TCIN = []
1527.           TCOUT = []
1528.           FCPC = []
1529.           THIN = []
1530.           THOUT = []
1531.           FCPH = []
1532.           H = []
1533.           C = []
1534.           # print(self.table_widget.rowCount())
1535.           for i in [3, 4, 6, 0]:
1536.               for j in range(self.table_widget.rowCount()):
1537.                   in_t = float(self.table_widget.item(j, 3).text())
1538.                   out_t = float(self.table_widget.item(j, 4).text())
1539.                   if in_t >= out_t and i == 3:
1540.                       THIN.append(float(self.table_widget.item(j, i).text()))
1541.                       # print(TH_in)
```

```
1542.
1543.                     if in_t <= out_t and i == 3:
1544.                         TCIN.append(float(self.table_widget.item(j, i).text()))
1545.                     if in_t >= out_t and i == 4:
1546.                         THOUT.append(float(self.table_widget.item(j, i).text()))
1547.                     if in_t <= out_t and i == 4:
1548.                         TCOUT.append(float(self.table_widget.item(j, i).text()))
1549.                     if in_t >= out_t and i == 6:
1550.                         FCPH.append(float(self.table_widget.item(j, i).text()))
1551.                     if in_t <= out_t and i == 6:
1552.                         FCPC.append(float(self.table_widget.item(j, i).text()))
1553.             vapor_high = 245
1554.             vapor_low = 100
1555.             cws_high = 30
1556.             cws_low = 20
1557.             DT_min = 8
1558.             NH = len(FCPH)
1559.             NC = len(FCPC)
1560.             NS = max(NH,NC)
1561.             self.index = 0
1562.             population = []
1563.             Qmax = 0
1564.             for i in range(NH):
1565.                 Qmax = Qmax + (THIN[i] - THOUT[i]) * FCPH[i]
1566.             for j in range(NC):
1567.                 Qmax = Qmax + (TCOUT[j] - TCIN[j]) * FCPC[j]
```

```
            temp_obj = Qmax
            Q_pinch,load = self.Pinch_compute()
            print(load)
            if self.text_max_energy.text()!='':
                max_energy = float(self.text_max_energy.text())
            else:
                max_energy = load
            x = []
            y = []
            z = []
            # self.design_console.setText("种群初始化")
            cut_num = 0

            max_energy_temp = max_energy
            while (1):
                self.logic_fasible = 0
                while (self.logic_fasible == 0):
                    [Z_stages_kji, Q_stages_kji, Z_HU, Z_CU, Q_HU, Q_CU, T] = self.initialize(NH, NC, NS, THIN, THOUT, TCIN,
                                                                                              TCOUT, FCPC, FCPH, DT_min)
                # print("got in")
                obj1 = sum(Q_HU)
                obj2 = sum(Q_CU)
                obj_all = obj1+obj2
                Z = [Z_HU, Z_stages_kji, Z_CU]
```

```
                Q = [Q_HU, Q_stages_kji, Q_CU]
                obj = [obj_all,obj1,obj2]


                temp = {'idx': self.index, 'structure': Z, 'energy': Q, 'temputer': T, 'object': obj, 'rank': Non
    e,
                        'crowd': None, 'pSpD': None}


                temp_obj = obj_all

                self.index = self.index + 1

                if temp['object'][0]<max_energy:
                    cut_num = cut_num+1
                    if temp['object'][0]<max_energy_temp:
                        max_energy_temp = temp['object'][0]
                        self.information = temp
                if cut_num>=10:
                    break
            end_time = time.time()
            self.time_cost = end_time - start_time


            self.design_console.setStyleSheet("font:20px")
```

```
            self.design_console.setText("该换热系统理论最大公用工程用量为：%2.f kW；理论最小公用工程用量为：%2.f kW。共生成可行解%d 个，目标解的实际公用工程用量为：%.2f kW，其中加热公用工程用量为%.2f kW，冷却公用工程用量为%.2f kW。对于现行换热网络节省公用工程用量：%.2f kW，预计减少碳排放：%.2f kg/h。共计耗时%.2f s。
            " % (Qmax,Q_pinch,self.index,self.information['object'][0] ,self.information['object'][1],self.information['object'][2],max_energy-max_energy_temp,0.3021*(max_energy-max_energy_temp),self.time_cost))

        def initialize(self,NH, NC, NS, THIN, THOUT, TCIN, TCOUT, FCPC, FCPH, DT_min):

            self.logic_fasible = 1
            Z = []
            Z_stages_kji = []
            z_stage = []
            Z_CU = [0 for _ in range(NH)]
            Z_HU = [0 for _ in range(NC)]

            for k in range(NS):
                temp_judge = []
                temp_index = 1000
                for j in range(NC):
                    z_stage.append([])
                    temp_z = [0 for _ in range(NH)]
                    while (1):
                        temp_index = random.randint(0, 2 * NH)
                        if temp_index not in temp_judge:
                            if temp_index < NH:
                                temp_z[temp_index] = 1
```

```
                                temp_judge.append(temp_index)
                            break
                    z_stage[j] = temp_z
                Z_stages_kji.append(z_stage)
                z_stage = []
            # print(Z_stages_kji)
            forbidden = []
            for i in range(NH):
                for j in range(NC):
                    if TCIN[j] > THOUT[i] and TCOUT[j] > THIN[i]:
                        forbidden.append([j, i])
            for k in range(NS):
                for i in range(NH):
                    for j in range(NC):
                        if [j, i] in forbidden:
                            Z_stages_kji[k][j][i] = 0
            # print(Z_stages_kji)
            # print(forbidden)
            ###2.2 换热器负荷，保证了换热器负荷不会超过目标换热量，但不保证正好等于，需要判断约束
            lenth_i = [0 for _ in range(NH)]
            lenth_j = [0 for _ in range(NC)]
            for i in range(NH):
                for j in range(NC):
                    for k in range(NS):
                        if Z_stages_kji[k][j][i] == 1:
                            lenth_j[j] = lenth_j[j] + 1
```

```
                            lenth_i[i] = lenth_i[i] + 1
        # print(lenth_i, lenth_j)
        for lenth in lenth_i:
            if lenth > NS:
                self.logic_fasible = 0
        temp_len_ki = []
        temp_len_kj = []
        for k in range(NS):
            temp_len_ki.append([0 for _ in range(NH)])
            temp_len_kj.append([0 for _ in range(NH)])
            for i in range(NH):
                for j in range(NC):
                    if Z_stages_kji[k][j][i]:
                        temp_len_kj[k][j] = temp_len_kj[k][j] + 1
                        temp_len_ki[k][i] = temp_len_ki[k][i] + 1
        for k in temp_len_ki:
            for i in k:
                if i >= 2:
                    self.logic_fasible = 0
        for k in temp_len_kj:
            for j in k:
                if j >= 2:
                    self.logic_fasible = 0

        if self.logic_fasible:
            for i in range(NH):
```

```
                    if lenth_i[i] == 0:
                        Z_CU[i] = 1
                for j in range(NC):
                    if lenth_j[j] == 0:
                        Z_HU[j] = 1
                self.logic_fasible = 0
                while (self.logic_fasible == 0):
                    self.logic_fasible = 1
                    temp_leni = lenth_i[::]
                    temp_lenj = lenth_j[::]
                    Q_i = [0 for _ in range(NH)]
                    Q_j = [0 for _ in range(NC)]
                    Q_stages_kji = []
                    z_stage = []
                    for k in range(NS):
                        for j in range(NC):
                            temp_z = [0 for _ in range(NH)]
                            z_stage.append(temp_z)
                            # for j in range(NC):
                            #     if Z_stages_kij[k][i][j] == 1:
                            #         temp_z[j] = 1
                        Q_stages_kji.append(z_stage)
                        z_stage = []
                    Q_HU = Z_HU[::]
                    Q_CU = Z_CU[::]
                    THIN_temp = []
```

```
                    THOUT_temp = []
                    TCIN_temp = []
                    TCOUT_temp = []
                    for t in range(NH):
                        THIN_temp.append(THIN[t])
                        THOUT_temp.append(THOUT[t])
                    for t in range(NC):
                        TCIN_temp.append(TCIN[t])
                        TCOUT_temp.append(TCOUT[t])
                    if True:
                        for i in range(NH):
                            for j in range(NC):
                                for k in range(NS):
                                    if Z_stages_kji[k][j][i] == 1:
                                        if THOUT_temp[i] < THIN_temp[i] and TCIN_temp[j] < TCOUT_temp[j]:
                                            if THIN_temp[i] >= TCOUT_temp[j] and THOUT_temp[i] >= TCIN_temp[j]:
                                                Qmin = min((TCOUT_temp[j] - TCIN_temp[j]) * FCPC[j],
                                                           (THIN_temp[i] - THOUT_temp[i]) * FCPH[i])
                                                q_ht_out = THIN_temp[i] - Qmin / FCPH[i]
                                                q_ct_in = TCOUT_temp[j] - Qmin / FCPC[j]
                                            elif THIN_temp[i] >= TCOUT_temp[j]:
                                                q_ht_out = TCIN_temp[j]
                                                Qmin = min((TCOUT_temp[j] - TCIN_temp[j]) * FCPC[j],
                                                           (THIN_temp[i] - q_ht_out) * FCPH[i])
                                                q_ht_out = THIN_temp[i] - Qmin / FCPH[i]
                                                q_ct_in = TCOUT_temp[j] - Qmin / FCPC[j]
```

```
                elif THOUT_temp[i] >= TCIN_temp[j]:
                    q_ct_in = THIN_temp[i]
                    Qmin = min((q_ct_in - TCIN_temp[j]) * FCPC[j],
                               (THIN_temp[i] - THOUT_temp[i]) * FCPH[i])
                    q_ht_out = THIN_temp[i] - Qmin / FCPH[i]
                    q_ct_in = TCOUT_temp[j] - Qmin / FCPC[j]
                else:
                    Z_stages_kji[k][j][i] = 0
                    temp_leni[i] = temp_leni[i] - 1
                    temp_lenj[j] = temp_lenj[j] - 1
            else:
                Z_stages_kji[k][j][i] = 0
                temp_leni[i] = temp_leni[i] - 1
                temp_lenj[j] = temp_lenj[j] - 1
            if Z_stages_kji[k][j][i]:
                Q_stages_kji[k][j][i] = Qmin
                THIN_temp[i] = q_ht_out
                TCOUT_temp[j] = q_ct_in
                temp_leni[i] = temp_leni[i] - 1
                temp_lenj[j] = temp_lenj[j] - 1
                Q_i[i] = Q_i[i] + Q_stages_kji[k][j][i]
                Q_j[j] = Q_j[j] + Q_stages_kji[k][j][i]
            # elif (temp_lenj[j] == 1 and Z_HU[j] == 0) or (temp_leni[i] == 1 and Z_C
    U[i] == 0):
            #     Q_stages_kji[k][j][i] = min((TCOUT[j]-TCIN[j])*FCPC[j]-Q_j[j], (THI
    N[i]-THOUT[i])*FCPH[i]-Q_i[i])
```

```
                                        #     temp_leni[i] = temp_leni[i] - 1
                                        #     temp_lenj[j] = temp_lenj[j] - 1
                                        #     Q_i[i] = Q_i[i] + Q_stages_kji[k][j][i]
                                        #     Q_j[j] = Q_j[j] + Q_stages_kji[k][j][i]
                                        # else:
                                        #     Q_stages_kji[k][j][i] = random.uniform(0, min( (TCOUT[j]-TCIN[j])*F
   CPC[j]-Q_j[j], (THIN[i]-THOUT[i])*FCPH[i]-Q_i[i]))
                                        #     Q_i[i] = Q_i[i] + Q_stages_kji[k][j][i]
                                        #     Q_j[j] = Q_j[j] + Q_stages_kji[k][j][i]
                                        #     temp_leni[i] = temp_leni[i] - 1
                                        #     temp_lenj[j] = temp_lenj[j] - 1
                    # else:
                    #     for i in range(NH-1,-1,-1):
                    #         for j in range(NC-1,-1,-1):
                    #             for k in range(NS-1,-1,-1):
                    #                 if Z_stages_kji[k][j][i] == 1:
                    #                     # if temp_leni[i] == lenth_i[i] or temp_lenj[j] == lenth_j[j]:
                    #                     #     Q_stages_kji[k][j][i] = random.uniform(0, min((TCOUT[j]-TCIN[j])*
   FCPC[j]-Q_j[j], (THIN[i]-THOUT[i])*FCPH[i]-Q_i[i]))
                    #                     #     temp_leni[i] = temp_leni[i] - 1
                    #                     #     temp_lenj[j] = temp_lenj[j] - 1
                    #                     #     Q_i[i] = Q_i[i] + Q_stages_kji[k][j][i]
                    #                     #     Q_j[j] = Q_j[j] + Q_stages_kji[k][j][i]
                    #                     # elif (temp_lenj[j] == 1 and Z_HU[j] == 0) or (temp_leni[i] == 1 and Z
   _CU[i] == 0):
```

```
1789.                   #                       #     Q_stages_kji[k][j][i] = min((TCOUT[j]-TCIN[j])*FCPC[j]-Q_j[j], (T
   HIN[i]-THOUT[i])*FCPH[i]-Q_i[i])
1790.                   #                       #     temp_leni[i] = temp_leni[i] - 1
1791.                   #                       #     temp_lenj[j] = temp_lenj[j] - 1
1792.                   #                       #     Q_i[i] = Q_i[i] + Q_stages_kji[k][j][i]
1793.                   #                       #     Q_j[j] = Q_j[j] + Q_stages_kji[k][j][i]
1794.                   #                       # else:
1795.                   #                       #     Q_stages_kji[k][j][i] = random.uniform(0, min( (TCOUT[j]-TCIN[j])
   *FCPC[j]-Q_j[j], (THIN[i]-THOUT[i])*FCPH[i]-Q_i[i]))
1796.                   #                       #     Q_i[i] = Q_i[i] + Q_stages_kji[k][j][i]
1797.                   #                       #     Q_j[j] = Q_j[j] + Q_stages_kji[k][j][i]
1798.                   #                       #     temp_leni[i] = temp_leni[i] - 1
1799.                   #                       #     temp_lenj[j] = temp_lenj[j] - 1
1800.                   #                       if THIN_temp[i] >= TCOUT_temp[j] and THOUT_temp[i] >= TCIN_temp[j]:
1801.                   #                           Qmin = min((TCOUT_temp[j] - TCIN_temp[j]) * FCPC[j] ,
1802.                   #                                      (THIN_temp[i] - THOUT_temp[i]) * FCPH[i] )
1803.                   #                           q_ht_in = THOUT_temp[i] + Qmin / FCPH[i]
1804.                   #                           q_ct_out = TCIN_temp[j] + Qmin / FCPC[j]
1805.                   #                       elif THIN_temp[i] >= TCOUT_temp[j]:
1806.                   #                           q_ht_in = TCIN_temp[j]
1807.                   #                           Qmin = min((TCOUT_temp[j] - TCIN_temp[j]) * FCPC[j] ,
1808.                   #                                      (q_ht_in - THOUT_temp[i]) * FCPH[i] )
1809.                   #                           q_ht_in = THOUT_temp[i] + Qmin / FCPH[i]
1810.                   #                           q_ct_out = TCIN_temp[j] + Qmin / FCPC[j]
1811.                   #                       elif THOUT_temp[i] >= TCIN_temp[j]:
1812.                   #                           q_ct_out = THIN_temp[i]
```

```
                    #                         Qmin = min((q_ct_out- TCIN_temp[j]) * FCPC[j] ,
                    #                                    (THIN_temp[i] - THOUT_temp[i]) * FCPH[i] )
                    #                         q_ht_in = THOUT_temp[i] + Qmin / FCPH[i]
                    #                         q_ct_out = TCIN_temp[j] + Qmin / FCPC[j]
                    #                     else:
                    #                         Z_stages_kji[k][j][i] = 0
                    #                         temp_leni[i] = temp_leni[i] - 1
                    #                         temp_lenj[j] = temp_lenj[j] - 1
                    #                     if Z_stages_kji[k][j][i]:
                    #                         Q_stages_kji[k][j][i] = Qmin
                    #                         THOUT_temp[i] = q_ht_in
                    #                         TCIN_temp[j] = q_ct_out
                    #                         temp_leni[i] = temp_leni[i] - 1
                    #                         temp_lenj[j] = temp_lenj[j] - 1
                    #                         Q_i[i] = Q_i[i] + Q_stages_kji[k][j][i]
                    #                         Q_j[j] = Q_j[j] + Q_stages_kji[k][j][i]
                    # elif index%2 ==0:
                    #     for i in range(NH):
                    #         for j in range(NC):
                    #             for k in range(NS):
                    #                 if Z_stages_kji[k][j][i] == 1:
                    #                     if temp_leni[i] == lenth_i[i] or temp_lenj[j] == lenth_j[j]:
                    #                         if temp_leni[i] == lenth_i[i] or temp_lenj[j] == lenth_j[j]:
                    #                             Q_stages_kji[k][j][i] = random.uniform(0, min((TCOUT[j]-TCIN[j]
    )*FCPC[j]-Q_j[j], (THIN[i]-THOUT[i])*FCPH[i]-Q_i[i]))
                    #                         temp_leni[i] = temp_leni[i] - 1
```

```
                    #                                 temp_lenj[j] = temp_lenj[j] - 1
                    #                                 Q_i[i] = Q_i[i] + Q_stages_kji[k][j][i]
                    #                                 Q_j[j] = Q_j[j] + Q_stages_kji[k][j][i]
                    #                             elif (temp_lenj[j] == 1 and Z_HU[j] == 0) or (temp_leni[i] == 1 and
    Z_CU[i] == 0):
                    #                                 Q_stages_kji[k][j][i] = min((TCOUT[j]-TCIN[j])*FCPC[j]-Q_j[j],
   (THIN[i]-THOUT[i])*FCPH[i]-Q_i[i])
                    #                                 temp_leni[i] = temp_leni[i] - 1
                    #                                 temp_lenj[j] = temp_lenj[j] - 1
                    #                                 Q_i[i] = Q_i[i] + Q_stages_kji[k][j][i]
                    #                                 Q_j[j] = Q_j[j] + Q_stages_kji[k][j][i]
                    #                             else:
                    #                                 Q_stages_kji[k][j][i] = random.uniform(0, min( (TCOUT[j]-TCIN[j
   ])*FCPC[j]-Q_j[j], (THIN[i]-THOUT[i])*FCPH[i]-Q_i[i]))
                    #                                 Q_i[i] = Q_i[i] + Q_stages_kji[k][j][i]
                    #                                 Q_j[j] = Q_j[j] + Q_stages_kji[k][j][i]
                    #                                 temp_leni[i] = temp_leni[i] - 1
                    #                                 temp_lenj[j] = temp_lenj[j] - 1
                    # else:
                    #     for i in range(NH - 1, -1, -1):
                    #         for j in range(NC - 1, -1, -1):
                    #             for k in range(NS - 1, -1, -1):
                    #                 if Z_stages_kji[k][j][i] == 1:
                    #                     if temp_leni[i] == lenth_i[i] or temp_lenj[j] == lenth_j[j]:
                    #                         Q_stages_kji[k][j][i] = random.uniform(0, min((TCOUT[j]-TCIN[j])*FC
   PC[j]-Q_j[j], (THIN[i]-THOUT[i])*FCPH[i]-Q_i[i]))
```

```
1860.                     #                         temp_leni[i] = temp_leni[i] - 1
1861.                     #                         temp_lenj[j] = temp_lenj[j] - 1
1862.                     #                         Q_i[i] = Q_i[i] + Q_stages_kji[k][j][i]
1863.                     #                         Q_j[j] = Q_j[j] + Q_stages_kji[k][j][i]
1864.                     #                     elif (temp_lenj[j] == 1 and Z_HU[j] == 0) or (temp_leni[i] == 1 and Z_C
      U[i] == 0):
1865.                     #                         Q_stages_kji[k][j][i] = min((TCOUT[j]-TCIN[j])*FCPC[j]-Q_j[j], (THI
      N[i]-THOUT[i])*FCPH[i]-Q_i[i])
1866.                     #                         temp_leni[i] = temp_leni[i] - 1
1867.                     #                         temp_lenj[j] = temp_lenj[j] - 1
1868.                     #                         Q_i[i] = Q_i[i] + Q_stages_kji[k][j][i]
1869.                     #                         Q_j[j] = Q_j[j] + Q_stages_kji[k][j][i]
1870.                     #                     else:
1871.                     #                         Q_stages_kji[k][j][i] = random.uniform(0, min( (TCOUT[j]-TCIN[j])*F
      CPC[j]-Q_j[j], (THIN[i]-THOUT[i])*FCPH[i]-Q_i[i]))
1872.                     #                         Q_i[i] = Q_i[i] + Q_stages_kji[k][j][i]
1873.                     #                         Q_j[j] = Q_j[j] + Q_stages_kji[k][j][i]
1874.                     #                         temp_leni[i] = temp_leni[i] - 1
1875.                     #                         temp_lenj[j] = temp_lenj[j] - 1
1876.
1877.                     for i in range(NH):
1878.                         for j in range(NC):
1879.                             for k in range(NS):
1880.                                 if Z_stages_kji[k][j][i] == 0:
1881.                                     Q_stages_kji[k][j][i] = 0
1882.                                 if Q_stages_kji[k][j][i] < 0.1:
```

```
                                    Z_stages_kji[k][j][i] = 0

                    # print(Q_stages_kji)

                    for j in range(NC):
                        if Z_HU[j]:
                            Q_HU[j] = (TCOUT[j] - TCIN[j]) * FCPC[j] - Q_j[j]
                    for i in range(NH):
                        if Z_CU[i]:
                            Q_CU[i] = (THIN[i] - THOUT[i]) * FCPH[i] - Q_i[i]
                    for i in range(NH):
                        if abs(Q_i[i] + Q_CU[i] - (THIN[i] - THOUT[i]) * FCPH[i]) > 0.1:
                            Z_CU[i] = 1
                            Q_CU[i] = (THIN[i] - THOUT[i]) * FCPH[i] - Q_i[i]
                            self.logic_fasible = 1
                            # print(i, (Q_i[i]+Q_CU[i]-(THIN[i]-THOUT[i])*FCPH[i]))
                    for j in range(NC):
                        if abs(Q_j[j] + Q_HU[j] - (TCOUT[j] - TCIN[j]) * FCPC[j]) > 0.1:
                            Z_HU[j] = 1
                            Q_HU[j] = (TCOUT[j] - TCIN[j]) * FCPC[j] - Q_j[j]
                            self.logic_fasible = 1
                            # print(j,(Q_j[j]+Q_HU[j]-(TCOUT[j]-TCIN[j])*FCPC[j]))
                    # print(Q_CU,Q_HU)
                    self.logic_fasible = 1
                # for k in range(NS):
                #     for i in range(NH):
```

```
1909.                #         for j in range(NC):
1910.                #             if Q_stages_kji[k][j][i] < 1:
1911.                #                 Z_stages_kji[k][j][i]=0
1912.                ###2.3 计算stage 温度，不保证温度满足约束，需要判断约束
1913.                if self.logic_fasible:
1914.                    THIN_stages_kij = Z_stages_kji[::]
1915.                    THOUT_stages_kij = Z_stages_kji[::]
1916.                    TCIN_stages_kij = Z_stages_kji[::]
1917.                    TCOUT_stages_kij = Z_stages_kji[::]
1918.                    TCIN_HU = Z_HU[::]
1919.                    TCOUT_HU = Z_HU[::]
1920.                    THIN_CU = Z_CU[::]
1921.                    THOUT_CU = Z_CU[::]
1922.                    temp_leni = lenth_i[::]
1923.                    temp_lenj = lenth_j[::]
1924.                    temp_len_ki = []
1925.                    temp_len_kj = []
1926.                    temp_Q_ki = []
1927.                    temp_Q_kj = []
1928.                    TH_stages_left = []
1929.                    TH_stages_right = []
1930.                    TC_stages_left = []
1931.                    TC_stages_right = []
1932.                    for j in range(NC):
1933.                        if Z_HU[j]:
1934.                            TCOUT_HU[j] = TCOUT[j]
```

```
                            TCIN_HU[j] = TCOUT[j] - Q_HU[j] / FCPC[j]
                    for i in range(NH):
                        if Z_CU[i]:
                            THOUT_CU[i] = THOUT[i]
                            THIN_CU[i] = Q_CU[i] / FCPH[i] + THOUT[i]
                    for k in range(NS):
                        temp_len_ki.append([0 for _ in range(NH)])
                        temp_len_kj.append([0 for _ in range(NH)])
                        temp_Q_ki.append([0 for _ in range(NH)])
                        temp_Q_kj.append([0 for _ in range(NC)])
                        TH_stages_left.append([0 for _ in range(NH)])
                        TH_stages_right.append([0 for _ in range(NH)])
                        TC_stages_left.append([0 for _ in range(NC)])
                        TC_stages_right.append([0 for _ in range(NC)])
                        for i in range(NH):
                            for j in range(NC):
                                if Z_stages_kji[k][j][i]:
                                    temp_len_kj[k][j] = temp_len_kj[k][j] + 1
                                    temp_len_ki[k][i] = temp_len_ki[k][i] + 1
                                    temp_Q_ki[k][i] = Q_stages_kji[k][j][i]
                                    temp_Q_kj[k][j] = Q_stages_kji[k][j][i]
                    for k in range(NS):
                        for i in range(NH):
                            for j in range(NC):
                                if k == 0:
                                    TH_stages_left[k][i] = THIN[i]
```

```
                                    if Z_HU[j]:
                                        TC_stages_left[k][j] = TCIN_HU[j]
                                    else:
                                        TC_stages_left[k][j] = TCOUT[j]
                                    TH_stages_right[k][i] = TH_stages_left[k][i] - temp_Q_ki[k][i] / FCPH[i]
                                    TC_stages_right[k][j] = TC_stages_left[k][j] - temp_Q_kj[k][j] / FCPC[j]
                                else:
                                    TH_stages_left[k][i] = TH_stages_right[k - 1][i]
                                    TH_stages_right[k][i] = TH_stages_left[k][i] - temp_Q_ki[k][i] / FCPH[i]
                                    TC_stages_left[k][j] = TC_stages_right[k - 1][j]
                                    TC_stages_right[k][j] = TC_stages_left[k][j] - temp_Q_kj[k][j] / FCPC[j]
                    # print(temp_len_kj,temp_len_ki)
                    for k in temp_len_ki:
                        for i in k:
                            if i >= 2:
                                self.logic_fasible = 0
                    for k in temp_len_kj:
                        for j in k:
                            if j >= 2:
                                self.logic_fasible = 0
                    # print("###################################")
                    # print(temp_len_ki)
                    k = NS - 1
                    for i in range(NH):
                        if abs(TH_stages_right[k][i] - THOUT[i]) > 0.1 and abs(TH_stages_right[k][i] - THIN_CU[i]
    ) > 0.1:
```

```
                            # print(temp_len_ki)
                            # print(TH_stages_right[1])
                            self.logic_fasible = 0
                    for j in range(NC):
                        if abs(TC_stages_right[k][j] - TCIN[j]) > 0.1:
                            # print(TC_stages_right[1])
                            # print(TCIN)
                            self.logic_fasible = 0
                    for k in range(NS):
                        for i in range(NH):
                            for j in range(NC):
                                if Z_stages_kji[k][j][i]:
                                    if TH_stages_left[k][i] - TC_stages_left[k][j] < 0:
                                        self.logic_fasible = 0
                                        # print(k,i,j,TH_stages_left[k][i] - TC_stages_left[k][j] )
                                    if TH_stages_right[k][i] - TC_stages_right[k][j] < 0:
                                        self.logic_fasible = 0
                                        # print(k,i,j,TH_stages_right[k][i] - TC_stages_right[k][j])
                    T = [TH_stages_left, TH_stages_right, THIN_CU, TC_stages_left, TCIN_HU, TC_stages_right]
            return Z_stages_kji, Q_stages_kji, Z_HU, Z_CU, Q_HU, Q_CU, T

        def show_child(self):
            # print("子窗口")
            if self.isset('self.information'):
                child_window = child(self.information,self.table_widget)
                # apply_stylesheet(child_window, theme='light_blue.xml')
```

```
2012.                 child_window.exec_()
2013.
2014.         def isset(self,v):
2015.             try:
2016.                 type(eval(v))
2017.             except:
2018.                 return 0
2019.             else:
2020.                 return 1
2021.
2022.         def switch_homepage(self):
2023.             self.right_frame_pics.hide()
2024.             self.right_frame_statistic.hide()
2025.             self.right_frame_design.hide()
2026.             self.right_frame_load.hide()
2027.             self.right_frame_pinch.hide()
2028.             self.right_frame_nonsense.hide()
2029.             self.right_frame_introduction.hide()
2030.             self.right_frame_homepage.show()
2031.
2032.
2033.         def switch_pics(self):
2034.             self.right_frame_statistic.hide()
2035.             self.right_frame_design.hide()
2036.             self.right_frame_load.hide()
2037.             self.right_frame_pinch.hide()
```

```
2038.             self.right_frame_homepage.hide()
2039.             self.right_frame_nonsense.hide()
2040.             self.right_frame_introduction.hide()
2041.             self.right_frame_pics.show()
2042.
2043.         def switch_statistic(self):
2044.             self.right_frame_pics.hide()
2045.             self.right_frame_design.hide()
2046.             self.right_frame_load.hide()
2047.             self.right_frame_pinch.hide()
2048.             self.right_frame_homepage.hide()
2049.             self.right_frame_nonsense.hide()
2050.             self.right_frame_introduction.hide()
2051.             self.right_frame_statistic.show()
2052.         def switch_pinch(self):
2053.             self.right_frame_pics.hide()
2054.             self.right_frame_statistic.hide()
2055.             self.right_frame_design.hide()
2056.             self.right_frame_load.hide()
2057.             self.right_frame_homepage.hide()
2058.             self.right_frame_nonsense.hide()
2059.             self.right_frame_introduction.hide()
2060.             self.right_frame_pinch.show()
2061.         def switch_load(self):
2062.             self.right_frame_pics.hide()
2063.             self.right_frame_statistic.hide()
```

```
2064.             self.right_frame_design.hide()
2065.             self.right_frame_pinch.hide()
2066.             self.right_frame_homepage.hide()
2067.             self.right_frame_nonsense.hide()
2068.             self.right_frame_introduction.hide()
2069.             self.right_frame_load.show()
2070.         def switch_design(self):
2071.             self.right_frame_pics.hide()
2072.             self.right_frame_statistic.hide()
2073.             self.right_frame_load.hide()
2074.             self.right_frame_pinch.hide()
2075.             self.right_frame_homepage.hide()
2076.             self.right_frame_nonsense.hide()
2077.             self.right_frame_introduction.hide()
2078.             self.right_frame_design.show()
2079.         def switch_nonsense(self):
2080.             self.right_frame_pics.hide()
2081.             self.right_frame_statistic.hide()
2082.             self.right_frame_load.hide()
2083.             self.right_frame_pinch.hide()
2084.             self.right_frame_homepage.hide()
2085.             self.right_frame_design.hide()
2086.             self.right_frame_introduction.hide()
2087.             self.right_frame_nonsense.show()
2088.         def switch_introduction(self):
2089.             self.right_frame_pics.hide()
```

```
2090.             self.right_frame_statistic.hide()
2091.             self.right_frame_load.hide()
2092.             self.right_frame_pinch.hide()
2093.             self.right_frame_homepage.hide()
2094.             self.right_frame_design.hide()
2095.             self.right_frame_nonsense.hide()
2096.             self.right_frame_introduction.show()
2097.
2098.
2099.         def switch_last(self):
2100.             self.image_index = self.image_index - 1
2101.             if self.image_index < 1:
2102.                 self.image_index = 1
2103.             if self.image_index == 1:
2104.                 self.img2.hide()
2105.                 self.img3.hide()
2106.                 self.img1.show()
2107.                 self.img4.hide()
2108.             if self.image_index == 2:
2109.                 self.img1.hide()
2110.                 self.img3.hide()
2111.                 self.img2.show()
2112.                 self.img4.hide()
2113.             if self.image_index == 3:
2114.                 self.img2.hide()
2115.                 self.img1.hide()
```

```
2116.                 self.img3.show()
2117.                 self.img4.hide()
2118.             if self.image_index == 4:
2119.                 self.img2.hide()
2120.                 self.img1.hide()
2121.                 self.img3.hide()
2122.                 self.img4.show()
2123.         def switch_next(self):
2124.             self.image_index = self.image_index + 1
2125.             if self.image_index > 4:
2126.                 self.image_index = 4
2127.             if self.image_index == 1:
2128.                 self.img2.hide()
2129.                 self.img3.hide()
2130.                 self.img1.show()
2131.                 self.img4.hide()
2132.             if self.image_index == 2:
2133.                 self.img1.hide()
2134.                 self.img3.hide()
2135.                 self.img2.show()
2136.                 self.img4.hide()
2137.             if self.image_index == 3:
2138.                 self.img2.hide()
2139.                 self.img1.hide()
2140.                 self.img3.show()
2141.                 self.img4.hide()
```

```
2142.             if self.image_index == 4:
2143.                 self.img2.hide()
2144.                 self.img1.hide()
2145.                 self.img3.hide()
2146.                 self.img4.show()
2147.
2148.         def event_btn_newline(self):
2149.
2150.             current_row_count = self.table_widget.rowCount()
2151.             self.table_widget.insertRow(current_row_count)
2152.             cell = QTableWidgetItem()
2153.             current_column_count = self.table_widget.columnCount()
2154.             for i in range(current_column_count - 1):
2155.                 self.color_signal = 3
2156.                 self.change_signal = 1
2157.                 self.table_widget.setItem(current_row_count, i, cell)
2158.             self.color_signal = 1
2159.             self.change_signal = 0
2160.
2161.
2162.         def select_file(self):
2163.             """选择文件对话框"""
2164.             # QFileDialog 组件定义
2165.             fileDialog = QFileDialog(self)
2166.             # QFileDialog 组件设置
2167.             fileDialog.setWindowTitle("选择上传文件")              # 设置对话框标题
```

```
2168.             fileDialog.setFileMode(QFileDialog.AnyFile)  # 设置能打开文件的格式
2169.             fileDialog.setDirectory(r'C:')  # 设置默认打开路径
2170.             fileDialog.setNameFilter("Json (*.json)")  # 按文件名过滤
2171.             file_path = fileDialog.exec()                # 窗口显示，返回文件路径
2172.             if file_path and fileDialog.selectedFiles():
2173.                 print("选择文件成功：{}".format(fileDialog.selectedFiles()[0]))
2174.                 print(fileDialog.selectedFiles()[0])
2175.                 row_count = self.table_widget.rowCount()
2176.                 while 1:
2177.                     self.table_widget.removeRow(0)
2178.                     row_count = row_count - 1
2179.                     if row_count < 0:
2180.                         break
2181.                 import json
2182.                 with open(fileDialog.selectedFiles()[0], mode='r', encoding='utf-8') as f:
2183.                     data = f.read()
2184.                 data_list = json.loads(data)
2185.                 row_count = self.table_widget.rowCount()
2186.                 while 1:
2187.                     self.table_widget.removeRow(0)
2188.                     row_count = row_count - 1
2189.                     if row_count < 0:
2190.                         break
2191.                 current_row_count = self.table_widget.rowCount()
2192.                 for row_list in data_list:
2193.                     self.table_widget.insertRow(current_row_count)
```

```
                    for i, statistic in enumerate(row_list):
                        cell = QTableWidgetItem(str(statistic))
                        self.table_widget.setItem(current_row_count, i, cell)
                    current_row_count += 1
                self.forbidden_color=0


        def save_file(self):
            row_count = self.table_widget.rowCount()
            column_count = self.table_widget.columnCount()
            data = [ [] for _ in range(column_count)]
            for i in range(column_count):
                for j in range(row_count):
                    data[i].append(self.table_widget.item(j,i).text())
            import pandas as pd
            data_set = dict()
            table_header = ["流股编号","流股描述","换热器编号","进口温度/℃","出口温度/℃","流量/(kg/h)","平均热容流率/(kg/℃)","热负荷/kW","换热介质"]
            for l in range(column_count):
                data_set[table_header[l]]=data[l]
            df = pd.DataFrame(data_set)
            """文件另存对话框"""
            file_path, file_type = QFileDialog.getSaveFileName(self, '保存文件', os.getcwd(), 'Excel (*.xlsx)')
            if file_path and file_type:
                content = '这是一段文本'
                with open(file_path, 'w', encoding='utf-8') as f:
```

```
                    f.write(content)
                print("保存文件成功：{}".format(file_path))
                writer = pd.ExcelWriter(file_path, engine='xlsxwriter')
                df.to_excel(writer)
                writer.save()
        def Pinch_Draw(self):
            TC_in = []
            TC_out = []
            CPC = []
            TH_in = []
            TH_out = []
            CPH = []
            # print(self.table_widget.rowCount())
            for i in [3, 4, 6]:
                for j in range(self.table_widget.rowCount()):
                    in_t = float(self.table_widget.item(j, 3).text())
                    out_t = float(self.table_widget.item(j, 4).text())
                    if in_t>=out_t and i == 3:
                        TH_in.append(self.table_widget.item(j, i).text())
                        # print(TH_in)

                    if in_t<=out_t and i == 3:
                        TC_in.append(self.table_widget.item(j, i).text())
                    if in_t>=out_t and i == 4:
                        TH_out.append(self.table_widget.item(j, i).text())
                    if in_t<=out_t and i == 4:
```

```
                    TC_out.append(self.table_widget.item(j, i).text())
                if in_t>=out_t and i == 6:
                    CPH.append(self.table_widget.item(j, i).text())
                if in_t<=out_t and i == 6:
                    CPC.append(self.table_widget.item(j, i).text())
        # TC_in = [43, 40, 73, 74.3, 123, 109.2, 105]
        # TC_out = [208, 61, 73.8, 89, 126, 109.3, 105.1]
        # CPC = [241.54, 63.81, 20300, 81.9, 13090, 359810, 10680]
        # TH_in = [242, 102, 72, 57, 125, 113, 117.4, 117.3, 68, 51, 69, 109]
        # TH_out = [102, 30, 57, 54, 102, 44, 117.3, 112.6, 53, 31, 49, 40]
        # CPH = [284.67, 25.64, 983.07, 28.33, 52.35, 33.01, 359810, 144.47, 2529, 27.65, 56.3, 9.75]
        DT_min = np.array([self.DTmin]).astype(float)[0]
        # print(DT_min)
        # TC_in=[20,80]
        # TC_out=[135,140]
        # TH_in=[170,150]
        # TH_out=[60,30]
        # DT_min=10
        # CPC=[2,4]
        # CPH=[3,1.5]
        # 创建虚拟温度
        self.TC_in_im = TC_in_im = np.array(TC_in).astype(float) + 0.5 * DT_min
        self.TC_out_im = TC_out_im = np.array(TC_out).astype(float) + 0.5 * DT_min
        self.TH_in_im = TH_in_im = np.array(TH_in).astype(float) - 0.5 * DT_min
        self.TH_out_im = TH_out_im = np.array(TH_out).astype(float) - 0.5 * DT_min
        # print(1)
```

```
            CPC = np.array(CPC).astype(float)
            CPH = np.array(CPH).astype(float)
            # 对温度去重并排序,计算温差
            TC_in = np.array(TC_in).astype(float)
            TC_out = np.array(TC_out).astype(float)
            TH_in = np.array(TH_in).astype(float)
            TH_out = np.array(TH_out).astype(float)

            TC_uni = np.unique([TC_in, TC_out])
            TH_uni = np.unique([TH_in, TH_out])
            # print(CPC)


            num_tc_uni = len(TC_uni)
            num_th_uni = len(TH_uni)
            DTC = np.zeros([num_tc_uni]).astype(float)
            DTH = np.zeros([num_th_uni]).astype(float)

            for i in range(num_tc_uni):
                DTC[i - 1] = TC_uni[i - 1] - TC_uni[i]
            for i in range(num_th_uni):
                DTH[i - 1] = TH_uni[i - 1] - TH_uni[i]

            ############计算各个温区的热负荷
            HeatLoad_C = np.zeros(num_tc_uni - 1)
            HeatLoad_H = np.zeros(num_th_uni - 1)
```

```
2297.           num_tc = len(TC_in)
2298.           num_th = len(TH_in)
2299.           TC_range = np.zeros([2, num_tc]).astype(int)
2300.           TH_range = np.zeros([2, num_th]).astype(int)
2301.           # 标记流股温度在温区中的位置
2302.           for j in range(num_tc):
2303.               for i in range(num_tc_uni):
2304.                   if TC_in[j] == TC_uni[i]:
2305.                       TC_range[0, j] = i
2306.                   if TC_out[j] == TC_uni[i]:
2307.                       TC_range[1, j] = i
2308.           for j in range(num_th):
2309.               for i in range(num_th_uni):
2310.                   if TH_in[j] == TH_uni[i]:
2311.                       TH_range[0, j] = i
2312.                   if TH_out[j] == TH_uni[i]:
2313.                       TH_range[1, j] = i
2314.           # print(TH_range)
2315.           # 计算冷热流股在各温区的热负荷
2316.           for j in range(num_tc):
2317.               for i in range(TC_range[0, j], TC_range[1, j]):
2318.                   HeatLoad_C[i] = HeatLoad_C[i] - DTC[i] * CPC[j]
2319.
2320.           for j in range(num_th):
2321.               for i in range(TH_range[1, j], TH_range[0, j]):
2322.                   HeatLoad_H[i] = HeatLoad_H[i] - DTH[i] * CPH[j]
```

```
        # print(HeatLoad_C)
        ############计算总温区温差和负荷
        T_u = []
        for a in TC_uni:
            T_u.append(a + 0.5 * DT_min)
        for a in TH_uni:
            T_u.append(a - 0.5 * DT_min)
        T_uni = np.unique(T_u)
        # print(T_uni)
        num_t_uni = len(T_uni)
        num_tc_im = len(TC_in_im)
        num_th_im = len(TH_in_im)
        T_range_tc = np.zeros([2, num_tc_im]).astype(int)
        T_range_th = np.zeros([2, num_th_im]).astype(int)
        # 流股标号
        for j in range(num_th_im):
            for i in range(num_t_uni):
                if T_uni[i] == TH_in_im[j]:
                    T_range_th[0, j] = i
                if T_uni[i] == TH_out_im[j]:
                    T_range_th[1, j] = i
        # print(T_range_th)

        for j in range(num_tc_im):
            for i in range(num_t_uni):
                if T_uni[i] == TC_in_im[j]:
```

```
                        T_range_tc[0, j] = i
                    if T_uni[i] == TC_out_im[j]:
                        T_range_tc[1, j] = i
            # 计算温差
            num_range = num_t_uni - 1
            DLoad = np.zeros(num_range)
            DT = np.zeros(num_range)
            for i in range(num_range):
                DT[i] = T_uni[i + 1] - T_uni[i]
            # print(DT)
            for j in range(num_tc_im):
                for i in range(T_range_tc[0, j], T_range_tc[1, j]):
                    DLoad[i] = DLoad[i] + DT[i] * CPC[j]
            for j in range(num_th_im):
                for i in range(T_range_th[1, j], T_range_th[0, j]):
                    DLoad[i] = DLoad[i] - DT[i] * CPH[j]
            # print(DLoad)
            DLoad_Inv = DLoad[::-1]
            # print(DLoad_Inv)
            calculate = np.zeros([2, num_range + 1])
            for i in range(num_range):
                calculate[1, i] = calculate[0, i] - DLoad_Inv[i]
                calculate[0, i + 1] = calculate[1, i]
            # print(calculate)
            Dload_total = []
            for a in calculate[1, :]:
```

```
                Dload_total.append(a)
            Dload_total.pop()
            Dload_total = np.array(Dload_total)
            # print(Dload_total)
            temp1 = 0
            temp2 = 0
            for i in range(num_range):
                if Dload_total[i] >= 0:
                    temp1 = temp1 + 1
                if Dload_total[i] <= 0 and Dload_total[i] >= Dload_total[num_range - 1]:
                    temp2 = temp2 - 1
            # print(temp2, temp1)
            Load = []
            for i in [1, 2, 3, 5, 7, 8, 9, 10, 11]:
                temp = (TH_in[i] - TH_out[i]) * CPH[i]
                Load.append(temp)
            for j in [1, 2, 5, 6]:
                temp = (0 - TC_in[j] + TC_out[j]) * CPC[j]
                Load.append(temp)
            if temp2 == -num_range:
                HC0 = 0
                HH0 = 0
                QH = sum(HeatLoad_C) - sum(HeatLoad_H)
                Q_all = sum(Load) - QH
                CO2 = Q_all*0.3021
                self.pinch_print.setHtml('<font color="red">kg/h.</font>')
```

```
2401.                 self.pinch_print.insertPlainText('阈值问题，只需要加热公用工程，加热公用工程用量为：')
2402.                 self.pinch_print.insertPlainText(str(int(round(QH,0))))
2403.                 self.pinch_print.insertPlainText('kW，')
2404.                 self.pinch_print.insertHtml('<font color="red">节能潜力为(仅针对内置数据)：</font>')
2405.                 self.pinch_print.insertPlainText(str(int(round(Q_all,0))))
2406.                 self.pinch_print.insertPlainText('kW，')
2407.                 self.pinch_print.insertPlainText('预计减少碳排放(仅针对内置数据)：')
2408.                 self.pinch_print.insertPlainText(str(round(CO2,2)))
2409.                 self.pinch_print.setReadOnly(True)
2410.             if temp1 == num_range:
2411.                 HH0 = 0
2412.                 HC0 = sum(HeatLoad_H) - sum(HeatLoad_C)
2413.                 QC = HC0
2414.                 Q_all = sum(Load) - QC
2415.                 CO2 = Q_all*0.3021
2416.                 self.pinch_print.setHtml('<font color="red">kg/h.</font>')
2417.                 self.pinch_print.insertPlainText('阈值问题，只需要冷却公用工程，冷却公用工程用量为：')
2418.                 self.pinch_print.insertPlainText(str(int(round(QC,0))))
2419.                 self.pinch_print.insertPlainText('kW，')
2420.                 self.pinch_print.insertHtml('<font color="red">节能潜力为(仅针对内置数据)：</font>')
2421.                 self.pinch_print.insertPlainText(str(int(round(Q_all,0))))
2422.                 self.pinch_print.insertPlainText('kW，')
2423.                 self.pinch_print.insertPlainText('预计减少碳排放(仅针对内置数据)：')
2424.                 self.pinch_print.insertPlainText(str(round(CO2,2)))
2425.                 self.pinch_print.setReadOnly(True)
2426.             if temp1 != num_range and temp2 != -num_range:
```

```
2427.                 max_index = Dload_total.argmin(0)
2428.                 self.T_pinch = T_pinch = T_uni[num_range - max_index - 1]
2429.                 HH0 = 0
2430.                 HC0 = sum(HeatLoad_H) - sum(HeatLoad_C) - Dload_total[max_index]
2431.             draw_HC = np.zeros(num_tc_uni)
2432.             draw_HC[0] = HC0
2433.             for i in range(1, num_tc_uni):
2434.                 draw_HC[i] = draw_HC[i - 1] + HeatLoad_C[i - 1]
2435.             # print(draw_HC)
2436.             draw_HH = np.zeros(num_th_uni)
2437.             draw_HH[0] = HH0
2438.             for i in range(1, num_th_uni):
2439.                 draw_HH[i] = draw_HH[i - 1] + HeatLoad_H[i - 1]
2440.             if temp1 != num_range and temp2 != -num_range:
2441.                 QC = HC0
2442.                 QH = draw_HC[num_tc_uni - 1] - draw_HH[num_th_uni - 1]
2443.                 Q_all = sum(Load) - QC - QH
2444.                 CO2 = 0.3021* Q_all
2445.                 self.pinch_print.setHtml('<font color="blue">℃。</font>')
2446.                 self.pinch_print.insertHtml('<font color="blue">夹点问题，冷却公用工程用量为：</font>')
2447.                 self.pinch_print.insertPlainText(str(int(round(QC,0))))
2448.                 self.pinch_print.insertPlainText('kW，')
2449.                 self.pinch_print.insertPlainText('加热公用工程用量为：')
2450.                 self.pinch_print.insertPlainText(str(int(round(QH,0))))
2451.                 self.pinch_print.insertPlainText('kW，')
2452.                 self.pinch_print.insertHtml('<font color="red">节能潜力为(仅针对内置数据)：</font>')
```

```
2453.                 self.pinch_print.insertPlainText(str(int(round(Q_all,0))))
2454.                 self.pinch_print.insertPlainText('kW，')
2455.                 self.pinch_print.insertPlainText('预计减少碳排放(仅针对内置数据)：')
2456.                 self.pinch_print.insertPlainText(str(round(CO2,2)))
2457.                 self.pinch_print.insertPlainText('kg/h，')
2458.                 self.pinch_print.insertHtml('<font color="blue">夹点温度为：</font>')
2459.                 self.pinch_print.insertPlainText(str(round(T_pinch,2)))
2460.                 self.pinch_print.insertPlainText('℃，夹点温差为：')
2461.                 self.pinch_print.insertPlainText(str(DT_min))
2462.                 self.pinch_print.setReadOnly(True)
2463.             # # 绘图
2464.             # plt.plot(draw_HC, TC_uni)
2465.             # plt.plot(draw_HH, TH_uni)
2466.             # plt.show()
2467.             judge = 0
2468.             return draw_HC, TC_uni, draw_HH, TH_uni, judge
2469.         def Pinch_compute(self):
2470.             TC_in = []
2471.             TC_out = []
2472.             CPC = []
2473.             TH_in = []
2474.             TH_out = []
2475.             CPH = []
2476.             # print(self.table_widget.rowCount())
2477.             for i in [3, 4, 6]:
2478.                 for j in range(self.table_widget.rowCount()):
```

```
2479.                     in_t = float(self.table_widget.item(j, 3).text())
2480.                     out_t = float(self.table_widget.item(j, 4).text())
2481.                     if in_t>=out_t and i == 3:
2482.                         TH_in.append(self.table_widget.item(j, i).text())
2483.                         # print(TH_in)
2484.
2485.                     if in_t<=out_t and i == 3:
2486.                         TC_in.append(self.table_widget.item(j, i).text())
2487.                     if in_t>=out_t and i == 4:
2488.                         TH_out.append(self.table_widget.item(j, i).text())
2489.                     if in_t<=out_t and i == 4:
2490.                         TC_out.append(self.table_widget.item(j, i).text())
2491.                     if in_t>=out_t and i == 6:
2492.                         CPH.append(self.table_widget.item(j, i).text())
2493.                     if in_t<=out_t and i == 6:
2494.                         CPC.append(self.table_widget.item(j, i).text())
2495.             # TC_in = [43, 40, 73, 74.3, 123, 109.2, 105]
2496.             # TC_out = [208, 61, 73.8, 89, 126, 109.3, 105.1]
2497.             # CPC = [241.54, 63.81, 20300, 81.9, 13090, 359810, 10680]
2498.             # TH_in = [242, 102, 72, 57, 125, 113, 117.4, 117.3, 68, 51, 69, 109]
2499.             # TH_out = [102, 30, 57, 54, 102, 44, 117.3, 112.6, 53, 31, 49, 40]
2500.             # CPH = [284.67, 25.64, 983.07, 28.33, 52.35, 33.01, 359810, 144.47, 2529, 27.65, 56.3, 9.75]
2501.             DT_min = np.array([self.DTmin]).astype(float)[0]
2502.             # print(DT_min)
2503.             # TC_in=[20,80]
2504.             # TC_out=[135,140]
```

```
2505.           # TH_in=[170,150]
2506.           # TH_out=[60,30]
2507.           # DT_min=10
2508.           # CPC=[2,4]
2509.           # CPH=[3,1.5]
2510.           # 创建虚拟温度
2511.           self.TC_in_im = TC_in_im = np.array(TC_in).astype(float) + 0.5 * DT_min
2512.           self.TC_out_im = TC_out_im = np.array(TC_out).astype(float) + 0.5 * DT_min
2513.           self.TH_in_im = TH_in_im = np.array(TH_in).astype(float) - 0.5 * DT_min
2514.           self.TH_out_im = TH_out_im = np.array(TH_out).astype(float) - 0.5 * DT_min
2515.           # print(1)
2516.           CPC = np.array(CPC).astype(float)
2517.           CPH = np.array(CPH).astype(float)
2518.           # 对温度去重并排序,计算温差
2519.           TC_in = np.array(TC_in).astype(float)
2520.           TC_out = np.array(TC_out).astype(float)
2521.           TH_in = np.array(TH_in).astype(float)
2522.           TH_out = np.array(TH_out).astype(float)
2523.
2524.           TC_uni = np.unique([TC_in, TC_out])
2525.           TH_uni = np.unique([TH_in, TH_out])
2526.           # print(CPC)
2527.
2528.
2529.           num_tc_uni = len(TC_uni)
2530.           num_th_uni = len(TH_uni)
```

```
2531.           DTC = np.zeros([num_tc_uni]).astype(float)
2532.           DTH = np.zeros([num_th_uni]).astype(float)
2533.
2534.           for i in range(num_tc_uni):
2535.               DTC[i - 1] = TC_uni[i - 1] - TC_uni[i]
2536.           for i in range(num_th_uni):
2537.               DTH[i - 1] = TH_uni[i - 1] - TH_uni[i]
2538.
2539.           ############计算各个温区的热负荷
2540.           HeatLoad_C = np.zeros(num_tc_uni - 1)
2541.           HeatLoad_H = np.zeros(num_th_uni - 1)
2542.           num_tc = len(TC_in)
2543.           num_th = len(TH_in)
2544.           TC_range = np.zeros([2, num_tc]).astype(int)
2545.           TH_range = np.zeros([2, num_th]).astype(int)
2546.           # 标记流股温度在温区中的位置
2547.           for j in range(num_tc):
2548.               for i in range(num_tc_uni):
2549.                   if TC_in[j] == TC_uni[i]:
2550.                       TC_range[0, j] = i
2551.                   if TC_out[j] == TC_uni[i]:
2552.                       TC_range[1, j] = i
2553.           for j in range(num_th):
2554.               for i in range(num_th_uni):
2555.                   if TH_in[j] == TH_uni[i]:
2556.                       TH_range[0, j] = i
```

```
                    if TH_out[j] == TH_uni[i]:
                        TH_range[1, j] = i
            # print(TH_range)
            # 计算冷热流股在各温区的热负荷
            for j in range(num_tc):
                for i in range(TC_range[0, j], TC_range[1, j]):
                    HeatLoad_C[i] = HeatLoad_C[i] - DTC[i] * CPC[j]

            for j in range(num_th):
                for i in range(TH_range[1, j], TH_range[0, j]):
                    HeatLoad_H[i] = HeatLoad_H[i] - DTH[i] * CPH[j]
            # print(HeatLoad_C)
            ############计算总温区温差和负荷
            T_u = []
            for a in TC_uni:
                T_u.append(a + 0.5 * DT_min)
            for a in TH_uni:
                T_u.append(a - 0.5 * DT_min)
            T_uni = np.unique(T_u)
            # print(T_uni)
            num_t_uni = len(T_uni)
            num_tc_im = len(TC_in_im)
            num_th_im = len(TH_in_im)
            T_range_tc = np.zeros([2, num_tc_im]).astype(int)
            T_range_th = np.zeros([2, num_th_im]).astype(int)
            # 流股标号
```

```
2583.             for j in range(num_th_im):
2584.                 for i in range(num_t_uni):
2585.                     if T_uni[i] == TH_in_im[j]:
2586.                         T_range_th[0, j] = i
2587.                     if T_uni[i] == TH_out_im[j]:
2588.                         T_range_th[1, j] = i
2589.             # print(T_range_th)
2590.
2591.             for j in range(num_tc_im):
2592.                 for i in range(num_t_uni):
2593.                     if T_uni[i] == TC_in_im[j]:
2594.                         T_range_tc[0, j] = i
2595.                     if T_uni[i] == TC_out_im[j]:
2596.                         T_range_tc[1, j] = i
2597.             # 计算温差
2598.             num_range = num_t_uni - 1
2599.             DLoad = np.zeros(num_range)
2600.             DT = np.zeros(num_range)
2601.             for i in range(num_range):
2602.                 DT[i] = T_uni[i + 1] - T_uni[i]
2603.             # print(DT)
2604.             for j in range(num_tc_im):
2605.                 for i in range(T_range_tc[0, j], T_range_tc[1, j]):
2606.                     DLoad[i] = DLoad[i] + DT[i] * CPC[j]
2607.             for j in range(num_th_im):
2608.                 for i in range(T_range_th[1, j], T_range_th[0, j]):
```

```
                    DLoad[i] = DLoad[i] - DT[i] * CPH[j]
            # print(DLoad)
            DLoad_Inv = DLoad[::-1]
            # print(DLoad_Inv)
            calculate = np.zeros([2, num_range + 1])
            for i in range(num_range):
                calculate[1, i] = calculate[0, i] - DLoad_Inv[i]
                calculate[0, i + 1] = calculate[1, i]
            # print(calculate)
            Dload_total = []
            for a in calculate[1, :]:
                Dload_total.append(a)
            Dload_total.pop()
            Dload_total = np.array(Dload_total)
            # print(Dload_total)
            temp1 = 0
            temp2 = 0
            for i in range(num_range):
                if Dload_total[i] >= 0:
                    temp1 = temp1 + 1
                if Dload_total[i] <= 0 and Dload_total[i] >= Dload_total[num_range - 1]:
                    temp2 = temp2 - 1
            # print(temp2, temp1)

            if temp2 == -num_range:
                HC0 = 0
```

```
2635.                 HH0 = 0
2636.                 QH = sum(HeatLoad_C) - sum(HeatLoad_H)
2637.                 Q_all = QH
2638.
2639.             if temp1 == num_range:
2640.                 HH0 = 0
2641.                 HC0 = sum(HeatLoad_H) - sum(HeatLoad_C)
2642.                 QC = HC0
2643.                 Q_all =  QC
2644.
2645.             if temp1 != num_range and temp2 != -num_range:
2646.                 max_index = Dload_total.argmin(0)
2647.                 self.T_pinch = T_pinch = T_uni[num_range - max_index - 1]
2648.                 HH0 = 0
2649.                 HC0 = sum(HeatLoad_H) - sum(HeatLoad_C) - Dload_total[max_index]
2650.             draw_HC = np.zeros(num_tc_uni)
2651.             draw_HC[0] = HC0
2652.             for i in range(1, num_tc_uni):
2653.                 draw_HC[i] = draw_HC[i - 1] + HeatLoad_C[i - 1]
2654.             # print(draw_HC)
2655.             draw_HH = np.zeros(num_th_uni)
2656.             draw_HH[0] = HH0
2657.             for i in range(1, num_th_uni):
2658.                 draw_HH[i] = draw_HH[i - 1] + HeatLoad_H[i - 1]
2659.             if temp1 != num_range and temp2 != -num_range:
2660.                 QC = HC0
```

```
2661.                 QH = draw_HC[num_tc_uni - 1] - draw_HH[num_th_uni - 1]
2662.                 Q_all = QC + QH
2663.             Load=[]
2664.             for i in [1, 2, 3, 5, 7, 8, 9, 10, 11]:
2665.                 temp = (TH_in[i] - TH_out[i]) * CPH[i]
2666.                 Load.append(temp)
2667.             for j in [1, 2, 5, 6]:
2668.                 temp = (0 - TC_in[j] + TC_out[j]) * CPC[j]
2669.                 Load.append(temp)
2670.             # CO2=0.3021*(sum(Load)-Q_all)
2671.             # # 绘图
2672.             # plt.plot(draw_HC, TC_uni)
2673.             # plt.plot(draw_HH, TH_uni)
2674.             # plt.show()
2675.
2676.             return Q_all,sum(Load)
2677.
2678.         def slot_plot(self):
2679.             [x1,y1,x2,y2,judge] = self.Pinch_Draw()
2680.             self.fig.axes.plot(x1, y1, 'b')
2681.             self.fig.axes.plot(x2, y2, 'r')
2682.             self.fig.axes.set_title('Cold and hot composite curve under current data')
2683.             self.fig.axes.set_ylabel('Temperature/℃')
2684.             self.fig.axes.set_xlabel('Enthalpy/kW')
2685.             self.fig.axes.grid()
2686.             self.fig.resize(450,360)
```

```
2687.             self.fig.draw()
2688.
2689.         def slot_plot2(self):
2690.             x1=[0,1204]
2691.             y11=[102,125]
2692.             y12=[74.3,89]
2693.             x2=[3204,41185]
2694.             y21=[117.3,117.4]
2695.             y22=[123,126]
2696.             x3=[41185,83039]
2697.             y31=[43,208]
2698.             y32=[102,242]
2699.             self.fig2.axes.clear()
2700.             self.fig2.axes.plot(x1, y11, 'r')
2701.             self.fig2.axes.plot(x1, y12, 'b')
2702.             self.fig2.axes.plot(x2, y21, 'b')
2703.             self.fig2.axes.plot(x2, y22, 'r')
2704.             self.fig2.axes.plot(x3, y31, 'b')
2705.             self.fig2.axes.plot(x3, y32, 'r')
2706.             temp_x1 = x3[1]
2707.             for i in [1, 2, 3, 5, 7, 8, 9, 10, 11, 13, 14, 17, 18]:
2708.                 temp_x1 = temp_x1 + 2000
2709.                 temp_x2 = float(self.table_widget.item(i,7).text()) + temp_x1
2710.                 x=[temp_x1,temp_x2]
2711.                 if i>11:
2712.                     y=[float(self.table_widget.item(i,3).text()),float(self.table_widget.item(i,4).text())]
```

```
2713.                 self.fig2.axes.plot(x, y, 'b')
2714.             else:
2715.                 y = [float(self.table_widget.item(i, 4).text()), float(self.table_widget.item(i, 3).text())]
2716.                 self.fig2.axes.plot(x, y, 'r')
2717.             temp_x1 = temp_x2
2718.
2719.         self.fig2.axes.set_title('T-H figure for original match')
2720.         self.fig2.axes.set_ylabel('Temperature/℃')
2721.         self.fig2.axes.set_xlabel('Enthalpy/kW')
2722.         self.fig2.axes.grid()
2723.         self.fig2.resize(450,380)
2724.         self.fig2.draw()
2725.
2726.     def slot_plot3(self):
2727.         [x2,y2,x1,y1,judge] = self.Pinch_Draw()
2728.         y1 = y1 - 0.5 * float(self.DTmin)
2729.         y2 = y2 + 0.5 * float(self.DTmin)
2730.         TH_max = y1[len(y1) - 1]
2731.         TH_min = y1[0]
2732.         TC_max = y2[len(y2) - 1]
2733.         TC_min = y2[0]
2734.         T = []
2735.         for i in y1:
2736.             T.append(i)
2737.         for j in y2:
2738.             T.append(j)
```

```
            T.sort()
            y1 = list(y1)
            y2 = list(y2)
            x1 = list(x1)
            x2 = list(x2)
            x = [0 for _ in range(len(T))]
            for k in range(len(T)):
                if T[k] < TC_min :
                    idx = y1.index(T[k])
                    x[k] = x2[0] - x1[idx]
                elif T[k] > TC_max :
                    idx = y1.index(T[k])
                    x[k] = x2[len(x2)-1] - x1[idx]
                else:
                    if T[k] in y1:
                        for t in range(len(y2)):
                            if T[k] < y2[t]:
                                xc = x2[t] + (x2[t-1] - x2[t])*(T[k] - y2[t])/(y2[t-1] - y2[t])
                                idx = y1.index(T[k])
                                x[k] = xc - x1[idx]
                                break
                    if T[k] in y2:
                        for t in range(len(y1)):
                            if T[k] < y1[t]:
                                xh = x1[t] + (x1[t-1] - x1[t])*(T[k] - y1[t])/(y1[t-1] - y1[t])
                                idx = y2.index(T[k])
```

```
                                x[k] = x2[idx] - xh
                                break
            self.fig3.axes.clear()
            for p in range(len(T)-1):
                self.fig3.axes.plot([x[p],x[p+1]], [T[p],T[p+1]], 'y')
            self.fig3.axes.set_title('Grand composite curve under current data')
            self.fig3.axes.set_ylabel('Temperature/℃')
            self.fig3.axes.set_xlabel('Enthalpy/kW')
            self.fig3.axes.grid()
            self.fig3.resize(450,340)
            self.fig3.draw()

        def event_pinch_click(self):
            if self.txt_pinch.text():
                self.DTmin = self.txt_pinch.text()
                [x1,y1,x2,y2,judge] = self.Pinch_Draw()
                self.fig.axes.clear()
                self.fig.axes.plot(x1, y1, 'b')
                self.fig.axes.plot(x2, y2, 'r')
                self.fig.axes.set_title('Cold and hot composite curve under current data')
                self.fig.axes.set_ylabel('Temperature/℃')
                self.fig.axes.set_xlabel('Enthalpy/kW')
                self.fig.axes.grid()
                self.fig.draw()
                self.slot_plot3()

```

```
2791.       def event_statistic_recover(self):
2792.           import json
2793.           file_path = os.path.join(BASE_DIR, "statistics", "origion_statistics.json")
2794.           with open(file_path, mode='r', encoding='utf-8') as f:
2795.               data = f.read()
2796.           data_list = json.loads(data)
2797.           row_count = self.table_widget.rowCount()
2798.           while 1:
2799.               self.table_widget.removeRow(0)
2800.               row_count = row_count - 1
2801.               if row_count < 0:
2802.                   break
2803.           current_row_count = self.table_widget.rowCount()
2804.           for row_list in data_list:
2805.               self.table_widget.insertRow(current_row_count)
2806.               for i, statistic in enumerate(row_list):
2807.                   cell = QTableWidgetItem(str(statistic))
2808.                   self.table_widget.setItem(current_row_count, i, cell)
2809.               current_row_count += 1
2810.           self.changed_index = []
2811.
2812.       def event_statistic_delete(self):
2813.           row_count = self.table_widget.rowCount()
2814.           while 1:
2815.               self.table_widget.removeRow(0)
2816.               row_count = row_count - 1
```

```
2817.                 if row_count < 0:
2818.                     break
2819.
2820.         def event_find_nonsense(self):
2821.             self.nonsense_print.clear()
2822.             across_pinch = []
2823.             above_pinch = []
2824.             below_pinch = []
2825.             HE_index_H = []
2826.             HE_index_C = []
2827.             E0 = ['E01', 'E02', 'E03']
2828.             EH = ['EH1', 'EH2', 'EH3', 'EH4']
2829.             EC = ['EC1', 'EC2', 'EC3', 'EC4', 'EC5', 'EC6', 'EC7', 'EC8', 'EC9']
2830.             for j in range(self.table_widget.rowCount()):
2831.                 if j <= 11:
2832.                     HE_index_C.append(self.table_widget.item(j, 2).text())
2833.                     # print(TH_in)
2834.                 if j > 11:
2835.                     HE_index_H.append(self.table_widget.item(j, 2).text())
2836.
2837.             for i in range(len(self.TC_in_im) - 1):
2838.                 if self.TC_in_im[i] < self.T_pinch and self.TC_out_im[i] > self.T_pinch and HE_index_H[i] in E0:
2839.                     across_pinch.append(HE_index_H[i])
2840.                 if self.TC_out_im[i] < self.T_pinch and HE_index_H[i] in EH:
2841.                     below_pinch.append(HE_index_H[i])
2842.
```

```
            for i in range(len(self.TH_in_im) - 1):
                if self.TH_in_im[i] > self.T_pinch and self.TH_out_im[i] < self.T_pinch and HE_index_C[i] in E0:
                    across_pinch.append(HE_index_C[i])
                if self.TH_in_im[i] > self.T_pinch and HE_index_C[i] in EC:
                    above_pinch.append(HE_index_C[i])


            across_pinch = list(set(across_pinch))
            self.nonsense_print.append('1.跨越夹点换热的换热器有：')
            if len(across_pinch):
                self.nonsense_print.append(across_pinch[0])
                across_pinch.pop(0)
                for i in across_pinch:
                    self.nonsense_print.insertPlainText('  ' + i)
            else:
                self.nonsense_print.append('无跨越夹点的换热器')

            self.nonsense_print.append('2.夹点之上的冷却器有：')
            if len(above_pinch):
                self.nonsense_print.append(above_pinch[0])
                above_pinch.pop(0)
                for i in above_pinch:
                    self.nonsense_print.insertPlainText('  ' + i)
            else:
                self.nonsense_print.append('无夹点之上的冷却器')

```

```
2869.             self.nonsense_print.append('3.夹点之下的加热器有：')
2870.             if below_pinch:
2871.                 self.nonsense_print.append(below_pinch[0])
2872.                 below_pinch.pop(0)
2873.                 for i in below_pinch:
2874.                     self.nonsense_print.insertPlainText('   ' + i)
2875.             else:
2876.                 self.nonsense_print.append('无夹点之下的加热器')
2877.             self.nonsense_print.setStyleSheet('font: 18px')
2878.             self.nonsense_print.setReadOnly(True)
2879. 
2880.         def event_cellchanged(self):
2881.             if self.forbidden_color:
2882.                 changed_background = QLabel()
2883.                 if self.color_signal == 1:
2884.                     changed_row_index = self.table_widget.currentRow()
2885.                     changed_column_index = self.table_widget.currentColumn()
2886.                     changed_background.setStyleSheet("background-color: rgba(255, 0, 0, 0.2)")
2887.                     self.table_widget.setCellWidget(changed_row_index, changed_column_index, changed_background)
2888.                 if self.color_signal == 0:
2889.                     changed_background.setStyleSheet("background-color: rgba(0, 255, 0, 0.2)")
2890.                     self.table_widget.setCellWidget(self.load_row_index, 7, changed_background)
2891.                     self.color_signal = 1
2892.                     self.change_signal = 1
2893.                 if self.color_signal == 2:
2894.                     changed_background.setStyleSheet("background-color: rgba(0, 255, 0, 0.2)")
```

```
2895.                     self.table_widget.setCellWidget(self.load_row_index, self.T_index, changed_background)
2896.                     self.color_signal = 1
2897.                     self.change_signal = 1
2898.
2899.
2900.                 if self.change_signal:
2901.                     self.change_signal = 0
2902.                 else:
2903.                     if changed_column_index in [3, 4, 6]:
2904.                         D_data = float(self.table_widget.item(changed_row_index, changed_column_index).text()) -
   float(
2905.                             self.data_list[changed_row_index][changed_column_index])
2906.                         self.changed_index.append([changed_row_index, changed_column_index, D_data])
2907.                     if [changed_row_index, changed_column_index] == [0, 4]:
2908.                         self.change_signal = 1
2909.                         changed_background = QLabel()
2910.                         changed_background.setStyleSheet("background-color: rgba(255, 0, 0, 0.2)")
2911.                         self.table_widget.setCellWidget(1, 3, changed_background)
2912.                         self.table_widget.item(1, 3).setText(self.table_widget.item(0, 4).text())
2913.                     if [changed_row_index, changed_column_index] == [1, 3]:
2914.                         self.change_signal = 1
2915.                         changed_background = QLabel()
2916.                         changed_background.setStyleSheet("background-color: rgba(255, 0, 0, 0.2)")
2917.                         self.table_widget.setCellWidget(0, 4, changed_background)
2918.                         self.table_widget.item(0, 4).setText(self.table_widget.item(1, 3).text())
2919.                     if [changed_row_index, changed_column_index] == [2, 4]:
```

```
2920.                         self.change_signal = 1
2921.                         changed_background = QLabel()
2922.                         changed_background.setStyleSheet("background-color: rgba(255, 0, 0, 0.2)")
2923.                         self.table_widget.setCellWidget(3, 3, changed_background)
2924.                         self.table_widget.item(3, 3).setText(self.table_widget.item(2, 4).text())
2925.                     if [changed_row_index, changed_column_index] == [3, 3]:
2926.                         self.change_signal = 1
2927.                         changed_background = QLabel()
2928.                         changed_background.setStyleSheet("background-color: rgba(255, 0, 0, 0.2)")
2929.                         self.table_widget.setCellWidget(2, 4, changed_background)
2930.                         self.table_widget.item(2, 4).setText(self.table_widget.item(3, 3).text())
2931.                     if [changed_row_index, changed_column_index] == [6, 4]:
2932.                         self.change_signal = 1
2933.                         changed_background = QLabel()
2934.                         changed_background.setStyleSheet("background-color: rgba(255, 0, 0, 0.2)")
2935.                         self.table_widget.setCellWidget(7, 3, changed_background)
2936.                         self.table_widget.item(7, 3).setText(self.table_widget.item(6, 4).text())
2937.                     if [changed_row_index, changed_column_index] == [7, 3]:
2938.                         self.change_signal = 1
2939.                         changed_background = QLabel()
2940.                         changed_background.setStyleSheet("background-color: rgba(255, 0, 0, 0.2)")
2941.                         self.table_widget.setCellWidget(6, 4, changed_background)
2942.                         self.table_widget.item(6, 4).setText(self.table_widget.item(7, 3).text())
2943.                     if [changed_row_index, changed_column_index] == [7, 4]:
2944.                         self.change_signal = 1
2945.                         changed_background = QLabel()
```

```
                    changed_background.setStyleSheet("background-color: rgba(255, 0, 0, 0.2)")
                    self.table_widget.setCellWidget(5, 3, changed_background)
                    self.table_widget.item(5, 3).setText(self.table_widget.item(7, 4).text())
                if [changed_row_index, changed_column_index] == [5, 3]:
                    self.change_signal = 1
                    changed_background = QLabel()
                    changed_background.setStyleSheet("background-color: rgba(255, 0, 0, 0.2)")
                    self.table_widget.setCellWidget(7, 4, changed_background)
                    self.table_widget.item(7, 4).setText(self.table_widget.item(5, 3).text())


    def event_load_shift_search(self):
        self.load_print.clear()
        road_list = [
            {'idx': [0, 3], 'road': ['E03', '→', 'EC1'], 'next_idx': [1, 3]},
            {'idx': [0, 4], 'road': ['EC1'], 'next_idx': []},
            {'idx': [0, 6], 'road': ['E03', '→', 'EC1'], 'next_idx': [1, 6]},
            {'idx': [1, 3], 'road': ['EC1'], 'next_idx': []},
            {'idx': [1, 4], 'road': ['不影响换热器负荷'], 'next_idx': []},
            {'idx': [1, 6], 'road': ['EC1'], 'next_idx': []},
            {'idx': [2, 3], 'road': ['EC2', '→', 'EC3'], 'next_idx': [3, 3]},
            {'idx': [2, 4], 'road': ['EC3'], 'next_idx': []},
            {'idx': [2, 6], 'road': ['EC2', '→', 'EC3'], 'next_idx': [3, 6]},
            {'idx': [3, 3], 'road': ['EC3'], 'next_idx': []},
            {'idx': [3, 4], 'road': ['不影响换热器负荷'], 'next_idx': []},
```

```
                {'idx': [3, 6], 'road': ['EC3'], 'next_idx': []},
                {'idx': [4, 3], 'road': ['E01'], 'next_idx': []},
                {'idx': [4, 4], 'road': ['不影响换热器负荷'], 'next_idx': []},
                {'idx': [4, 6], 'road': ['E01']},
                {'idx': [5, 3], 'road': ['EC4']},
                {'idx': [5, 4], 'road': ['不影响换热器负荷']},
                {'idx': [5, 6], 'road': ['EC4']},
                {'idx': [6, 3], 'road': ['EC5', '→', 'EC4']},
                {'idx': [6, 4], 'road': ['EC4']},
                {'idx': [6, 6], 'road': ['EC5', '→', 'EC4']},
                {'idx': [7, 3], 'road': ['E02', '→', 'EC5', '→', 'EC4']},
                {'idx': [7, 4], 'road': ['EC5', '→', 'EC4']},
                {'idx': [7, 6], 'road': ['E02', '→', 'EC5', '→', 'EC4']},
                {'idx': [8, 3], 'road': ['EC6']},
                {'idx': [8, 4], 'road': ['不影响换热器负荷']},
                {'idx': [8, 6], 'road': ['EC6']},
                {'idx': [9, 3], 'road': ['EC7']},
                {'idx': [9, 4], 'road': ['不影响换热器负荷']},
                {'idx': [9, 6], 'road': ['EC7']},
                {'idx': [10, 3], 'road': ['EC8']},
                {'idx': [10, 4], 'road': ['不影响换热器负荷']},
                {'idx': [10, 6], 'road': ['EC8']},
                {'idx': [11, 3], 'road': ['EC9']},
                {'idx': [11, 4], 'road': ['不影响换热器负荷']},
                {'idx': [11, 6], 'road': ['EC9']},
                {'idx': [12, 3], 'road': ['E03', '→', 'EC1']},
```

```
2998.                 {'idx': [12, 4], 'road': ['EC1']},
2999.                 {'idx': [12, 6], 'road': ['E03', '→', 'EC1']},
3000.                 {'idx': [13, 3], 'road': ['EH1']},
3001.                 {'idx': [13, 4], 'road': ['不影响换热器负荷']},
3002.                 {'idx': [13, 6], 'road': ['EH1']},
3003.                 {'idx': [14, 3], 'road': ['EH2']},
3004.                 {'idx': [14, 4], 'road': ['不影响换热器负荷']},
3005.                 {'idx': [14, 6], 'road': ['E01']},
3006.                 {'idx': [15, 3], 'road': ['E01']},
3007.                 {'idx': [15, 4], 'road': ['不影响换热器负荷']},
3008.                 {'idx': [15, 6], 'road': ['E01']},
3009.                 {'idx': [16, 3], 'road': ['E02', '→', 'EC5', '→', 'EC4']},
3010.                 {'idx': [16, 4], 'road': ['不影响换热器负荷']},
3011.                 {'idx': [16, 6], 'road': ['E02', '→', 'EC5', '→', 'EC4']},
3012.                 {'idx': [17, 3], 'road': ['EH3']},
3013.                 {'idx': [17, 4], 'road': ['不影响换热器负荷']},
3014.                 {'idx': [17, 6], 'road': ['EH3']},
3015.                 {'idx': [18, 3], 'road': ['EH4']},
3016.                 {'idx': [18, 4], 'road': ['不影响换热器负荷']},
3017.                 {'idx': [18, 6], 'road': ['EH4']},
3018.             ]
3019.
3020.             Exchanger = []
3021.             for j in range(self.table_widget.rowCount()):
3022.                 Exchanger.append(str(self.table_widget.item(j, 2).text()))
3023.             # print(len(Exchanger))
```

```
3024.
3025.
3026.           #输出负荷变化路径
3027.           for i in self.changed_index:
3028.               for m in road_list:
3029.                   if i[0] == m['idx'][0] and i[1] == m['idx'][1]:
3030.                       if self.load_shift_chose.currentIndex() == 1:
3031.                           if (len(m['road']) == 1 and m['road'][0] != '不影响换热器负荷') or m['idx'][0] > 11:
3032.                               load_shift_nonsense_text = '现行条件无法满足换热器负荷不变'
3033.                           if i[1] != 6:
3034.                               Dload = i[2] * float(self.table_widget.item(i[0], 6).text())
3035.                           else:
3036.                               Dload = i[2] * abs(float(self.table_widget.item(i[0], 3).text()) - float(
3037.                                   self.table_widget.item(i[0], 4).text()))
3038.                           if len(m['road'])>=3:
3039.                               next_row_index = Exchanger.index(m['road'][2])
3040.                           else:
3041.                               next_row_index = Exchanger.index(m['road'][0])
3042.                           if i[1] == 6:
3043.                               if (next_row_index <= 11 and i[0] <= 11) or (next_row_index > 11 and i[0] > 11):
3044.                                   next_load = float(self.table_widget.item(next_row_index, 6).text()) * abs(
3045.                                       float(self.table_widget.item(next_row_index, 3).text()) - float(
3046.                                           self.table_widget.item(next_row_index, 4).text())) - Dload
3047.                                   next_T_in = float(self.table_widget.item(next_row_index, 3).text()) + (next_l
   oad - float(self.table_widget.item(next_row_index,7).text())) / float(self.table_widget.item(i[0],6).text())
3048.                           else:
```

```
                                    next_load = float(self.table_widget.item(next_row_index, 6).text()) * abs(
                                        float(self.table_widget.item(next_row_index, 3).text()) - float(
                                            self.table_widget.item(next_row_index, 4).text())) + Dload
                                    next_T_in = float(self.table_widget.item(next_row_index, 3).text()) - (next_l
    oad - float(self.table_widget.item(next_row_index,7).text())) / float(self.table_widget.item(i[0],6).text())
                                else:
                                    if (next_row_index <= 11 and i[0] <= 11) or (next_row_index > 11 and i[0] > 11):
                                        next_load = float(self.table_widget.item(next_row_index, 7).text()) - Dload
                                        next_T_in = float(self.table_widget.item(next_row_index, 3).text()) + Dload /
     float(self.table_widget.item(i[0],6).text())
                                    else:
                                        next_load = float(self.table_widget.item(next_row_index, 7).text()) + Dload
                                        next_T_in = float(self.table_widget.item(next_row_index, 3).text()) - Dload /
     float(self.table_widget.item(i[0],6).text())

                                if i[1] == 6 and len(m['road']) >= 5:
                                    nnext_row_index = Exchanger.index(m['road'][4])
                                    if nnext_row_index <= 11:
                                        nnext_load = float(self.table_widget.item(nnext_row_index, 6).text()) * abs(
                                            float(self.table_widget.item(nnext_row_index, 3).text()) - float(
                                                self.table_widget.item(nnext_row_index, 4).text()))
                                if i[1] == 3:
                                    print = '●' + '检测到' + self.table_widget.item(i[0], 0).text() + '进口温度发生变化，
    所有可能负荷转移路径为：'
                                    self.load_print.append(print)
                                    if 'load_shift_nonsense_text' in dir():
```

```
                                    self.load_print.insertPlainText(load_shift_nonsense_text)
                                else:
                                    for k in m['road']:
                                        self.load_print.insertPlainText(k)
                                        while (len(m['road']) >= 3):
                                            if k == m['road'][2] and 'next_load' in dir():
                                                self.color_signal = 0
                                                self.load_row_index = next_row_index
                                                next_load_changed_text = '(' + '负荷由' + self.table_widget.item(
                                                    self.load_row_index, 7).text() + 'kW' + '变为' + str(
                                                    next_load) + 'kW' + ')'
                                                self.table_widget.item(self.load_row_index, 7).setText(str(next_l
oad))
                                                self.color_signal = 2
                                                self.T_index = 4
                                                self.load_row_index = i[0]
                                                self.table_widget.item(i[0], 4).setText(str(next_T_in))
                                                self.color_signal = 2
                                                self.T_index = 3
                                                self.load_row_index = next_row_index
                                                self.table_widget.item(next_row_index, 3).setText(str(next_T_in))
                                            if k == m['road'][2] and 'next_load_changed_text' in dir():
                                                self.load_print.insertPlainText(next_load_changed_text)
                                            break
                            if i[1] == 4:
```

```
                                print = '●' + '检测到' + self.table_widget.item(i[0], 0).text() + '出口温度发生变化，
    所有可能负荷转移路径为：'
                                self.load_print.append(print)
                                if m['road'] != ['不影响换热器负荷']:
                                    if 'load_shift_nonsense_text' in dir():
                                        self.load_print.insertPlainText(load_shift_nonsense_text)
                                    else:
                                        for k in m['road']:
                                            self.load_print.insertPlainText(k)
                                            while (len(m['road']) >= 3):
                                                if k == m['road'][2] and 'next_load' in dir():
                                                    next_load_changed_text = '(' + '负荷由' + str(
                                                        self.table_widget.item(next_row_index,
                                                                               7).text()) + 'kW' + '变为' + str(
                                                        next_load) + 'kW' + ')'
                                                    self.color_signal = 0
                                                    self.load_row_index = next_row_index
                                                    self.table_widget.item(self.load_row_index, 7).setText(
                                                        str(next_load))
                                                    self.color_signal = 2
                                                    self.T_index = 4
                                                    self.load_row_index = i[0]
                                                    self.table_widget.item(i[0], 4).setText(str(next_T_in))
                                                    self.color_signal = 2
                                                    self.T_index = 3
                                                    self.load_row_index = next_row_index
```

```
                                                self.table_widget.item(next_row_index, 3).setText(str(next_T_
    in))
                                            if k == m['road'][2] and 'next_load_changed_text' in dir():
                                                self.load_print.insertPlainText(next_load_changed_text)
                                            break
                                    else:
                                        self.load_print.insertPlainText(m['road'][0])

                                if i[1] == 6:
                                    print = '●' + '检测到' + self.table_widget.item(i[0], 0).text() + '平均热容流率发生
    变化，所有可能负荷转移路径为：'
                                    self.load_print.append(print)
                                    if m['road'] != ['不影响换热器负荷']:
                                        if 'load_shift_nonsense_text' in dir():
                                            self.load_print.insertPlainText(load_shift_nonsense_text)
                                        else:
                                            for k in m['road']:
                                                self.load_print.insertPlainText(k)
                                                while (len(m['road']) >= 3):
                                                    if k == m['road'][2] and 'next_load' in dir():
                                                        next_load_changed_text = '(' + '负荷由
    ' + self.table_widget.item(
                                                            next_row_index, 7).text() + 'kW' + '变为' + str(
                                                            next_load) + 'kW' + ')'
                                                self.color_signal = 0
                                                self.load_row_index = next_row_index
```

```
                                                self.table_widget.item(self.load_row_index, 7).setText(
                                                    str(next_load))
                                                self.color_signal = 2
                                                self.T_index = 4
                                                self.load_row_index = i[0]
                                                self.table_widget.item(i[0], 4).setText(str(next_T_in))
                                                self.color_signal = 2
                                                self.T_index = 3
                                                self.load_row_index = next_row_index
                                                self.table_widget.item(next_row_index, 3).setText(str(next_T_
in))
                                                self.load_print.insertPlainText(next_load_changed_text)
                                            break
                                        while (len(m['road']) >= 5):
                                            if k == m['road'][4] and 'nnext_load' in dir():
                                                nnext_load_changed_text = '(' + '负荷由
' + self.table_widget.item(
                                                    nnext_row_index, 7).text() + 'kW' + '变为' + str(
                                                    nnext_load) + 'kW' + ')'
                                                self.color_signal = 0
                                                self.load_row_index = nnext_row_index
                                                self.table_widget.item(self.load_row_index, 7).setText(
                                                    str(nnext_load))
                                                self.load_print.insertPlainText(nnext_load_changed_text)
                                            break
                                else:
```

```
                                    self.load_print.insertPlainText(m['road'][0])
                            else:
                                if i[1] != 6:
                                    Dload = i[2] * float(self.table_widget.item(i[0], 6).text())
                                else:
                                    Dload = i[2] * abs(float(self.table_widget.item(i[0], 3).text())-float(self.table_widget.item(i[0],4).text()))
                                load_shift_nonsense = 0
                                for d in m['road']:
                                    if d in ['E01', 'E02', 'E03'] and m['idx'][0] <= 11:
                                        load_shift_nonsense = 1
                                    if load_shift_nonsense:
                                        load_shift_nonsense_text = '现行条件无法满足换热器出口温度不变'
                                load = float(self.table_widget.item(i[0], 7).text()) + Dload
                                if Exchanger[i[0]] in ['E01', 'E02', 'E03'] and len(m['road']) >= 3:
                                    next_row_index = Exchanger.index(m['road'][2])
                                    if (next_row_index <= 11 and i[0] <= 11) or (next_row_index > 11 and i[0] > 11):
                                        next_load = float(self.table_widget.item(next_row_index, 7).text()) - Dload
                                    else:
                                        next_load = float(self.table_widget.item(next_row_index, 7).text()) + Dload
                                if (i[1] == 6) and (len(m['road']) >= 3):
                                    next_row_index = Exchanger.index(m['road'][2])
                                    if (next_row_index<=11 and i[0]<=11)or(next_row_index>11 and i[0]>11):
                                        next_load = float(self.table_widget.item(next_row_index, 6).text()) * abs(float(self.table_widget.item(next_row_index, 3).text())-float(self.table_widget.item(next_row_index,4).text())) - Dload
```

```
                                else:
                                    next_load = float(self.table_widget.item(next_row_index, 6).text()) * abs(
                                        float(self.table_widget.item(next_row_index, 3).text()) - float(
                                            self.table_widget.item(next_row_index, 4).text())) + Dload
                            if i[1] == 6 and len(m['road']) >= 5:
                                nnext_row_index = Exchanger.index(m['road'][4])
                                if nnext_row_index <= 11:
                                    nnext_load = float(self.table_widget.item(nnext_row_index, 6).text()) * abs(f
   loat(self.table_widget.item(nnext_row_index, 3).text())-float(self.table_widget.item(nnext_row_index,4).text()))
                            if (m['road'][0] in ['E02', 'E03']) and i[0] >11:
                                for y in range(len(Exchanger)-1):
                                    if Exchanger[y-1] == m['road'][0] and y <= 12:
                                        T_row_index = y-1
                                next_T_in = float(self.table_widget.item(T_row_index, 4).text()) + Dload / float(
   self.table_widget.item(T_row_index,6).text())
                                self.color_signal = 2
                                self.load_row_index = T_row_index
                                self.T_index = 4
                                self.table_widget.item(self.load_row_index,self.T_index).setText(str(next_T_in))
                                self.color_signal = 2
                                self.load_row_index = Exchanger.index(m['road'][2])
                                self.T_index = 3
                                self.table_widget.item(self.load_row_index,self.T_index).setText(str(next_T_in))
                            if i[1] == 3:
                                print = '●' + '检测到' + self.table_widget.item(i[0], 0).text() + '进口温度发生变化，
   所有可能负荷转移路径为：'
```

```
                                    self.load_print.append(print)
                                    if 'load_shift_nonsense_text' in dir():
                                        self.load_print.insertPlainText(load_shift_nonsense_text)
                                    else:
                                        for k in m['road']:
                                            self.load_print.insertPlainText(k)
                                            if k == m['road'][0] and self.load_shift_chose.currentIndex() == 0:
                                                load_changed_text = '(' + '负荷由
   ' + self.table_widget.item(i[0], 7).text() + 'kW' + '变为' + str(load) + 'kW' + ')'
                                                self.load_print.insertPlainText(load_changed_text)
                                                for t in range(len(Exchanger) - 1):
                                                    if k == Exchanger[t]:
                                                        self.color_signal = 0
                                                        self.load_row_index = int(t)
                                                        self.table_widget.item(self.load_row_index, 7).setText(str(lo
   ad))
                                                        # self.table_widget.item(t, 6).setText('500')
                                            while(len(m['road']) >= 3):
                                                if k == m['road'][2] and 'next_load' in dir():
                                                    self.color_signal = 0
                                                    self.load_row_index = next_row_index
                                                    next_load_changed_text = '(' + '负荷由' + self.table_widget.item(
                                                        self.load_row_index, 7).text() + 'kW' + '变为
   ' + str(next_load) + 'kW' + ')'
                                                    self.table_widget.item(self.load_row_index, 7).setText(str(next_l
   oad))
```

```
                                        if k == m['road'][2] and 'next_load_changed_text' in dir():
                                            self.load_print.insertPlainText(next_load_changed_text)
                                        break

                            if i[1] == 4:
                                print = '●' + '检测到' + self.table_widget.item(i[0], 0).text() + '出口温度发生变化，所有可能负荷转移路径为：'
                                self.load_print.append(print)
                                if m['road'] != ['不影响换热器负荷']:
                                    if 'load_shift_nonsense_text' in dir():
                                        self.load_print.insertPlainText(load_shift_nonsense_text)
                                    else:
                                        for k in m['road']:
                                            self.load_print.insertPlainText(k)
                                            if k == m['road'][0] and self.load_shift_chose.currentIndex() == 0:
                                                load_changed_text = '(' + '负荷由' + self.table_widget.item(i[0], 7).text() + 'kW' + '变为' + str(load) + 'kW' + ')'
                                                self.load_print.insertPlainText(load_changed_text)
                                                for t in range(len(Exchanger) - 1):
                                                    if k == Exchanger[t]:
                                                        self.color_signal = 0
                                                        self.load_row_index = int(t)
                                                        self.table_widget.item(self.load_row_index, 7).setText(str(load))

                                        while (len(m['road']) >= 3):
```

```
                                            if k == m['road'][2] and 'next_load' in dir():
                                                next_load_changed_text = '(' + '负荷由
    ' + str(self.table_widget.item(next_row_index, 7).text()) + 'kW' + '变为' + str(next_load) + 'kW' + ')'
                                                self.color_signal = 0
                                                self.load_row_index = next_row_index
                                                self.table_widget.item(self.load_row_index, 7).setText(
                                                    str(next_load))
                                            if k == m['road'][2] and 'next_load_changed_text' in dir():
                                                self.load_print.insertPlainText(next_load_changed_text)
                                            break
                                    else:
                                        self.load_print.insertPlainText(m['road'][0])

                                if i[1] == 6:
                                    print = '●' + '检测到' + self.table_widget.item(i[0], 0).text() + '平均热容流率发生
    变化，所有可能负荷转移路径为：'
                                    self.load_print.append(print)
                                    if m['road'] != ['不影响换热器负荷']:
                                        if 'load_shift_nonsense_text' in dir():
                                            self.load_print.insertPlainText(load_shift_nonsense_text)
                                        else:
                                            for k in m['road']:
                                                self.load_print.insertPlainText(k)
                                                if k == m['road'][0] and self.load_shift_chose.currentIndex() == 0:
```

```
                                                load_changed_text = '(' + '负荷由
   ' + self.table_widget.item(i[0], 7).text() + 'kW' + '变为' + str(load) + 'kW' + ')'
                                                self.load_print.insertPlainText(load_changed_text)
                                                for t in range(len(Exchanger) - 1):
                                                    if k == Exchanger[t]:
                                                        self.color_signal = 0
                                                        self.load_row_index = int(t)
                                                        self.table_widget.item(self.load_row_index, 7).setText(st
   r(load))


                                            while (len(m['road']) >= 3):
                                                if k == m['road'][2] and 'next_load' in dir():
                                                    next_load_changed_text = '(' + '负荷由
   ' + self.table_widget.item(
                                                            next_row_index, 7).text() + 'kW' + '变为' + str(
                                                            next_load) + 'kW' + ')'
                                                    self.color_signal = 0
                                                    self.load_row_index = next_row_index
                                                    self.table_widget.item(self.load_row_index, 7).setText(
                                                            str(next_load))
                                                    self.load_print.insertPlainText(next_load_changed_text)
                                                break
                                            while (len(m['road']) >= 5):
                                                if k == m['road'][4] and 'nnext_load' in dir():
```

```
                                                    nnext_load_changed_text = '(' + '负荷由
    ' + self.table_widget.item(nnext_row_index, 7).text() + 'kW' + '变为' + str(nnext_load) + 'kW' + ')'
                                                    self.color_signal = 0
                                                    self.load_row_index = nnext_row_index
                                                    self.table_widget.item(self.load_row_index, 7).setText(
                                                            str(nnext_load))
                                                    self.load_print.insertPlainText(nnext_load_changed_text)
                                                break
                                    else:
                                        self.load_print.insertPlainText(m['road'][0])


            self.load_print.setStyleSheet('font: 18px')
            self.load_print.setReadOnly(True)




    if __name__ == '__main__':
        app = QApplication(sys.argv)
        apply_stylesheet(app, theme='light_blue.xml')
```

```
3328.       w = MainWindow()
3329.       w.show()
3330.       sys.exit(app.exec())
```