高比例风电电力系统储能运行及配置分析

摘要

本文围绕可再生能源输出功率强随机波动性导致系统运行中功率实时平衡困难以及储能成本相对昂贵，利用储能平衡系统功率将增加系统运行成本等问题，通过计算得到的结果分析风电装机替代火电机组造成的影响，并建立优化模型来解决供电平衡问题。

针对问题一：输入一天的负荷曲线利用等微增率法建立优化模型求出三台机组的最优发电计划，经过判断筛选得到正确的发电计划；随后将其代入题目所给公式求得煤使用量，即可得到火电运行成本；随后求出碳捕集成本；最后将火电运行成本与碳捕集成本相加求出火电成本，即可求出单位供电成本。

针对问题二：首先求出风电机组替换机组 3 后的净负荷曲线，根据净负荷曲线并采用等微增率法求出机组 1 和 2 的发电计划，经过判断筛选得到正确的发电计划，根据发电计划算出弃风功率曲线，并对弃风功率进行积分即可求出总的弃风量；最后根据出机组总功率与负荷功率之差的最小值计算出风机功率的富余量。

针对问题三：首先求出风电机组替换机组 2 后的净负荷曲线，根据净负荷曲线并采用等微增率法求出机组 1 和 2 的发电计划，经过判断筛选得到正确的发电计划；随后算出弃风功率曲线与失负荷功率曲线，求积分可得总的弃风量以及总失负荷量；最后根据机组总功率与负荷功率之差的最小值计算出风机功率的缺额量。

针对问题四：根据问题二和问题三，可以得到弃风总量，从而计算出弃风损失；接着计算出失负荷功率并对其求积分得到失负荷量 (问题二中不需要考虑失负荷量)，并计算出失负荷损失，并如问题一所述求得火电成本，从而求出发电总成本，即可得到系统单位供电。

针对问题五：首先求出风电机组替换机组 2，3 后的净负荷曲线，根据净负荷曲线并采用等微增率法求出机组 1 的发电计划；并算出总的弃风量以及总失负荷量，接着算出失负荷损失、火电成本和风电运维成本；接着结合计算出的投资成本和运维成本即可求得储能成本，最后求出发电总成本即可求得单位供电成本。

针对问题六：由求解上述问题所得到风电装机 300MW 替代机组 3、风电装机 600MW 替代机组 2 时所得的功率平衡图以及相关指标统计分析出风电替代容量的递增会破坏系统的功率平衡，对系统的供电能力带来了不稳定性与波动性。但同时系统单位供电成本会随着风电替代容量的递增而减少。

针对问题七：我们针对装机容量 1200MW 替代机组 2，3 的情况分析功率平衡所存在的问题，通过建立以最优供电成本为目标、储能功率和储能容量为决策变量的优化模型，并采用遗传算法计算上述模型，得到该系统最优供电成本的功率平衡解决方案。

关键字： 等微增率；弃风量；最优功率；失负荷量

一、 无风电接入时系统单位供电成本

1.1 本题解题流程

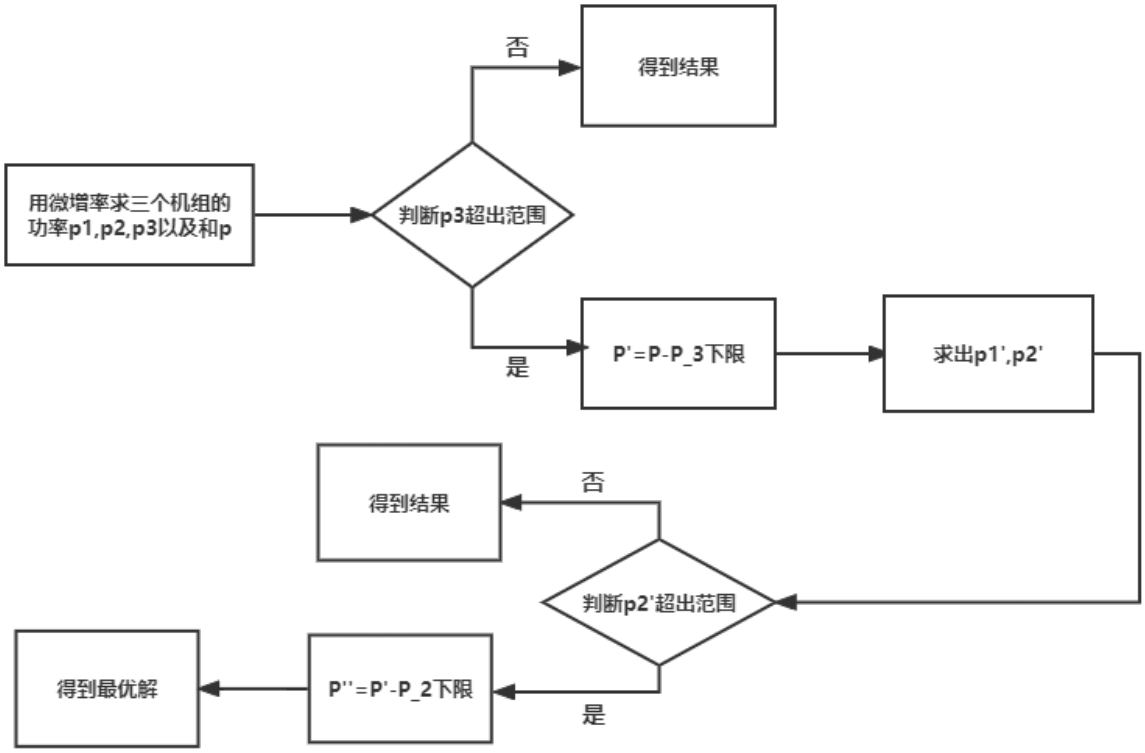

图 **1**　第一题最优功率求解过程

上图是第一题最优功率求解流程图，其过程如下：

1. 三机组求解最优功率
2. 判断三机组功率是否超出范围
3. 若超出范围，则判断是第几号机组超限，将其剔除再使用其他二者进行双机组最优功率求解
4. 判断所得双机组最优功率解是否在范围内
5. 若超出范围，则未超限机组功率为剩余目标功率与超限机组功率限制之差

1.2 数据处理

附件一中使用的数据是标幺值（$p.u.$），因此需要将其转换为真实值。其转换公式如下：

$$A = a \times A_B \tag{1}$$

式中：A 为数据的真实值，a 为标幺值（$p.u.$），A_B 为数据的基准值。

将附件一中的负荷、风电功率数据处理，存放于同一表（**附件一.xlsx**）中。其中：**l900** 为负荷的真实功率，**w300** 为装机容量为 300MW 时的风机功率，**w600** 为装机容量为 600MW 时的风机功率，**w900** 为装机容量为 900MW 时的风机功率。

数据处理部分后文不再赘述。

1.3 三机组等微增率最优解

三机组等微增率最优解共需要计算 3 个参数，即机组 1 最优功率 P_1、机组 2 最优功率 P_2、机组 3 最优功率 P_3，其需要传入的参数为：目标功率 P_d 和各机组煤耗量系数 a，b，c。

根据式13，写出三台机组的目标函数如下：

$$\begin{cases} c_1 &= 0.226p_1^2 + 30.42p_1 + 786.8 \\ c_2 &= 0.588p_2^2 + 65.12p_2 + 451.32 \\ c_3 &= 0.785p_3^2 + 139.6p_3 + 1049.5 \end{cases} \tag{2}$$

分别在上述目标函数对 p_i 求偏导，可得：

$$\begin{cases} \dfrac{\partial c_1}{\partial p_1} &= 0.452p_1 + 30.42 \\ \dfrac{\partial c_2}{\partial p_2} &= 1.176p_2 + 65.12 \\ \dfrac{\partial c_3}{\partial p_3} &= 1.57p_3 + 139.6 \end{cases} \tag{3}$$

由约束条件可知：

$$\sum_{i=1}^{3} p_i = p_d \tag{4}$$

且：

$$\frac{\partial c_1}{\partial p_1} = \frac{\partial c_2}{\partial p_2} = \frac{\partial c_3}{\partial p_3} \tag{5}$$

将上述方程求解即可得到单次三机组等微增率最优解。

1.4 双机组等微增率最优解

双机组等微增率最优解算法与三机组类似，其目标函数与偏导函数为 2 个，共计 2 条方程。通过约束条件与偏导函数即可求解双机组等微增率最优解。

1.5 算法代码实现

将1.3中的过程可以用如下流程图表示：

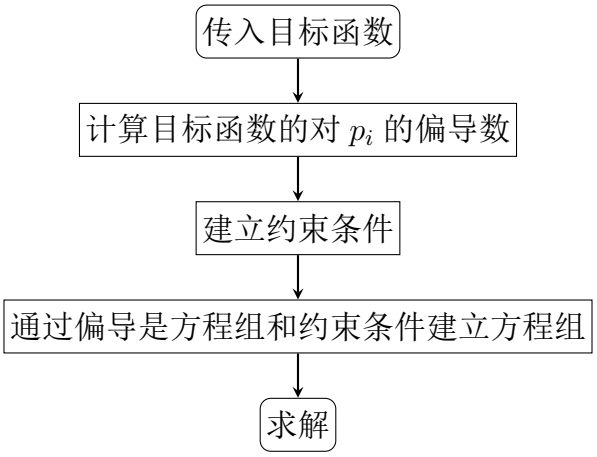

图 **2**　三机组等微增率计算流程

1.6 本题求解过程

1.6.1 煤使用量及运行维护成本计算

煤使用量 F 计算公式如下：

$$F_i = a_i P_i^2 + b_i P_i + c_i \tag{6}$$

其中 a，b，c 为各机组煤耗量系数。

1.6.2 碳捕集成本

当日发电量为发电功率在时间上的积分，因此可以根据发电计划对时间积分即可求出当日发电量。碳捕集量为 当日发电量 × 机组碳排放量参数。

1.6.3 单位供电成本

系统总负荷量为系统负荷功率在时间上的积分，因此可以根据系统负荷曲线进行系统总负荷量的求解。

根据定义，单位供电成本为 系统总成本/系统总负荷量，而系统总成本为 火电成本 + 风电成本 + 储能成本 + 弃风损失 + 失负荷损失，本题中，后四者均无需考虑，因此在本题中系统总成本为 火电成本，即 火电运行成本 + 碳捕集成本。

1.7 求解结果

1.7.1 发电计划曲线

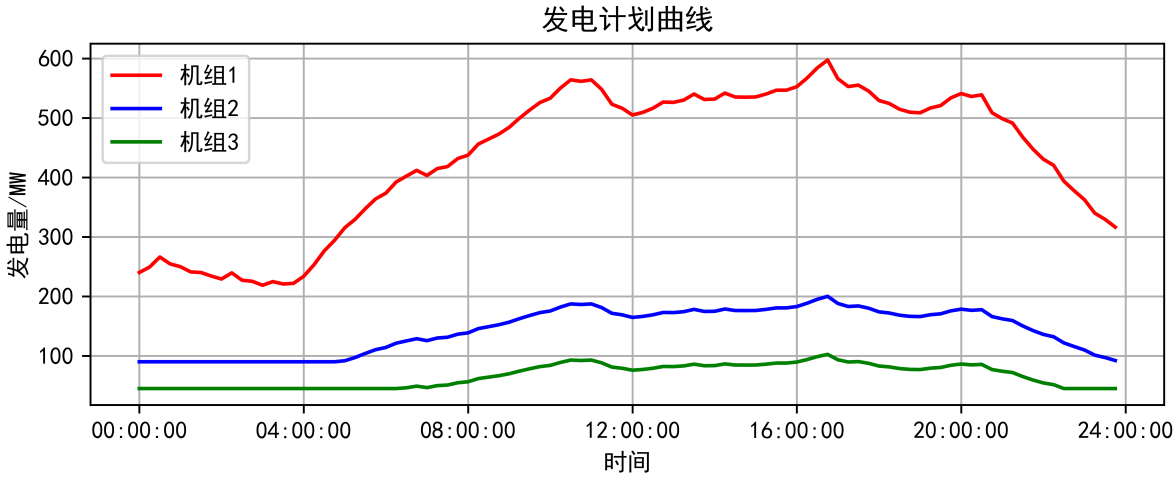

图 **3**　机组 **1**、**2**、**3** 发电计划曲线

1.7.2 风电电量占比为 0 时系统相关指标统计

表 **1**　风电电量占比为 **0** 时系统相关指标统计

碳捕集成本（元/t）	火电运行成本（万元）	碳捕集成本（万元）	总发电成本（万元）	单位供电成本（元/KWh）
0	244.573	0	244.573	0.158
60	244.573	68.288	312.861	0.202
80	244.573	91.05	335.624	0.216
100	244.573	113.813	358.386	0.231

1.8 代码封装及实现

将节1.3中的过程封装为函数 **equalIR_algorithm_3**，其详细代码见**附录 B-Python 程序源代码-第一题-第一题：等微增率求解发电曲线.py**。其中传入参数为 P_d，即目标功率；返回结果为三个机组的最优功率值 P_1，P_2，P_3。

双机组等微增率求解封装为函数 **equalIR_algorithm_2**,其详细代码见**附录 B-Python 程序源代码-第一题-第一题：等微增率求解发电曲线.py**。其中传入参数为 P_d，即目标功率；返回结果为两个机组的最优功率值 P_1，P_2。

节1.1中最优功率求解过程范围是否超限使用函数 **judge** 判断，其传入参数为 p：功率，up：功率上限，down：功率下限，其详细代码见**附录 B-Python 程序源代码-第一题-第一题：等微增率求解发电曲线.py**。

将将节1.6.1各机组煤耗量系数封装为函数 **fireStationCoalCost**，将运行维护成本（0.5 倍的煤耗成本）封装为 **fireStationRunCost**，上述函数存放在**附录 B-Python 程序源代码-第一题-fireStationCost.py** 中。

将节1.6.2的计算过程封装为函数 **carbonEmission**，其传入参数为：1、发电计划，2、机组碳排放量参数；返回结果为碳捕集量。碳捕集成本为 碳捕集量 × 碳捕集单价。据此即可计算碳捕集成本。

上述函数存放在**附录 B-Python 程序源代码-第一题-fireStationCost.py** 中。

二、 300MW 风机替代机组 3 后系统功率平衡变化情况分析

2.1 本题求解流程

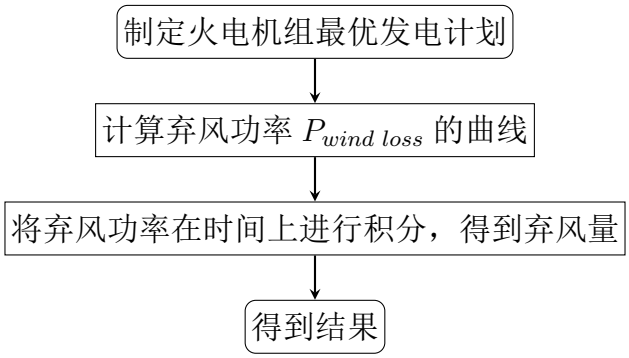

图 4　第二题求解流程

上图是第二题求解的流程图，我们先通过等微增率算法计算得出出火电机组 1、2 的最优发电计划，再计算出弃风功率 $P_{wind\ loss}$ 的曲线，接着将弃风功率在时间上进行积分，得到弃风量，最后得到结果。

2.2 弃风量的计算

弃风功率为系统功率超出负荷所需功率的部分。弃风功率 $P_{wind\ loss}$ 可以通过下面式子进行表达：

$$P_{wind\ loss} = P_{fire} + P_{wind} - P_d \tag{7}$$

式中：P_{fire} 为火电机组功率总和，P_{wind} 为风电机组功率。

弃风量为弃风功率在时间上的积分，即：

$$\text{弃风量} = \int_{t_0}^{t_1} P_{wind\ loss} dt \tag{8}$$

2.3 失负荷量的计算

失负荷功率为系统功率小于负荷所需功率的部分。失负荷功率 $P_{load\ loss}$ 可以通过下面式子进行表达：

$$P_{load\ loss} = P_d - (P_{fire} + P_{wind}) \tag{9}$$

式中：P_{fire} 为火电机组功率总和，P_{wind} 为风电机组功率。

失负荷量为失负荷功率在时间上的积分，即：

$$\text{失负荷量} = \int_{t_0}^{t_1} P_{load\ loss} dt \tag{10}$$

2.4 计算弃风量、失负荷量

将节2.2、节2.3中的流程封装为函数 **get_wind_loss**，其传入参数为：1、风电功率，2、各机组功率，3、负荷功率；返回值为 *系统机组总功率之和 – 负荷功率*。

对于该函数的返回值可以发现，其正值为起风功率，负值则为失负荷功率。

因此根据弃风量的定义，可以将弃风量封装成函数 **get_wind_load**，其传入参数为上一函数 **get_wind_loss** 的返回值，返回值为弃风量。

对于失负荷量，本题中未要求计算，因此在本题代码中未对其进行封装。封装代码在第五题中体现。

上述函数存放在**附录 B-Python 程序源代码-第二题-第二题：弃风.py** 中。

2.5 本题求解过程

2.5.1 火电机组最优功率

本题中，风机替代了机组 3，根据节1.4中双机组最优功率求解可以计算出机组 1、2 的最优功率发电曲线。

2.5.2 计算风机功率富余量

通过节2.4中的函数 **get_wind_loss** 可以计算系统机组总功率与负荷功率之差，其中差值的最小值即为系统的风机功率富余量。

2.6 计算结果

可减少的风电装机容量为：40.789MW，即最小风电装机容量为：259.211MW。弃风电量为：20.113MWh。

功率平衡变化如下图：

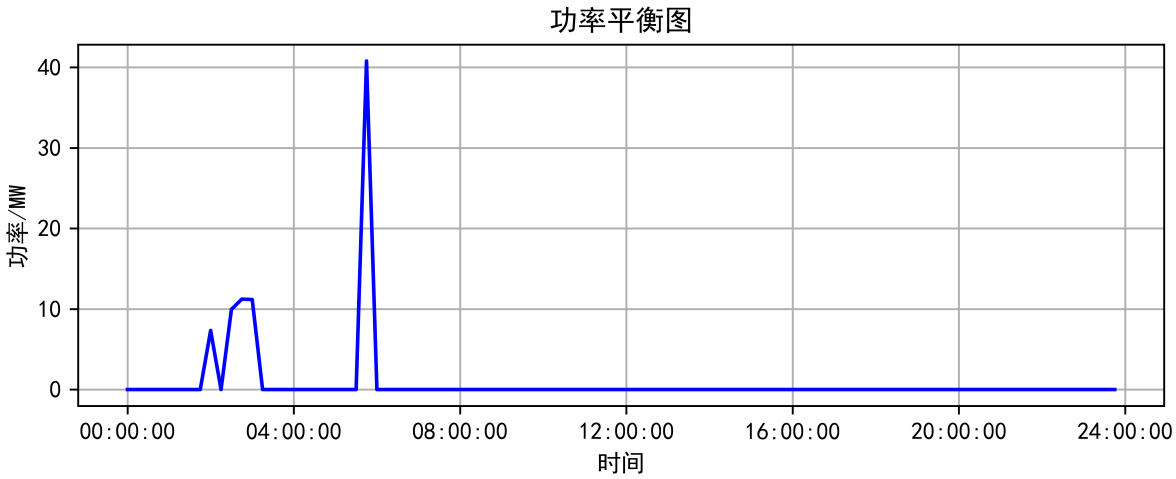

图 **5**　功率平衡图

从图中可以看到，功率平衡存在 *系统总功率 ≥ 负荷所需功率* 的情况，因此会导致出现弃风现象。

三、 600MW 风机替代机组 2 后系统功率平衡变化情况分析

3.1 本题求解过程

3.1.1 本题求解流程

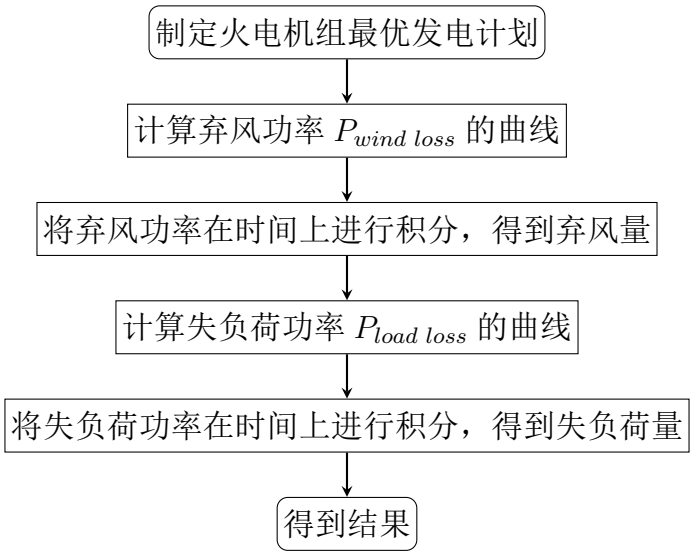

图 **6**　第三题求解流程

上图是第三题求解的流程图，我们首先通过等微增率算法计算制定火电机组 1、3 的最优发电计划，然后再计算弃风功率 $P_{wind\ loss}$ 的曲线，接着将弃风功率在时间上进行积分，得到弃风量，而后再计算失负荷功率 $P_{load\ loss}$ 的曲线，再而后将失负荷功率在时间上进行积分，得到失负荷量，最后得到结果。

3.1.2 火电机组最优功率

本题中，风机替代了机组 2，根据**节1.4**中双机组最优功率求解可以计算出机组 1、3 的最优功率发电曲线。

3.1.3 计算风机功率缺额

通过节2.4中的函数 **get_wind_loss** 可以计算系统机组总功率与负荷功率之差，其中差值的最小值即为系统的风机功率缺额。

3.2 计算结果

缺额的风电装机容量为：47.67MW，即需要增加接入风机装机容量为：47.67MW。弃风电量为：268.861MWh。

功率平衡变化如下图：

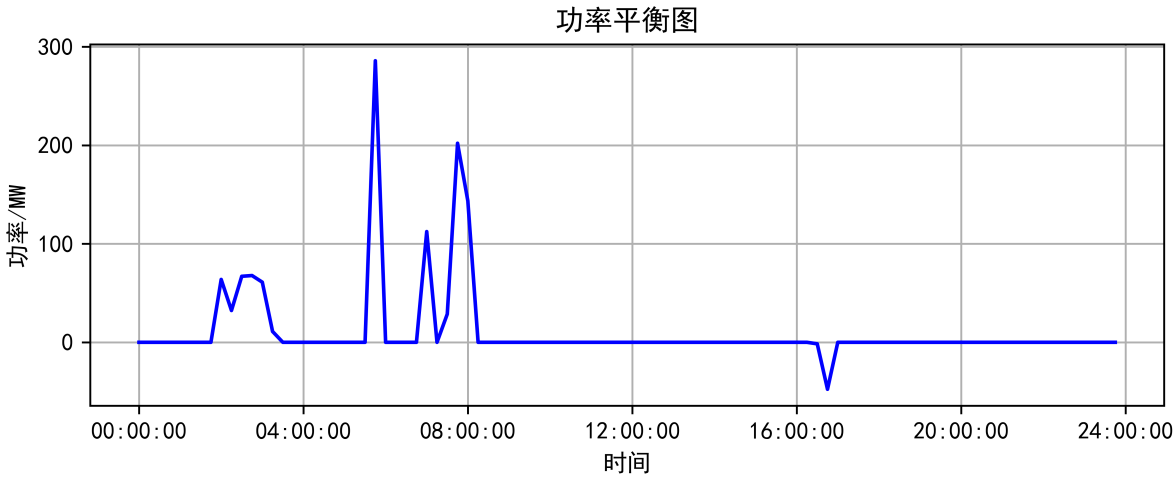

图 **7**　功率平衡图

从图中可以看到，功率平衡存在 *系统总功率* ≥ *负荷所需功率* 的情况，因此会导致出现弃风现象。

同时也存在 *系统总功率* ≤ *负荷所需功率* 的情况，此时会导致出现失负荷现象。

四、 针对上述两种替代情景计算相关数据

4.1 本题求解过程

4.1.1 本题求解流程

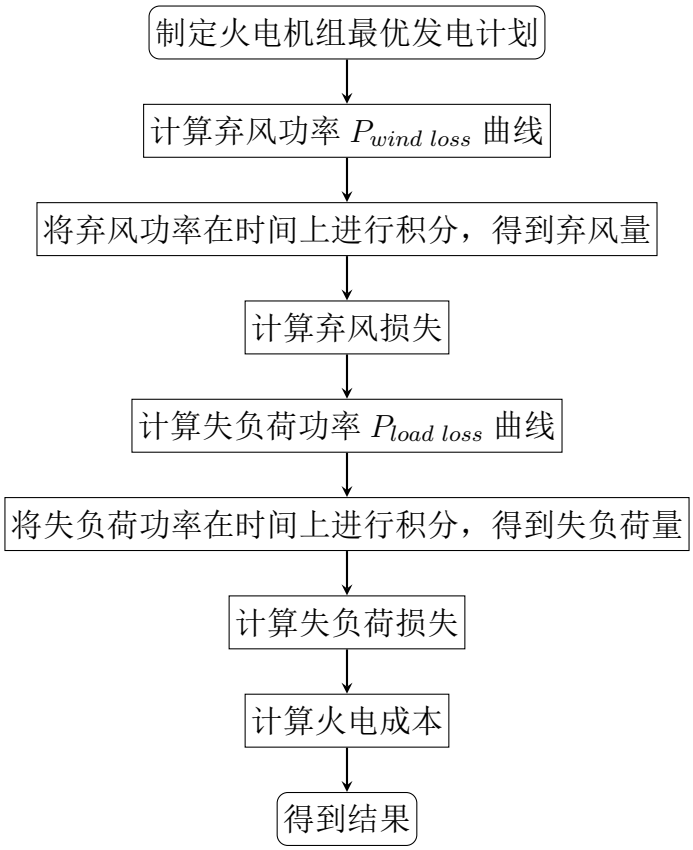

图 **8**　第四题求解流程

上图是第四题求解的流程图，我们先通过等微增率算法计算得出火电机组最优发电计划以及计算出弃风功率 $P_{wind\ loss}$ 的曲线，然后将弃风功率在时间上进行积分，得到弃风量，结合以上结果计算出弃风损失。再然后我们计算失负荷功率 $P_{load\ loss}$ 曲线，同时将失负荷功率在时间上进行积分，得到失负荷量。之后，我们计算失负荷损失以及火电成本。最后得出结果。

4.2 对于 300MW 风机替代机组 3

原始数据可以直接使用第二题的结果。

从第二题存储的运行数据中取出差值，调用函数 **get_wind_load** 计算出弃风量，从而计算出弃风损失。由于替代机组 3 的情况下没有失负荷，因此不需要计算失负荷量。

火电成本可以根据第一题方式进行计算。将不同的碳捕集单价放入函数中即可得到不同的系统运行单价。

计算结果

表 **2**　风电装机 **300MW** 替代机组 **3** 时系统相关指标统计

碳捕集成本（元/t）	火电运行成本（万元）	碳捕集成本（万元）	风电运维成本（万元）	弃风损失		失负荷损失		单位供电成本（元/KWh）
				弃风电量（MW）	弃风损失（万元）	失负荷电量（MW）	失负荷损失（万元）	
0.000	201.892	0.000	9.115	20.113	0.603	0.000	0.000	0.136
60.000	201.892	58.939	9.115	20.113	0.603	0.000	0.000	0.174
80.000	201.892	78.585	9.115	20.113	0.603	0.000	0.000	0.187
100.000	201.892	98.232	9.115	20.113	0.603	0.000	0.000	0.200

4.3 对于 600MW 风机替代机组 2

原始数据可以直接使用第三题的结果。

从第三题存储的运行数据中取出差值。调用函数 **get_wind_load** 计算出弃风量，从而计算出弃风损失。调用函数 **get_loss_load** 计算出失负荷量，从而计算出失负荷损失。

火电成本可以根据第一题方式进行计算。将不同的碳捕集单价放入函数中即可得到不同的系统运行单价。

计算结果

表 **3**　风电装机 **600MW** 替代机组 **2** 时系统相关指标统计

碳捕集成本（元/t）	火电运行成本（万元）	碳捕集成本（万元）	风电运维成本（万元）	弃风损失		失负荷损失		单位供电成本（元/KWh）
				弃风电量（MW）	弃风损失（万元）	失负荷电量（MW）	失负荷损失（万元）	
0.000	167.326	0.000	18.230	268.861	8.066	12.304	9.843	0.131
60.000	167.326	50.906	18.230	268.861	8.066	12.304	9.843	0.164
80.000	167.326	67.875	18.230	268.861	8.066	12.304	9.843	0.175
100.000	167.326	84.844	18.230	268.861	8.066	12.304	9.843	0.186

4.4 代码封装以及实现

函数 **get_loss_load** 存放在附录 **B-Python** 程序源代码-第四题-**600MW** 风电替代机组 **2**-第四题：计算费用-**600MW** 替代**.py** 中。

五、 计算 900MW 替代 2，3 机组相关数据

5.1 本题求解过程

5.1.1 本题求解流程

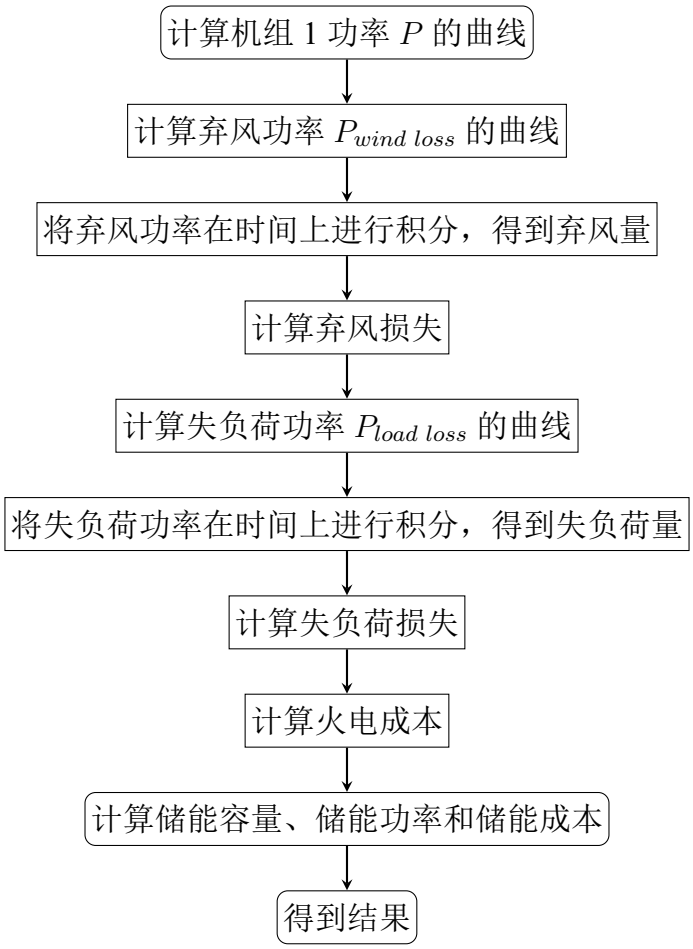

图 **9**　第五题求解流程

上图是第五题求解的流程图，我们先计算机组 1 功率 P 以及计算弃风功率 $P_{wind\ loss}$ 的曲线，然后将弃风功率在时间上进行积分，得到弃风量。然后计算弃风损失以及失负荷功率 $P_{load\ loss}$ 的曲线，将失负荷功率在时间上进行积分，得到失负荷量。在之后我们计算失负荷损失，火电成本以及储能成本。最后得出结果。

5.1.2 机组功率

在本题中，由于风电替代了机组 2、3，因此机组 1 功率可以用 $目标功率 - 风电功率$ 进行计算。但计算的同时需要关注机组出力上下限问题，因此需要使用函数 **judge** 进行辅助判断。

5.1.3 储能成本

储能成本分为两大类：投资成本和运维成本。

1）投资成本

投资成本又分为单位功率成本和单位能量成本。其中单位功率成本指：每建设 1KW 功率的储能系统所需的成本；单位能量成本指：所建设储能系统的容量每增加 1KWh 所需要的成本。

由于投资成本是一次性费用，因此需要将其均摊才能计入系统总成本计算中。此处储能系统的设计寿命是 10 年，因此所计算的储能成本应 /10 年/365 天，所得结果方可使用。

2）运维成本

单位运维成本指储能系统每放出 1KWh 所需的运维费用，因此总运维费用可以用总放电量 × 单位运维成本 进行计算。

5.1.4 储能容量与功率

对于本题需要的储存容量和储能功率，其计算方式如下：

1. 计算系统最大功率缺额
2. 所需储能功率 ≥ 最大功率缺额 × 储能效率
3. 计算系统最大容量缺额
4. 所需储存容量 ≥ 最大容量缺额 × 储能效率

5.2 计算结果

风电装机 900MW、替代机组 2、3 时，失负荷电量 215.891MWh，最大失负荷功率为 146.505MW，需要配备最小储能容量为 162.783MWh。考虑碳捕集成本 60 元/吨，此时单位供电成本为 0.147 元/KWh。

六、 探究风电接入对系统供电的影响

根据节 1.6 中表 1 风电电量占比为 0 时系统相关指标统计可得，火电运行成本不随着单位碳捕集成本的改变而改变，为 244.573 万元。同时，当单位碳捕集成本逐渐增大时，碳捕集成本、总发电成本以及单位供电成本都在不断增加。

风电装机 300MW、替代机组 3 时，根据节 2.6 中图 5 功率平衡图可得，系统功率在 2 时到 4 时期间以及 6 时左右经历了两次陡增，破坏了原有的功率平衡，风功率过剩，面临保消纳的问题，对系统可靠供电造成挑战。同时将节 4.2 中表 2 风电装机 300MW 替代机组 3 时系统相关指标统计与节 1.6 中表 1 风电电量占比为 0 时系统相关指标统计的数据进行比对，发现：风电装机 300MW 替代机组 3，单位碳捕集成本相同时，单位供电成本均得到了 13% 左右的下降。

风电装机 600MW、替代机组 2 时，根据节 3.2 中图 7 功率平衡图可得，系统功率在 2 时到 4 时期间、6 时到 8 时期间、系统功率陡增，风功率过剩，面临保消纳的问题。17 时左右系统功率骤降，供应不足，系统面临着保供应的问题。二者都对系统可靠用电造成挑战。同时将节 4.3 中表 3 风电装机 600MW 替代机组 2 时系统相关指标统计与节 1.6 中表 1 风电电量占比为 0 时系统相关指标统计的数据进行比对，发现：风电装机 600MW 替代机组 2，单位碳捕集成本相同时，单位供电成本均得到了 17% 左右的下降。

综合上述分析可得，风电替代容量的递增会破坏系统的功率平衡，对系统的供电能力带来了不稳定性与波动性。但同时为保障可靠供电，系统单位供电成本会随着风电替代容量的递增而减少。

七、 风电装机 1200MW 替代机组 2，3 时对功率平衡的分析

7.1 功率平衡

本节功率平衡计算与第三题中功率平衡算法相同。下面是其结果：

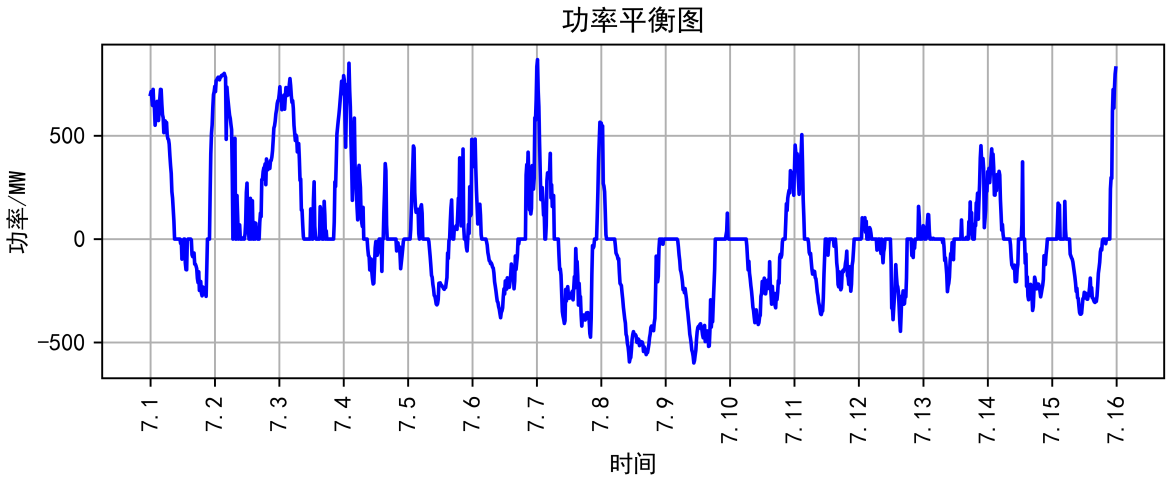

图 **10**　功率平衡图

7.2 遗传算法

遗传算法（Genetic Algorithm, GA）是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法 [1]。

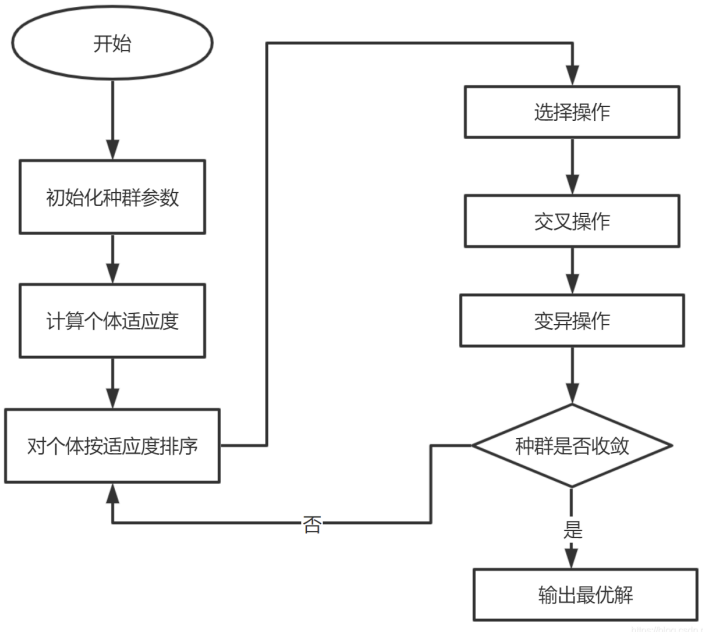

图 **11**　遗传算法流程

本节通过该算法进行最优解计算。

7.3 系统优化

7.3.1 优化目标

系统优化目标为系统供电成本 C（元/KWh）。即：

$$C = f(C_{es}, P_{es}) \tag{11}$$

其中：C_{es} 为储能系统的容量，P_{es} 为储能系统的功率。

目标函数使用使用第五题中的算法，其详细过程见节5.1。

7.3.2 优化变量

需要优化的变量有两个：储能系统的容量 C_{es} 和储能系统的功率 P_{es}。根据节7.1结果，最大功率缺额约为 600MW，最大容量缺额为 150MWh。因此，我们可以得到以下的约束条件：

$$\begin{cases} 0 \leq P_{es} \leq 600 \\ 0 \leq C_{es} \leq 150 \end{cases} \tag{12}$$

7.3.3 遗传算法求解

根据约束条件，即式12，可以得到：对于单个变量而言，若精确到小数点后 5 为，则应有的染色体数目为 20。

因此，本节中，我们使每个个体为 40 条染色体，其中前 20 代表储能系统的容量 C_{es}，后 20 为储能系统的功率 P_{es}；种群共有 50 个个体。

7.3.4 遗传算法结果（碳捕集成本为 0 元/吨）

使用参数为：$c = 0.4$，$m = 0.02$，$iter\_num = 500$，迭代过程如下图：

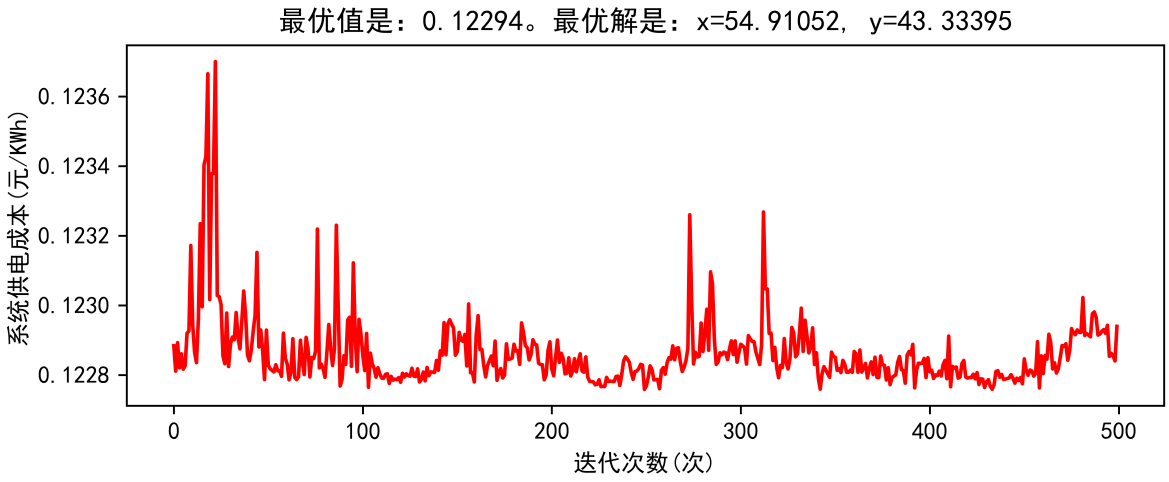

图 **12**　遗传算法结果

可以发现，当储能容量为 54.911MWH，储能功率为 43.334MW 时，所得的系统供电成本最低，为 0.12294 元/KWh。

7.3.5 其他情况

其他碳捕集成本下的情况如下表所示：

表 **4**　不同碳捕集成本下的储能最优解

碳捕集成本（元/吨）	储能容量（MWh）	储能功率（MW）	系统供电成本（元/KWh）
0	54.911	43.334	0.12294
60	127.633	40.916	0.14501
80	45.568	74.929	0.15209
100	45.332	19.703	0.15934

表 **5**　风电装机 **1200MW** 替代机组 **2,3** 时供电成本

碳捕集成本（元/吨）	储能容量（MWh）	储能功率（MW）	系统供电成本（元/KWh）
0	0	0	0.12276
60	0	0	0.14462
80	0	0	0.15191
100	0	0	0.15920

由于不同各种储能装置由于具有对功率和能量的时间迁移能力,是充分改善现有的功率不平衡的有效手段。结合表 4 表 5 的数据，不同单位碳捕集成本下，在储能最优解时，系统供电成本与同情况下储能未接入时系统供电成本所差无几。在经济性上，本方案具有一定的可行性。同时由于储能元件的自身的特性，本方案也具有显著的有效性。

7.4 代码的封装及实现

将节7.3.3的遗传算法封装为函数 **ga**，其传入参数为交叉概率 c，变异概率 m，迭代次数 $iter\_num$。

函数 **ga** 的详细代码在**附录 B-Python 程序源代码-第七题-第七题：遗传算法求最优解.py** 中。

八、 结论

问题一：碳捕集成本为 0 时，单位供电成本为 0.158 元/KWh; 碳捕集成本为 60 时，单位供电成本为 0.202 元/KWh; 碳捕集成本为 80 时，单位供电成本为 0.216 元/KWh; 碳捕集成本为 100 时，单位供电成本为 0.231 元/KWh。

问题二:风电装机 300MW 替代机组 3 时,功率平衡存在 $系统总功率 \geq 负荷所需功率$ 的情况,弃风电量为 20.113MWh,为减少弃风又不失负荷风电装机容量需要降低 40.789MW。

问题三:风电装机 600MW 替代机组 2 时,功率平衡存在 $系统总功率 \geq 负荷所需功率$ 的情况，因此会导致出现弃风现象。同时也存在 $系统总功率 \leq 负荷所需功率$ 的情况，此时会导致出现失负荷现象。为不失负荷，风电装机容量要增加 47.67MW。

问题四：对于风电装机 300MW 替代机组 3 时，碳捕集成本为 0 时，单位供电成本为 0.136 元/KWh; 碳捕集成本为 60 时，单位供电成本为 0.174 元/KWh; 碳捕集成本为 80 时，单位供电成本为 0.187 元/KWh; 碳捕集成本为 100 时，单位供电成本为 0.200 元/KWh。对于风电装机 600MW 替代机组 2 时，碳捕集成本为 0 时，单位供电成本为 0.131 元/KWh; 碳捕集成本为 60 时，单位供电成本为 0.164 元/KWh; 碳捕集成本为 80 时，单位供电成本为 0.175 元/KWh; 碳捕集成本为 100 时，单位供电成本为 0.186 元/KWh。

问题五：风电机组 900MW 替代机组 2，3 时，失负荷电量为 215.891MWh。为不失负荷，最小储能容量为 162.783MWh。碳捕抓成本为 60 元/t 时，系统单位供电成本为 0.147 元/KWh。

问题六：结合上述题目所得到的表格数据可分析，风电替代容量的递增会破坏系统的功率平衡，对系统的供电能力带来了不稳定性与波动性。但同时为保障可靠供电，系统单位供电成本会随着风电替代容量的递增而减少。

问题七：风电机组 1200MW 替代机组 2，3 时功率平衡地波动性较大。通过计算可得，当储能容量为 54.911MWH，储能功率为 43.334MW 时，所得的系统供电成本最低，为 0.12294 元/KWh。我们通过建立以最优供电成本为目标、储能功率和储能容量为决策变量的优化模型，并采用遗传算法计算上述模型，得到该系统最优供电成本的功率平衡解决方案。

参考文献

[1] 大灰狼学编程，遗传算法 Python 代码实现 [OL]，https://blog.csdn.net/a_hui_tai_lang/article/details/119900038，2022.5.29.

[2] 王葵，孙莹，电力系统自动化（第三版）[M]，北京：中国电力出版社，2012，86-87.

[3] 王锡凡，现代电力系统分析 [M] ，北京：科学出版社，2003，117-134.

[4] 李明节, 陈国平, 董存, 梁志峰, 王伟胜, 范高锋，新能源电力系统电力电量平衡问题研究 [J]，电网技术, 2019, 43(11): 3979-3986.

[5] 鲁宗相, 李海波, 乔颖，高比例可再生能源并网的电力系统灵活性评价与平衡机理 [J]，中国电机工程学报, 2017, 37(01): 9-20.

[6] 舒印彪, 张智刚, 郭剑波, 张正陵，新能源消纳关键因素分析及解决措施研究 [J]，中国电机工程学报, 2017, 37(01): 1-9.

附录 A　火电机组最优功率计算过程

实际情况中，发电燃料成本可以表示成功率的二次函数，即 $c = \alpha + \beta P + \gamma P^2$。燃料成本曲线上某一点的斜率即为成本微增率。

等微增率算法 [2]

在忽略线路损耗的前提下，总的负荷需求等于总的发电机有功输出功率。C_i 是每一个发电厂的成本函数，并且是已知的。于是求最优功率的过程就转化成求每一个发电厂的有功功率以使得目标函数为：

$$C_t = \sum_{i=1}^{n_g} C_i = \sum_{i=1}^{n_g} \alpha_i + \beta_i P_i + \gamma_i P_i^2 \tag{13}$$

最小。由于总负荷等于总输出功率，因此可以得到目标约束函数为：

$$\sum_{i=1}^{n_g} P_i = P_D \tag{14}$$

式中：n_g 为发电厂总数量；P_D 为总的负荷需求；C_t 为总的发电成本；C_i 为第 i 个发电厂的发电成本；P_i 为第 i 个发电厂的有效输出功率。

构造拉格朗日函数把约束条件放到目标函数中去，则有：

$$l = C_t + (P_D - \sum_{i=1}^{n_g} P_i)\lambda \tag{15}$$

取得极值的条件为偏导数为零。则有：

$$\frac{\partial L}{\partial P_i} = 0 \tag{16}$$

$$\frac{\partial L}{\partial \lambda} = 0 \tag{17}$$

首先由式16得

$$\frac{\partial C_t}{\partial P_i} + \lambda(0 - 1) = 0 \tag{18}$$

由于 C_t=C_1+C_2+C_3+...+C_{n_g}, 所以有：

$$\frac{\partial C_t}{\partial P_i} = \frac{dC_i}{dP_i} = \lambda \tag{19}$$

因此最优分配条件变为：

$$\frac{dC_i}{dP_i} = \lambda \qquad i = 1, ..., n_g \tag{20}$$

或者

$$\beta_i + 2\gamma_i P_i = \lambda \tag{21}$$

再由式17得

$$\sum_{i=1}^{n_g} P_i = P_D \tag{22}$$

式22是最初的等式约束条件。总的来说，当忽略线路损耗并且无发电机输出功率限制的情况下，为使总花费最小，应按照相等的燃料成本微增率在发电设备或发电厂之间分配负荷。

附录 B　Python 程序源代码

2.1　第一题

2.1.1　fireStationCost.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 17:13
# @File :  火电运行成本.py
# @IDE :  PyCharm

#########
# 煤用量
# 火电烧煤运行成本
def fireStationCoalCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    return p ** 2 * a + p * b + c


# 火电煤维护成本
def fireStationMaintenanceCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
```

```
    :param c: 煤用量参数3
    :return:
    '''
    coalCost = fireStationCoalCost(p, a, b, c)
    return coalCost * 0.5


# 单次火电煤运行成本
def fireStationRunCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    coalCost = fireStationCoalCost(p, a, b, c)
    maintenanceCost = fireStationMaintenanceCost(p, a, b, c)
    return coalCost + maintenanceCost


######
# 碳费用
# 总发电量
def totalGeneration(powerPlan):
    '''
    :param powerPlan: 电力计划
    :return:
    '''
    return sum(powerPlan) * 0.25


# 碳排放量
def carbonEmission(powerPlan, carbonWeight):
    '''
    :param powerPlan: 电力计划
    :param carbonWeight: 碳排放参数
    :return:
    '''
    return totalGeneration(powerPlan) * carbonWeight
```

2.1.2 第一题：等微增率求解发电曲线.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 9:14
```

```python
# @File :  第一题：等微增率求解发点曲线.py
# @IDE :  PyCharm

import pandas as pd
import sympy as sp
from matplotlib import pyplot as plt


# 三机组等微增率计算
def equalIR_algorithm_3(pd):
    # 初始化功率参数
    p1, p2, p3 = sp.symbols('p_1 p_2 p_3')
    # 发电参数
    c1 = 0.226 * p1 ** 2 + 30.42 * p1 + 786.80
    c2 = 0.588 * p2 ** 2 + 65.12 * p2 + 451.32
    c3 = 0.785 * p3 ** 2 + 139.6 * p3 + 1049.50
    # 求导
    dc1 = sp.diff(c1, p1)
    dc2 = sp.diff(c2, p2)
    dc3 = sp.diff(c3, p3)
    # 组建方程组
    f1 = dc1 - dc2
    f2 = dc2 - dc3
    f3 = dc3 - dc1
    f4 = p1 + p2 + p3 - pd
    f5 = f1 - f2
    f6 = f2 - f3
    f7 = f3 - f1
    # 求解
    res = sp.solve({f4, f5, f6, f7})
    p1 = res[p1]
    p2 = res[p2]
    p3 = res[p3]
    return p1, p2, p3


# 双机组等微增率计算
def equalIR_algorithm_2(pd):
    # 初始化功率参数
    p1, p2 = sp.symbols('p_1 p_2')
    # 发电参数
    c1 = 0.226 * p1 ** 2 + 30.42 * p1 + 786.80
    c2 = 0.588 * p2 ** 2 + 65.12 * p2 + 451.32
    # 求导
    dc1 = sp.diff(c1, p1)
    dc2 = sp.diff(c2, p2)
    # 组建方程组
```

```
    f1 = dc1 - dc2
    f4 = p1 + p2 - pd
    # 求解
    res = sp.solve({f4, f1})
    p1 = res[p1]
    p2 = res[p2]
    return p1, p2


# 判断是否超出范围
def judge(p, up, down):
    if p > up:
        return up, 'up'
    elif p < down:
        return down, 'down'
    else:
        return p, 'ok'


# 单次求解发电最优解
def single_algorithm(pd, up1=600, down1=180, up2=300, down2=90, up3=150, down3=45):
    p1, p2, p3 = equalIR_algorithm_3(pd)
    p1, flag1 = judge(p1, up1, down1)
    p2, flag2 = judge(p2, up2, down2)
    p3, flag3 = judge(p3, up3, down3)
    if flag3 == 'down':
        p1, p2 = equalIR_algorithm_2(pd - down3)
        p1, flag1 = judge(p1, up1, down1)
        p2, flag2 = judge(p2, up2, down2)
        if flag2 == 'down':
            p1 = pd - down3 - down2
            p1, flag1 = judge(p1, up1, down1)
    # print('p1:', p1, 'p2:', p2, 'p3:', p3)
    return p1, p2, p3


# 追加写入文件
def write_file(pd, p1, p2, p3):
    with open('第一题：目标、机组1、2、3、差值.csv', 'a', encoding='utf-8') as f:
        dp = p1 + p2 + p3 - pd
        pd, p1, p2, p3, dp = round(pd, 3), round(p1, 3), round(p2, 3), round(p3, 3), round(dp, 3)
        pd, p1, p2, p3, dp = str(pd), str(p1), str(p2), str(p3), str(dp)
        f.write('{},{},{},{},{}\n'.format(pd, p1, p2, p3, dp))
        f.close()


# 画图
```

```python
def draw_pic(p1, p2, p3):
    from matplotlib.ticker import MultipleLocator

    plt.figure(figsize=(7, 3), dpi=800)
    ax = plt.subplot(111)
    data_wind = pd.read_excel('附件1.xlsx')
    data_wind = data_wind.dropna()
    time = data_wind['时间']

    time_new = []
    for i in range(0, len(time), 16):
        time_new.append(time[i])

    time_new.append('24:00:00')

    data_new = []
    for i in range(0, 96, 16):
        data_new.append(i)
    data_new.append(96)

    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.plot(p1, 'r-', label='机组1')
    plt.plot(p2, 'b-', label='机组2')
    plt.plot(p3, 'g-', label='机组3')
    plt.title('发电计划曲线')
    plt.xlabel('时间')
    plt.ylabel('发电量/MW')
    plt.legend()
    plt.grid()
    plt.tight_layout()
    xmajorLocator = MultipleLocator(16)
    ax.xaxis.set_major_locator(xmajorLocator)
    plt.xticks(data_new, time_new)

    plt.savefig('三台火电机组最优发电曲线.png')
    plt.show()


if __name__ == '__main__':
    # 删除文件
    import os

    if os.path.exists('第一题：目标、机组1、2、3、差值.csv'):
        os.remove('第一题：目标、机组1、2、3、差值.csv')
    f = open('第一题：目标、机组1、2、3、差值.csv', 'a', encoding='utf-8')
    f.write('目标功率,机组1,机组2,机组3,差值\n')
    f.close()
```

```
    data = pd.read_excel('附件1.xlsx')
    data = data.dropna()
    p1_list, p2_list, p3_list = [], [], []
    for i in data['l900']:
        p1, p2, p3 = single_algorithm(i)
        # write_file(i, p1, p2, p3)
        p1_list.append(p1)
        p2_list.append(p2)
        p3_list.append(p3)
    draw_pic(p1_list, p2_list, p3_list)

    for i in range(len(p1_list)):
        write_file(data['l900'][i], p1_list[i], p2_list[i], p3_list[i])
```

2.1.3 第一题：根据发电计划计算煤用量.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 15:29
# @File :  第一题：根据发电计划计算煤用量.py
# @IDE :  PyCharm

import numpy as np
import pandas as pd

import fireStationCost


# 煤费用
def coalCost(data_group_1, data_group_2, data_group_3):
    # 计算煤用量
    cost_1, cost_2, cost_3 = [], [], []
    for i in range(len(data_group_1)):
        cost_1.append(fireStationCost.fireStationRunCost(data_group_1[i], 0.226, 30.42, 786.80))
        cost_2.append(fireStationCost.fireStationRunCost(data_group_2[i], 0.588, 65.12, 451.32))
        cost_3.append(fireStationCost.fireStationRunCost(data_group_3[i], 0.785, 139.6, 1049.50))
    # 将结果转化为array数组
    cost_1, cost_2, cost_3 = np.array(cost_1), np.array(cost_2), np.array(cost_3)
    # 计算15min内的用煤量
    cost_1, cost_2, cost_3 = cost_1 * 0.25, cost_2 * 0.25, cost_3 * 0.25
    # 计算15min内的用煤费用（700元/吨）
    cost_1, cost_2, cost_3 = cost_1 * 0.7, cost_2 * 0.7, cost_3 * 0.7
    return cost_1, cost_2, cost_3
```

```python
# 碳费用
def carbonCost(data_group_1, data_group_2, data_group_3, carbon_price):
    cost_1 = fireStationCost.carbonEmission(data_group_1, 0.72) * carbon_price
    cost_2 = fireStationCost.carbonEmission(data_group_2, 0.75) * carbon_price
    cost_3 = fireStationCost.carbonEmission(data_group_3, 0.79) * carbon_price
    return cost_1, cost_2, cost_3


# 主函数
def main(carbon_price):
    data = pd.read_csv('第一题：目标、机组1、2、3、差值.csv')
    data_group_1 = data['机组1']
    data_group_2 = data['机组2']
    data_group_3 = data['机组3']
    data_load = data['目标功率']

    # 煤费用
    # 火电运行成本
    coal_cost_1, coal_cost_2, coal_cost_3 = coalCost(data_group_1, data_group_2, data_group_3)
    coal_cost_1 = sum(coal_cost_1)
    coal_cost_2 = sum(coal_cost_2)
    coal_cost_3 = sum(coal_cost_3)
    coalSum = coal_cost_1 + coal_cost_2 + coal_cost_3
    # 碳排放成本
    carbon_cost_1, carbon_cost_2, carbon_cost_3 = carbonCost(data_group_1, data_group_2,
        data_group_3, carbon_price)
    carbonSum = carbon_cost_1 + carbon_cost_2 + carbon_cost_3
    # 总成本
    total_cost = coalSum + carbonSum
    loadSum = sum(data_load) * 0.25
    # print(loadSum)
    print('当碳捕集成本为{}元/吨时，火电运行成本为{}万元，碳捕集成本为{}万元，'
          '总成本为{}万元，系统单位供电成本为{}元/KWh'.format(round(carbon_price, 3),
              round(coalSum / 10000, 3),
                                        round(carbonSum / 10000, 3),
                                        round(total_cost / 10000, 3),
                                        round(total_cost / loadSum / 1000, 3),
                                        3))


if __name__ == '__main__':
    carbonPrice = [0, 60, 80, 100]
    for i in carbonPrice:
        main(i)
```

2.2 第二题

2.2.1 第二题：弃风.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 10:33
# @File :  第二题：弃风.py
# @IDE :  PyCharm


import pandas as pd
import sympy as sp


def write_file(pd, p1, p2, pw, loss, time):
    with open('第二题：目标、机组1、2、风电、差值、弃风量.csv', 'a', encoding='utf-8') as f:
        # 清空文件内容
        f.truncate()
        dp = p1 + p2 + pw - pd
        pd, p1, p2, pw, loss = round(pd, 3), round(p1, 3), round(p2, 3), round(pw, 3),
            round(loss, 3)
        dp = round(dp, 3)
        pd, p1, p2, pw, dp = str(pd), str(p1), str(p2), str(pw), str(dp)
        f.write('{},{},{},{},{},{},{}\n'.format(time, pd, p1, p2, pw, dp, loss))
        f.close()


# 计算弃风量
def get_wind_loss(**kwargs):
    wind_loss = []
    for i in range(len(kwargs['data_wind_300'])):
        wind_loss_num = kwargs['data_wind_300'][i] + kwargs['p1'][i] + kwargs['p2'][i] -
            kwargs['data_load'][i]
        # print(i, ':', wind_loss_num)
        wind_loss.append(round(wind_loss_num, 3))
    return wind_loss


# 双机组等微增率
def equalIR_algorithm_2(pd):
    # 初始化功率参数
    p1, p2 = sp.symbols('p_1 p_2')
    # 发电参数
    c1 = 0.226 * p1 ** 2 + 30.42 * p1 + 786.80
    c2 = 0.588 * p2 ** 2 + 65.12 * p2 + 451.32
    # 求导
    dc1 = sp.diff(c1, p1)
```

```
    dc2 = sp.diff(c2, p2)
    # 组建方程组
    f1 = dc1 - dc2
    f4 = p1 + p2 - pd
    # 求解
    res = sp.solve({f4, f1})
    p1 = res[p1]
    p2 = res[p2]
    return p1, p2


# 判断是否超出范围
def judge(p, up, down):
    if p > up:
        return up, 'up'
    elif p < down:
        return down, 'down'
    else:
        return p, 'ok'


# 单次求解发电最优解
def single_algorithm(pd, up1=600, down1=180, up2=300, down2=90):
    p1, p2 = equalIR_algorithm_2(pd)
    p1, flag1 = judge(p1, up1, down1)
    p2, flag2 = judge(p2, up2, down2)
    if flag2 == 'down':
        p1 = pd - down2
        p1, flag1 = judge(p1, up1, down1)
        return p1, p2
    if flag1 == 'up':
        p2 = pd - up1
        p2, flag2 = judge(p2, up2, down2)
        return p1, p2
    return p1, p2


def get_wind_load(load_loss):
    loss_wind_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] > 0:
            loss_wind_sum += load_loss[i]
    return loss_wind_sum


# 读取数据
def read_data():
```

```
    data_wind = pd.read_excel('附件1.xlsx')
    data_wind = data_wind.dropna()
    time = data_wind['时间']
    data_load = data_wind['l900']
    data_wind_300 = data_wind['w300']
    return data_load, data_wind_300, time


if __name__ == '__main__':
    # 删除文件
    import os

    if os.path.exists('第二题：目标、机组1、2、风电、差值、弃风量.csv'):
        os.remove('第二题：目标、机组1、2、风电、差值、弃风量.csv')
    f = open('第二题：目标、机组1、2、风电、差值、弃风量.csv', 'a',encoding='utf-8')
    f.write('时间,目标,机组1,机组2,风电,差值,弃风量\n')
    f.close()
    # 读取数据
    data_load, data_wind_300, time = read_data()

    # 计算除风电外的机组等微增率
    p1_list, p2_list = [], []
    for i in range(len(data_load)):
        p1, p2 = single_algorithm(data_load[i] - data_wind_300[i])
        # print(p1, p2)
        # 写入文件
        # write_file(data_load[i], p1, p2, data_wind_300[i])
        p1_list.append(p1)
        p2_list.append(p2)

    # 计算弃风量
    wind_loss = get_wind_loss(data_wind_300=data_wind_300, p1=p1_list, p2=p2_list,
        data_load=data_load)
    # print(wind_loss)

    # 写入文件
    for i in range(len(wind_loss)):
        write_file(data_load[i], p1_list[i], p2_list[i], data_wind_300[i], wind_loss[i], time[i])

    # 取出弃风量的最大值
    max_wind_loss = max(wind_loss)
    # print(max_wind_loss)

    # 计算可以减少的风电装机容量
    wind_power_min = 300 - max_wind_loss
    print('可减少的风电装机容量为：{}MW，即最小风电装机容量为：{}MW'.format(max_wind_loss,
        wind_power_min))
```

```
    # 计算弃风总量
    wind_loss_all = get_wind_load(wind_loss) * 0.25
    print('弃风电量为：{}MWh'.format(wind_loss_all))
```

2.2.2 第二题：功率平衡绘图.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/28 14:48
# @File :  第二题：功率平衡绘图.py
# @IDE :  PyCharm

import matplotlib.pyplot as plt
import pandas as pd
from matplotlib.ticker import MultipleLocator

# 读取数据
df = pd.read_csv('第二题：目标、机组1、2、风电、差值、弃风量.csv')
data = df['差值']
time = df['时间']
print(time)

# 每隔16个数取出time
time_new = []
for i in range(0, len(time), 16):
    time_new.append(time[i])

time_new.append('24:00:00')
# print(time_new)
# 生成0-96的数据
data_new = []
for i in range(0, 96, 16):
    data_new.append(i)

data_new.append(96)
# print(data_new)

# 绘图
plt.figure(figsize=(7, 3), dpi=800)
ax = plt.subplot(111)

plt.plot(data, color='blue', linewidth=1.5, linestyle='-')
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.title('功率平衡图')
```

```
plt.xlabel('时间')
plt.ylabel('功率/MW')
plt.tight_layout()
plt.grid(True)
xmajorLocator = MultipleLocator(16)
ax.xaxis.set_major_locator(xmajorLocator)
plt.xticks(data_new,time_new)
plt.savefig('第二题：功率平衡图.png')

plt.show()
```

2.3 第三题

2.3.1 第三题：弃风.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 10:33
# @File :  第二题：弃风.py
# @IDE :  PyCharm


import os

import pandas as pd
import sympy as sp


def write_file(pd, p1, p2, pw, loss, time):
    with open('第三题：目标、机组1、3、风电、差值、弃风量.csv', 'a', encoding='utf-8') as f:
        # 清空文件内容
        f.truncate()
        dp = p1 + p2 + pw - pd
        pd, p1, p2, pw, loss = round(pd, 3), round(p1, 3), round(p2, 3), round(pw, 3),
            round(loss, 3)
        dp = round(dp, 3)
        pd, p1, p2, pw, dp = str(pd), str(p1), str(p2), str(pw), str(dp)
        f.write('{},{},{},{},{},{},{}\n'.format(time, pd, p1, p2, pw, dp, loss))
        f.close()


# 计算弃风量
def get_wind_loss(**kwargs):
    wind_loss = []
    for i in range(len(kwargs['data_wind_600'])):
        wind_loss_num = kwargs['data_wind_600'][i] + kwargs['p1'][i] + kwargs['p2'][i] -
```

```
            kwargs['data_load'][i]
        # print(i, ':', wind_loss_num)
        wind_loss.append(round(wind_loss_num, 3))
    return wind_loss


def get_wind_load(load_loss):
    loss_wind_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] > 0:
            loss_wind_sum += load_loss[i]
    return loss_wind_sum


# 双机组等微增率
def equalIR_algorithm_2(pd):
    # 初始化功率参数
    p1, p3 = sp.symbols('p_1 p_2')
    # 发电参数
    c1 = 0.226 * p1 ** 2 + 30.42 * p1 + 786.80
    c2 = 0.785 * p3 ** 2 + 139.6 * p3 + 1049.50
    # 求导
    dc1 = sp.diff(c1, p1)
    dc2 = sp.diff(c2, p3)
    # 组建方程组
    f1 = dc1 - dc2
    f4 = p1 + p3 - pd
    # 求解
    res = sp.solve({f4, f1})
    p1 = res[p1]
    p3 = res[p3]
    return p1, p3


# 判断是否超出范围
def judge(p, up, down):
    if p > up:
        return up, 'up'
    elif p < down:
        return down, 'down'
    else:
        return p, 'ok'


# 单次求解发电最优解
def single_algorithm(pd, up1=600, down1=180, up3=150, down3=45):
    p1, p2 = equalIR_algorithm_2(pd)
```

```
    p1, flag1 = judge(p1, up1, down1)
    p2, flag2 = judge(p2, up3, down3)
    if flag2 == 'down':
        p1 = pd - down3
        p1, flag1 = judge(p1, up1, down1)
        return p1, p2
    if flag1 == 'up':
        p2 = pd - up1
        p2, flag2 = judge(p2, up3, down3)
        return p1, p2
    return p1, p2


# 读取数据
def read_data():
    data_wind = pd.read_excel('附件1.xlsx')
    data_wind = data_wind.dropna()
    # print(data_wind)
    data_wind_600 = data_wind['w600 ']
    time = data_wind['时间']
    data_load = data_wind['l900']
    return data_wind_600, time, data_load


if __name__ == '__main__':
    data_wind_600, time, data_load = read_data()

    if os.path.exists('第三题：目标、机组1、3、风电、差值、弃风量.csv'):
        os.remove('第三题：目标、机组1、3、风电、差值、弃风量.csv')
    f = open('第三题：目标、机组1、3、风电、差值、弃风量.csv', 'a', encoding='utf-8')
    f.write('时间,目标,机组1,机组3,风电,差值,弃风量\n')
    f.close()

    p1_list, p2_list = [], []
    for i in range(len(data_load)):
        p1, p2 = single_algorithm(data_load[i] - data_wind_600[i])
        # print(p1, p2)
        # 写入文件
        # write_file(data_load[i], p1, p2, data_wind_300[i])
        p1_list.append(p1)
        p2_list.append(p2)

    # 计算弃风量
    wind_loss = get_wind_loss(data_wind_600=data_wind_600, p1=p1_list, p2=p2_list,
        data_load=data_load)
    # print(wind_loss)
    # 写入文件
```

```
    for i in range(len(wind_loss)):
        write_file(data_load[i], p1_list[i], p2_list[i], data_wind_600[i], wind_loss[i], time[i])

    # 取出弃风量的最大值
    min_wind_loss = min(wind_loss)
    # print(min_wind_loss)

    # 计算可以减少的风电装机容量
    wind_power_min = 600 - min_wind_loss
    print('缺额的风电装机容量为：{}MW，即需要增加接入容量为：{}MW'.format(abs(min_wind_loss),
        round(abs(min_wind_loss), 3)))

    # 计算弃风总量
    wind_loss_all = get_wind_load(wind_loss) * 0.25
    print('弃风电量为：{}MWh'.format(wind_loss_all))
```

2.3.2 第三题：功率平衡绘图.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/28 14:48
# @File :  第二题：功率平衡绘图.py
# @IDE :  PyCharm

import matplotlib.pyplot as plt
import pandas as pd
from matplotlib.ticker import MultipleLocator

# 读取数据
df = pd.read_csv('第三题：目标、机组1、3、风电、差值、弃风量.csv')
data = df['差值']
time = df['时间']
print(time)

# 每隔16个数取出time
time_new = []
for i in range(0, len(time), 16):
    time_new.append(time[i])

time_new.append('24:00:00')
# print(time_new)
# 生成0-96的数据
data_new = []
for i in range(0, 96, 16):
    data_new.append(i)
```

```
data_new.append(96)
# print(data_new)

# 绘图
plt.figure(figsize=(7, 3), dpi=800)
ax = plt.subplot(111)

plt.plot(data, color='blue', linewidth=1.5, linestyle='-')
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.title('功率平衡图')
plt.xlabel('时间')
plt.ylabel('功率/MW')
plt.tight_layout()
plt.grid(True)
xmajorLocator = MultipleLocator(16)
ax.xaxis.set_major_locator(xmajorLocator)
plt.xticks(data_new,time_new)
plt.savefig('第三题：功率平衡图.png')

plt.show()
```

2.4 第四题

2.4.1 300MW 风电替代机组 3

fireStationCost.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 17:13
# @File :  火电运行成本.py
# @IDE :  PyCharm

################
# 煤用量
# 火电烧煤运行成本
def fireStationCoalCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    return p ** 2 * a + p * b + c
```

```
# 火电煤维护成本
def fireStationMaintenanceCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    coalCost = fireStationCoalCost(p, a, b, c)
    return coalCost * 0.5


# 单次火电煤运行成本
def fireStationRunCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    coalCost = fireStationCoalCost(p, a, b, c)
    maintenanceCost = fireStationMaintenanceCost(p, a, b, c)
    return coalCost + maintenanceCost


###############
# 碳费用
# 总发电量
def totalGeneration(powerPlan):
    '''
    :param powerPlan: 电力计划
    :return:
    '''
    return sum(powerPlan) * 0.25


# 碳排放量
def carbonEmission(powerPlan, carbonWeight):
    '''
    :param powerPlan: 电力计划
    :param carbonWeight: 碳排放参数
    :return:
    '''
    return totalGeneration(powerPlan) * carbonWeight
```

第四题：计算费用-300MW 替代.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 15:29
# @File :  第四题：计算费用-300MW替代.py
# @IDE :  PyCharm

import csv

import numpy as np
import pandas as pd

import fireStationCost


# 煤费用
def coalCost(data_group_1, data_group_2):
    # 计算煤用量
    cost_1, cost_2 = [], []
    for i in range(len(data_group_1)):
        cost_1.append(fireStationCost.fireStationRunCost(data_group_1[i],
                                                         0.226, 30.42, 786.80))
        cost_2.append(fireStationCost.fireStationRunCost(data_group_2[i],
                                                         0.588, 65.12, 451.32))
    # 将结果转化为array数组
    cost_1, cost_2 = np.array(cost_1), np.array(cost_2)
    # 计算15min内的用煤量
    cost_1, cost_2 = cost_1 * 0.25, cost_2 * 0.25
    # 计算15min内的用煤费用（700元/吨）
    cost_1, cost_2 = cost_1 * 0.7, cost_2 * 0.7
    return cost_1, cost_2


# 碳费用
def carbonCost(data_group_1, data_group_2, carbon_price):
    cost_1 = fireStationCost.carbonEmission(data_group_1, 0.72) * carbon_price
    cost_2 = fireStationCost.carbonEmission(data_group_2, 0.75) * carbon_price
    return cost_1, cost_2


# 计算弃风量
def get_wind_load(load_loss):
    loss_wind_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] > 0:
```

```
            loss_wind_sum += load_loss[i]
    return loss_wind_sum


# 主函数
def main(carbon_price):
    data = pd.read_csv('第二题：目标、机组1、2、风电、差值、弃风量.csv')
    data_group_1 = data['机组1']
    data_group_2 = data['机组2']
    data_wind_loss = data['弃风量']
    data_wind = data['风电']
    data_load = data['目标']

    # 计算弃风量
    loss_wind_sum = get_wind_load(data_wind_loss) * 0.25
    # 计算弃风量损失
    loss_wind_cost = loss_wind_sum * 0.3 * 1000

    # 计算风电运维成本
    wind_sum = sum(data_wind) * 0.25
    wind_sum = wind_sum * 0.045 * 1000
    # 煤费用
    # 火电运行成本
    coal_cost_1, coal_cost_2 = coalCost(data_group_1, data_group_2)
    coal_cost_1 = sum(coal_cost_1)
    coal_cost_2 = sum(coal_cost_2)
    coalSum = coal_cost_1 + coal_cost_2
    # 碳排放成本
    carbon_cost_1, carbon_cost_2 = carbonCost(data_group_1, data_group_2, carbon_price)
    carbonSum = carbon_cost_1 + carbon_cost_2
    # 总成本
    total_cost = coalSum + carbonSum + loss_wind_cost + wind_sum
    loadSum = sum(data_load) * 0.25
    # print(loadSum)
    print('当碳捕集成本为{}元/吨时，火电运行成本为{}万元，碳捕集成本为{}万元，风电运维成本为{}万元，'
          '弃风电量为{}MWh，弃风损失为{}万元，系统单位供电成本为{}元/KWh'.format(
        round(carbon_price, 3),
        round(coalSum / 10000, 3),
        round(carbonSum / 10000, 3),
        round(wind_sum / 10000, 3),
        round(loss_wind_sum, 3),
        round(loss_wind_cost / 10000, 3),
        round(total_cost / loadSum / 1000, 3),
        3))
    # 写入csv
    with open('第四题：计算费用-300MW替代.csv', 'a', newline='') as f:
        writer = csv.writer(f)
```

```
        writer.writerow([carbon_price, coalSum / 10000, carbonSum / 10000, wind_sum / 10000,
                         loss_wind_sum, loss_wind_cost / 10000, total_cost / loadSum / 1000])


if __name__ == '__main__':
    carbonPrice = [0, 60, 80, 100]
    for i in carbonPrice:
        main(i)
```

2.4.2 600MW 风电替代机组 2

fireStationCost.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 17:13
# @File :  火电运行成本.py
# @IDE :  PyCharm

#################
# 煤用量
# 火电烧煤运行成本
def fireStationCoalCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    return p ** 2 * a + p * b + c


# 火电煤维护成本
def fireStationMaintenanceCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    coalCost = fireStationCoalCost(p, a, b, c)
    return coalCost * 0.5


# 单次火电煤运行成本
```

```
def fireStationRunCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    coalCost = fireStationCoalCost(p, a, b, c)
    maintenanceCost = fireStationMaintenanceCost(p, a, b, c)
    return coalCost + maintenanceCost


##############
# 碳费用
# 总发电量
def totalGeneration(powerPlan):
    '''
    :param powerPlan: 电力计划
    :return:
    '''
    return sum(powerPlan) * 0.25


# 碳排放量
def carbonEmission(powerPlan, carbonWeight):
    '''
    :param powerPlan: 电力计划
    :param carbonWeight: 碳排放参数
    :return:
    '''
    return totalGeneration(powerPlan) * carbonWeight
```

第四题：计算费用-600MW 替代.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 15:29
# @File :  第四题：计算费用-600MW替代.py
# @IDE :  PyCharm

import csv

import numpy as np
import pandas as pd

import fireStationCost
```

```
# 煤费用
def coalCost(data_group_1, data_group_2):
    # 计算煤用量
    cost_1, cost_2 = [], []
    for i in range(len(data_group_1)):
        cost_1.append(fireStationCost.fireStationRunCost(data_group_1[i],
                                                         0.226, 30.42, 786.80))
        cost_2.append(fireStationCost.fireStationRunCost(data_group_2[i],
                                                         0.588, 65.12, 451.32))
    # 将结果转化为array数组
    cost_1, cost_2 = np.array(cost_1), np.array(cost_2)
    # 计算15min内的用煤量
    cost_1, cost_2 = cost_1 * 0.25, cost_2 * 0.25
    # 计算15min内的用煤费用（700元/吨）
    cost_1, cost_2 = cost_1 * 0.7, cost_2 * 0.7
    return cost_1, cost_2


# 碳费用
def carbonCost(data_group_1, data_group_2, carbon_price):
    cost_1 = fireStationCost.carbonEmission(data_group_1, 0.72) * carbon_price
    cost_2 = fireStationCost.carbonEmission(data_group_2, 0.75) * carbon_price
    return cost_1, cost_2


# 计算弃风量
def get_wind_load(load_loss):
    loss_wind_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] > 0:
            loss_wind_sum += load_loss[i]
    return loss_wind_sum


# 计算丢负荷量
def get_loss_load(load_loss):
    loss_load_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] < 0:
            loss_load_sum += load_loss[i]
    return loss_load_sum


# 主函数
def main(carbon_price):
    data = pd.read_csv('第三题：目标、机组1、3、风电、差值、弃风量.csv')
```

```
data_group_1 = data['机组1']
data_group_2 = data['机组3']
data_wind_loss = data['弃风量']
data_wind = data['风电']
data_load = data['目标']

# 计算弃风量
loss_wind_sum = get_wind_load(data_wind_loss) * 0.25
# 计算弃风量损失
loss_wind_cost = loss_wind_sum * 0.3 * 1000

# 计算风电运维成本
wind_sum = sum(data_wind) * 0.25
wind_sum = wind_sum * 0.045 * 1000

# 计算丢负荷量
loss_load_sum = get_loss_load(data_wind_loss) * 0.25
# 取绝对值
loss_load_sum = abs(loss_load_sum)
# 计算丢负荷量损失
loss_load_cost = loss_load_sum * 8 * 1000

# 煤费用
# 火电运行成本
coal_cost_1, coal_cost_2 = coalCost(data_group_1, data_group_2)
coal_cost_1 = sum(coal_cost_1)
coal_cost_2 = sum(coal_cost_2)
coalSum = coal_cost_1 + coal_cost_2

# 碳排放成本
carbon_cost_1, carbon_cost_2 = carbonCost(data_group_1, data_group_2, carbon_price)
carbonSum = carbon_cost_1 + carbon_cost_2

# 总成本
total_cost = coalSum + carbonSum + loss_wind_cost + wind_sum + loss_load_cost
loadSum = sum(data_load) * 0.25
# print(loadSum)
print(
    '当碳捕集成本为{}元/吨时，火电运行成本为{}万元，碳捕集成本为{}万元，'
    '风电运维成本为{}万元，弃风电量为{}MWh，弃风损失为{}万元,丢负荷量为{}MW，'
    '负荷损失为{}万元，系统单位供电成本为{}元/KWh'.format(
        round(carbon_price, 3),
        round(coalSum / 10000, 3),
        round(carbonSum / 10000, 3),
        round(wind_sum / 10000, 3),
        round(loss_wind_sum, 3),
        round(loss_wind_cost / 10000, 3),
```

```
            round(loss_load_sum, 3),
            round(loss_load_cost / 10000, 3),
            round(total_cost / loadSum / 1000, 3),
            3))
    # 写入csv
    with open('第四题：计算费用-600MW替代.csv', 'a', newline='') as f:
        writer = csv.writer(f)
        writer.writerow(
            [carbon_price, coalSum / 10000, carbonSum / 10000,
             wind_sum / 10000, loss_wind_sum, loss_wind_cost / 10000,
             loss_load_sum, loss_load_cost / 10000,
             total_cost / loadSum / 1000])


if __name__ == '__main__':
    carbonPrice = [0, 60, 80, 100]
    for i in carbonPrice:
        main(i)
```

2.5 第五题

2.5.1 fireStationCost.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 17:13
# @File :  火电运行成本.py
# @IDE :  PyCharm

##################
# 煤用量
# 火电烧煤运行成本
def fireStationCoalCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    return p ** 2 * a + p * b + c


# 火电煤维护成本
def fireStationMaintenanceCost(p, a, b, c):
    '''
```

```
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    coalCost = fireStationCoalCost(p, a, b, c)
    return coalCost * 0.5


# 单次火电煤运行成本
def fireStationRunCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    coalCost = fireStationCoalCost(p, a, b, c)
    maintenanceCost = fireStationMaintenanceCost(p, a, b, c)
    return coalCost + maintenanceCost


#############
# 碳费用
# 总发电量
def totalGeneration(powerPlan):
    '''
    :param powerPlan: 电力计划
    :return:
    '''
    return sum(powerPlan) * 0.25


# 碳排放量
def carbonEmission(powerPlan, carbonWeight):
    '''
    :param powerPlan: 电力计划
    :param carbonWeight: 碳排放参数
    :return:
    '''
    return totalGeneration(powerPlan) * carbonWeight
```

2.5.2 第五题：失负荷量.py

```python
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 10:33
# @File :  第二题：弃风.py
# @IDE :  PyCharm
import numpy as np
import pandas as pd

import fireStationCost


def write_file(pd):
    with open('第五题：失负荷量.csv', 'a') as f:
        pd = round(pd, 3)
        pd = str(pd)
        f.write('{}\n'.format(pd))
        f.close()


# 计算弃风量
def get_wind_loss(**kwargs):
    wind_loss = []
    for i in range(len(kwargs['data_wind_300'])):
        wind_loss_num = -kwargs['target_fire'][i] + kwargs['fireStation1'][i] +
            kwargs['data_wind_300'][i]
        wind_loss.append(round(wind_loss_num, 3))
    return wind_loss


# 计算总失负荷量
def get_loss_load(load_loss):
    loss_load_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] < 0:
            loss_load_sum += load_loss[i]
    return -loss_load_sum


# 判断是否超出范围
def judge(p, up, down):
    if p > up:
        return up, 'up'
    elif p < down:
        return down, 'down'
    else:
        return p, 'ok'
```

```
def get_wind_load(load_loss):
    loss_wind_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] > 0:
            loss_wind_sum += load_loss[i]
    return loss_wind_sum


# 煤费用
def coalCost(data_group_1):
    # 计算煤用量
    cost_1 = []
    for i in range(len(data_group_1)):
        cost_1.append(fireStationCost.fireStationRunCost(data_group_1[i], 0.226, 30.42, 786.80))
    # 将结果转化为array数组
    cost_1 = np.array(cost_1)
    # 计算15min内的用煤量
    cost_1 = cost_1 * 0.25
    # 计算15min内的用煤费用（700元/吨）
    cost_1 = cost_1 * 0.7
    return cost_1


# 碳费用
def carbonCost(data_group_1, carbon_price):
    cost_1 = fireStationCost.carbonEmission(data_group_1, 0.72) * carbon_price
    return cost_1


if __name__ == '__main__':
    # 读取风电场数据
    data_wind = pd.read_excel('附件1.xlsx')
    data_wind = data_wind.dropna()
    data_wind_900 = data_wind['w900']
    data_target = data_wind['l900']

    # 转换为array
    data_wind_900, data_target = np.array(data_wind_900), np.array(data_target)
    # 计算机组1功率
    fireStation = data_target - data_wind_900
    # 判断机组1功率是否超出范围
    fireStation1 = []
    for i in range(len(fireStation)):
        p, judge_result = judge(fireStation[i], 600, 180)
        fireStation1.append(p)
```

```
# 计算弃风量
load_loss = get_wind_loss(data_wind_300=data_wind_900, fireStation1=fireStation1,
                          target_fire=data_target)
# print(load_loss)

# 计算总失负荷量
loss_load_sum = get_loss_load(load_loss) * 0.25
# 计算丢负荷损失
loss_load_cost = loss_load_sum * 8 * 1000
# print(loss_load_sum)

#####计算储能成本
# 计算单次最大丢负荷量
loss_load_max = min(load_loss)
# 取绝对值
loss_load_max = abs(loss_load_max)
# 所需储能容量
storage_capacity = loss_load_max / 0.9
## 储能投资成本
# 单位功率成本
unit_cost = storage_capacity * 3000 * 1000
# 单位能量成本
unit_energy_cost = storage_capacity * 0.25 * 3000 * 1000
## 储能运维成本
storage_capacity_cost = loss_load_sum * 0.05 * 1000
# 总储能成本
storage_cost = (unit_cost + unit_energy_cost) / 10 / 365 + storage_capacity_cost

# 计算弃风量
loss_wind_sum = get_wind_load(load_loss) * 0.25
# 计算弃风损失
loss_wind_cost = loss_wind_sum * 0.3 * 1000

# 煤费用
cost_coal = coalCost(data_group_1=fireStation1)
coalSum = sum(cost_coal)

# 碳排放费用
carbon_price = 60
cost_carbon = carbonCost(data_group_1=fireStation1, carbon_price=carbon_price)

# 总成本
total_cost = loss_wind_cost + coalSum + cost_carbon + storage_cost
loadSum = sum(data_target) * 0.25

print('风电装机900MW、替代机组2、3时，丢负荷电量{}MWh，最大失负荷功率为{}MW，需要配备最小储能容量为{}MWh。考虑碳
    round(loss_load_sum, 3),
```

```
        round(loss_load_max, 3),
        round(storage_capacity, 3),
        round(total_cost / loadSum / 1000, 3)))
```

2.6 第七题

2.6.1 fireStationCost.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 17:13
# @File :  火电运行成本.py
# @IDE :  PyCharm

###############
# 煤用量
# 火电烧煤运行成本
def fireStationCoalCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    return p ** 2 * a + p * b + c


# 火电煤维护成本
def fireStationMaintenanceCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    coalCost = fireStationCoalCost(p, a, b, c)
    return coalCost * 0.5


# 单次火电煤运行成本
def fireStationRunCost(p, a, b, c):
    '''
    :param p: 功率
    :param a: 煤用量参数1
```

```
    :param b: 煤用量参数2
    :param c: 煤用量参数3
    :return:
    '''
    coalCost = fireStationCoalCost(p, a, b, c)
    maintenanceCost = fireStationMaintenanceCost(p, a, b, c)
    return coalCost + maintenanceCost


############
# 碳费用
# 总发电量
def totalGeneration(powerPlan):
    '''
    :param powerPlan: 电力计划
    :return:
    '''
    return sum(powerPlan) * 0.25


# 碳排放量
def carbonEmission(powerPlan, carbonWeight):
    '''
    :param powerPlan: 电力计划
    :param carbonWeight: 碳排放参数
    :return:
    '''
    return totalGeneration(powerPlan) * carbonWeight
```

2.6.2 q7singleRun.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 10:33
# @File :  第二题：弃风.py
# @IDE :  PyCharm
import numpy as np
import pandas as pd

import fireStationCost


def write_file(pd):
    with open('第七题：失负荷量.csv', 'a') as f:
        pd = round(pd, 3)
        pd = str(pd)
```

```
        f.write('{}\n'.format(pd))
        f.close()


# 计算弃风量
def get_wind_loss(**kwargs):
    wind_loss = []
    for i in range(len(kwargs['data_wind_300'])):
        wind_loss_num = -kwargs['target_fire'][i] + kwargs['fireStation1'][i]\
                    + kwargs['data_wind_300'][i]
        wind_loss.append(round(wind_loss_num, 3))
    return wind_loss


# 计算总失负荷量
def get_loss_load(load_loss):
    loss_load_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] < 0:
            loss_load_sum += load_loss[i]
    return -loss_load_sum


# 判断是否超出范围
def judge(p, up, down):
    if p > up:
        return up, 'up'
    elif p < down:
        return down, 'down'
    else:
        return p, 'ok'


def get_wind_load(load_loss):
    loss_wind_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] > 0:
            loss_wind_sum += load_loss[i]
    return loss_wind_sum


# 煤费用
def coalCost(data_group_1):
    # 计算煤用量
    cost_1 = []
    for i in range(len(data_group_1)):
        cost_1.append(fireStationCost.fireStationRunCost(data_group_1[i],
```

```
                                                0.226, 30.42, 786.80))
    # 将结果转化为array数组
    cost_1 = np.array(cost_1)
    # 计算15min内的用煤量
    cost_1 = cost_1 * 0.25
    # 计算15min内的用煤费用（700元/吨）
    cost_1 = cost_1 * 0.7
    return cost_1


# 碳费用
def carbonCost(data_group_1, carbon_price):
    cost_1 = fireStationCost.carbonEmission(data_group_1, 0.72) * carbon_price
    return cost_1


def singleRun(storage_capacity, power_capacity, carbon_price=0):
    # 读取风电场数据
    data_wind = pd.read_excel('附件2.xlsx')
    data_wind = data_wind.dropna()
    data_wind_900 = data_wind['风电功率(MW)']
    data_target = data_wind['负荷功率(MW)']
    # 转换为array
    data_wind_900, data_target = np.array(data_wind_900), np.array(data_target)
    # 计算机组1功率
    fireStation = data_target - data_wind_900
    # 判断机组1功率是否超出范围
    fireStation1 = []
    for i in range(len(fireStation)):
        p, judge_result = judge(fireStation[i], 600, 180)
        fireStation1.append(p)
    # 计算弃风量
    load_loss = get_wind_loss(data_wind_300=data_wind_900, fireStation1=fireStation1,
                              target_fire=data_target)
    # print(load_loss)
    # 计算总失负荷量
    loss_load_sum = get_loss_load(load_loss) * 0.25
    # 计算丢负荷损失
    loss_load_cost = loss_load_sum * 8 * 1000
    # print(loss_load_sum)
    #####计算储能成本
    # 计算单次最大丢负荷量
    loss_load_max = min(load_loss)
    # 取绝对值
    loss_load_max = abs(loss_load_max)
    # 所需储能容量
    # storage_capacity = loss_load_max / 0.9
```

```
    ## 储能投资成本
    # 单位功率成本
    unit_cost = storage_capacity * 3000 * 1000
    # 单位能量成本
    unit_energy_cost = power_capacity * 0.25 * 3000 * 1000
    ## 储能运维成本
    storage_capacity_cost = loss_load_sum * 0.05 * 1000
    # 总储能成本
    storage_cost = (unit_cost + unit_energy_cost) / 10 / 365 + storage_capacity_cost
    # 计算弃风量
    loss_wind_sum = get_wind_load(load_loss) * 0.25
    # 计算弃风损失
    loss_wind_cost = loss_wind_sum * 0.3 * 1000
    # 煤费用
    cost_coal = coalCost(data_group_1=fireStation1)
    coalSum = sum(cost_coal)
    # 碳排放费用

    cost_carbon = carbonCost(data_group_1=fireStation1, carbon_price=carbon_price)
    # 总成本
    total_cost = loss_wind_cost + coalSum + cost_carbon + storage_cost
    loadSum = sum(data_target) * 0.25
    return total_cost / loadSum / 1000


if __name__ == '__main__':
    print(1)
```

2.6.3 第七题：功率平衡绘图.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 21:35
# @File :  第七题：弃风量和失负荷量.py
# @IDE :  PyCharm

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator

###################################
# 读取数据并处理
# 将数组转换为列表
def list_data(data):
    data_list = data.tolist()
```

```
    return data_list


# 数据按96个时间点分组
def group_data(data_load_list, data_wind_list, data_time_list):
    data_load_group = []
    data_wind_group = []
    data_time_group = []
    for i in range(0, len(data_load_list), 96):
        data_load_group.append(data_load_list[i:i + 96])
        data_wind_group.append(data_wind_list[i:i + 96])
        data_time_group.append(data_time_list[i:i + 96])
    return data_load_group, data_wind_group, data_time_group


# 将时间戳转换为字符串
def time_to_str(data_time_group):
    data_time_str = []
    for i in range(len(data_time_group)):
        data_time_str.append(str(data_time_group[i]))
    return data_time_str


# 读取数据
def read_data():
    data = pd.read_excel('附件2.xlsx')
    data_load = data['负荷功率(MW)']
    data_wind = data['风电功率(MW)']
    data_time = data['时间']

    data_load_list = list_data(data_load)
    data_wind_list = list_data(data_wind)
    data_time_list = list_data(data_time)

    data_time_list = time_to_str(data_time_list)

    data_load_group, data_wind_group, data_time_group = group_data(data_load_list,
                                                                   data_wind_list,
                                                                   data_time_list)
    data_load_group, data_wind_group, data_time_group = np.array(data_load_group), \
                                                        np.array(data_wind_group),\
                                                        np.array(
        data_time_group)
    return data_load_group, data_wind_group, data_time_group


##########################
```

```
##弃风总量和丢负荷总量

# 计算弃风量
def get_wind_load(load_loss):
    loss_wind_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] > 0:
            loss_wind_sum += load_loss[i]
    return loss_wind_sum


# 计算丢负荷量
def get_loss_load(load_loss):
    loss_load_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] < 0:
            loss_load_sum += load_loss[i]
    return loss_load_sum


############################
## 计算弃风量

def get_wind_loss(**kwargs):
    wind_loss = []
    for i in range(len(kwargs['data_wind_300'])):
        wind_loss_num = -kwargs['target_fire'][i] + kwargs['fireStation1'][i]\
                    + kwargs['data_wind_300'][i]
        wind_loss.append(round(wind_loss_num, 3))
    return wind_loss


################################
## 判断范围
# 判断是否超出范围
def judge(p, up, down):
    if p > up:
        return up, 'up'
    elif p < down:
        return down, 'down'
    else:
        return p, 'ok'


if __name__ == '__main__':
    data_load, data_wind, data_time = read_data()
```

```
# 计算火电功率
data_fire_raw = data_load - data_wind
data_fire = []
# 判断火电功率范围
for i in range(len(data_fire_raw)):
    data_fire.append([])
    for j in range(len(data_fire_raw[i])):
        data_fire_temp, flag = judge(data_fire_raw[i][j], 600, 180)
        data_fire[i].append(data_fire_temp)
        # print(data_fire[0])

# 计算弃风量
data_wind_loss = []
for i in range(len(data_fire)):
    data_wind_loss.append(
        get_wind_loss(data_wind_300=data_wind[i],
                      target_fire=data_load[i], fireStation1=data_fire[i]))
# print(data_wind_loss[0])

###取出最值
# 每日最大值
data_wind_loss_max_day = np.array(data_wind_loss).max(axis=1)
# print(data_wind_loss_max_day)
# 每日最小值
data_wind_loss_min_day = np.array(data_wind_loss).min(axis=1)
# print(data_wind_loss_min_day)
# 总最大值
data_wind_loss_max_sum = np.array(data_wind_loss).max()
# print(data_wind_loss_max_sum)
# 总最小值
data_wind_loss_min_sum = np.array(data_wind_loss).min()
# print(data_wind_loss_min_sum)

###计算弃风总量和丢负荷总量
# 计算弃风总量和丢负荷总量
loss_wind_sum, loss_load_sum = [], []
for i in range(len(data_wind_loss)):
    loss_wind_sum.append(get_wind_load(data_wind_loss[i]))
    loss_load_sum.append(get_loss_load(data_wind_loss[i]))
# print(loss_wind_sum)
# print(loss_load_sum)

data_wind_loss=np.array(data_wind_loss)
#展开成一维
data_wind_loss_1d=data_wind_loss.flatten()

str_base='7.'
```

```
    time_new=[]
    for i in range(16):
        time_new.append(str_base+str(i+1))
    data_new = []
    for i in range(0, 96*16, 96):
        data_new.append(i)

    plt.figure(figsize=(7, 3), dpi=800)
    ax = plt.subplot(111)

    plt.plot(data_wind_loss_1d, color='blue', linewidth=1.5, linestyle='-')
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.title('功率平衡图')
    plt.xlabel('时间')
    plt.ylabel('功率/MW')
    plt.grid(True)
    xmajorLocator = MultipleLocator(96)
    ax.xaxis.set_major_locator(xmajorLocator)
    plt.xticks(data_new, time_new)
    #旋转x轴刻度
    plt.xticks(rotation=90)

    plt.tight_layout()
    plt.savefig('第七题：功率平衡图.png')
    plt.show()
```

2.6.4 第七题：功率平衡绘图.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 21:35
# @File :  第七题：弃风量和失负荷量.py
# @IDE :  PyCharm

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator

###################################
# 读取数据并处理
# 将数组转换为列表
def list_data(data):
    data_list = data.tolist()
    return data_list
```

```
# 数据按96个时间点分组
def group_data(data_load_list, data_wind_list, data_time_list):
    data_load_group = []
    data_wind_group = []
    data_time_group = []
    for i in range(0, len(data_load_list), 96):
        data_load_group.append(data_load_list[i:i + 96])
        data_wind_group.append(data_wind_list[i:i + 96])
        data_time_group.append(data_time_list[i:i + 96])
    return data_load_group, data_wind_group, data_time_group


# 将时间戳转换为字符串
def time_to_str(data_time_group):
    data_time_str = []
    for i in range(len(data_time_group)):
        data_time_str.append(str(data_time_group[i]))
    return data_time_str


# 读取数据
def read_data():
    data = pd.read_excel('附件2.xlsx')
    data_load = data['负荷功率(MW)']
    data_wind = data['风电功率(MW)']
    data_time = data['时间']

    data_load_list = list_data(data_load)
    data_wind_list = list_data(data_wind)
    data_time_list = list_data(data_time)

    data_time_list = time_to_str(data_time_list)

    data_load_group, data_wind_group, data_time_group = group_data(data_load_list,
                                                                   data_wind_list,
                                                                   data_time_list)
    data_load_group, data_wind_group, data_time_group = np.array(data_load_group), \
                                                        np.array(data_wind_group),\
                                                        np.array(
        data_time_group)
    return data_load_group, data_wind_group, data_time_group


##########################
##弃风总量和丢负荷总量
```

```python
# 计算弃风量
def get_wind_load(load_loss):
    loss_wind_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] > 0:
            loss_wind_sum += load_loss[i]
    return loss_wind_sum


# 计算丢负荷量
def get_loss_load(load_loss):
    loss_load_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] < 0:
            loss_load_sum += load_loss[i]
    return loss_load_sum


############################
## 计算弃风量

def get_wind_loss(**kwargs):
    wind_loss = []
    for i in range(len(kwargs['data_wind_300'])):
        wind_loss_num = -kwargs['target_fire'][i] + kwargs['fireStation1'][i]\
                        + kwargs['data_wind_300'][i]
        wind_loss.append(round(wind_loss_num, 3))
    return wind_loss


################################
## 判断范围
# 判断是否超出范围
def judge(p, up, down):
    if p > up:
        return up, 'up'
    elif p < down:
        return down, 'down'
    else:
        return p, 'ok'


if __name__ == '__main__':
    data_load, data_wind, data_time = read_data()

    # 计算火电功率
```

```
data_fire_raw = data_load - data_wind
data_fire = []
# 判断火电功率范围
for i in range(len(data_fire_raw)):
    data_fire.append([])
    for j in range(len(data_fire_raw[i])):
        data_fire_temp, flag = judge(data_fire_raw[i][j], 600, 180)
        data_fire[i].append(data_fire_temp)
        # print(data_fire[0])

# 计算弃风量
data_wind_loss = []
for i in range(len(data_fire)):
    data_wind_loss.append(
        get_wind_loss(data_wind_300=data_wind[i],
                      target_fire=data_load[i], fireStation1=data_fire[i]))
# print(data_wind_loss[0])

###取出最值
# 每日最大值
data_wind_loss_max_day = np.array(data_wind_loss).max(axis=1)
# print(data_wind_loss_max_day)
# 每日最小值
data_wind_loss_min_day = np.array(data_wind_loss).min(axis=1)
# print(data_wind_loss_min_day)
# 总最大值
data_wind_loss_max_sum = np.array(data_wind_loss).max()
# print(data_wind_loss_max_sum)
# 总最小值
data_wind_loss_min_sum = np.array(data_wind_loss).min()
# print(data_wind_loss_min_sum)

###计算弃风总量和丢负荷总量
# 计算弃风总量和丢负荷总量
loss_wind_sum, loss_load_sum = [], []
for i in range(len(data_wind_loss)):
    loss_wind_sum.append(get_wind_load(data_wind_loss[i]))
    loss_load_sum.append(get_loss_load(data_wind_loss[i]))
# print(loss_wind_sum)
# print(loss_load_sum)

data_wind_loss=np.array(data_wind_loss)
#展开成一维
data_wind_loss_1d=data_wind_loss.flatten()

str_base='7.'
time_new=[]
```

```
    for i in range(16):
        time_new.append(str_base+str(i+1))
    data_new = []
    for i in range(0, 96*16, 96):
        data_new.append(i)

    plt.figure(figsize=(7, 3), dpi=800)
    ax = plt.subplot(111)

    plt.plot(data_wind_loss_1d, color='blue', linewidth=1.5, linestyle='-')
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.title('功率平衡图')
    plt.xlabel('时间')
    plt.ylabel('功率/MW')
    plt.grid(True)
    xmajorLocator = MultipleLocator(96)
    ax.xaxis.set_major_locator(xmajorLocator)
    plt.xticks(data_new, time_new)
    #旋转x轴刻度
    plt.xticks(rotation=90)

    plt.tight_layout()
    plt.savefig('第七题：功率平衡图.png')
    plt.show()
```

2.6.5 第七题：失负荷量.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/27 10:33
# @File :  第二题：弃风.py
# @IDE :  PyCharm
import numpy as np
import pandas as pd

import fireStationCost


def write_file(pd):
    with open('第七题：失负荷量.csv', 'a') as f:
        pd = round(pd, 3)
        pd = str(pd)
        f.write('{}\n'.format(pd))
        f.close()
```

```
# 计算弃风量
def get_wind_loss(**kwargs):
    wind_loss = []
    for i in range(len(kwargs['data_wind_300'])):
        wind_loss_num = -kwargs['target_fire'][i] + kwargs['fireStation1'][i]\
                    + kwargs['data_wind_300'][i]
        wind_loss.append(round(wind_loss_num, 3))
    return wind_loss


# 计算总失负荷量
def get_loss_load(load_loss):
    loss_load_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] < 0:
            loss_load_sum += load_loss[i]
    return -loss_load_sum


# 判断是否超出范围
def judge(p, up, down):
    if p > up:
        return up, 'up'
    elif p < down:
        return down, 'down'
    else:
        return p, 'ok'


def get_wind_load(load_loss):
    loss_wind_sum = 0
    for i in range(len(load_loss)):
        if load_loss[i] > 0:
            loss_wind_sum += load_loss[i]
    return loss_wind_sum


# 煤费用
def coalCost(data_group_1):
    # 计算煤用量
    cost_1 = []
    for i in range(len(data_group_1)):
        cost_1.append(fireStationCost.fireStationRunCost(data_group_1[i],
                                                         0.226, 30.42, 786.80))
    # 将结果转化为array数组
    cost_1 = np.array(cost_1)
```

```
    # 计算15min内的用煤量
    cost_1 = cost_1 * 0.25
    # 计算15min内的用煤费用（700元/吨）
    cost_1 = cost_1 * 0.7
    return cost_1


# 碳费用
def carbonCost(data_group_1, carbon_price):
    cost_1 = fireStationCost.carbonEmission(data_group_1, 0.72) * carbon_price
    return cost_1


def singleRun(storage_capacity, power_capacity, carbon_price=0):
    # 读取风电场数据
    data_wind = pd.read_excel('附件2.xlsx')
    data_wind = data_wind.dropna()
    data_wind_900 = data_wind['风电功率(MW)']
    data_target = data_wind['负荷功率(MW)']
    # 转换为array
    data_wind_900, data_target = np.array(data_wind_900), np.array(data_target)
    # 计算机组1功率
    fireStation = data_target - data_wind_900
    # 判断机组1功率是否超出范围
    fireStation1 = []
    for i in range(len(fireStation)):
        p, judge_result = judge(fireStation[i], 600, 180)
        fireStation1.append(p)
    # 计算弃风量
    load_loss = get_wind_loss(data_wind_300=data_wind_900, fireStation1=fireStation1,
                              target_fire=data_target)
    # print(load_loss)
    # 计算总失负荷量
    loss_load_sum = get_loss_load(load_loss) * 0.25
    # 计算丢负荷损失
    loss_load_cost = loss_load_sum * 8 * 1000
    # print(loss_load_sum)
    #####计算储能成本
    # 计算单次最大丢负荷量
    loss_load_max = min(load_loss)
    # 取绝对值
    loss_load_max = abs(loss_load_max)
    # 所需储能容量
    # storage_capacity = loss_load_max / 0.9
    # power_capacity = storage_capacity
    ## 储能投资成本
    # 单位功率成本
```

```
    unit_cost = storage_capacity * 3000 * 1000
    # 单位能量成本
    unit_energy_cost = power_capacity * 0.25 * 3000 * 1000
    ## 储能运维成本
    storage_capacity_cost = loss_load_sum * 0.05 * 1000
    # 总储能成本
    storage_cost = (unit_cost + unit_energy_cost) / 10 / 365 + storage_capacity_cost
    # 计算弃风量
    loss_wind_sum = get_wind_load(load_loss) * 0.25
    # 计算弃风损失
    loss_wind_cost = loss_wind_sum * 0.3 * 1000
    # 煤费用
    cost_coal = coalCost(data_group_1=fireStation1)
    coalSum = sum(cost_coal)
    # 碳排放费用

    cost_carbon = carbonCost(data_group_1=fireStation1, carbon_price=carbon_price)
    # 总成本
    total_cost = loss_wind_cost + coalSum + cost_carbon + storage_cost
    loadSum = sum(data_target) * 0.25
    print('风电装机1200MW、替代机组2、3时，丢负荷电量{}MWh，需要配备最小储能容量为{}MWh'
          '。考虑碳捕集成本{}元/吨，此时单位供电成本为{}元/KWh'.format(
        round(loss_load_sum, 3),
        round(storage_capacity, 3),
        carbon_price,
        round(total_cost / loadSum / 1000,5)))
    return total_cost / loadSum / 1000


if __name__ == '__main__':
    singleRun(storage_capacity=0, power_capacity=0, carbon_price=100)
```

2.6.6 第七题：遗传算法求最优解.py

```
# -*- coding: utf-8 -*-
# @Time :  2022/5/28 15:57
# @File :  遗传算法.py
# @IDE :  PyCharm

import matplotlib.pyplot as plt

# from 第七题: 失负荷量 import *
from q7singleRun import *
```

```
def fitness_func(X, price):
    # 目标函数，即适应度值，X是种群的表现型
    a = 10
    pi = np.pi
    x = X[:, 0]
    y = X[:, 1]
    res = singleRun(x, y, carbon_price=price)

    return res


def decode_X(X: np.array):
    """对整个种群的基因解码，上面的decode是对某个染色体的某个变量进行解码"""
    X2 = np.zeros((X.shape[0], 2))
    for i in range(X.shape[0]):
        xi = decode(X[i, :20], 0, 600)
        yi = decode(X[i, 20:], 0, 150)
        X2[i, :] = np.array([xi, yi])
    return X2


def decode(x, a, b):
    """解码，即基因型到表现型"""
    xt = 0
    for i in range(len(x)):
        xt = xt + x[i] * np.power(2, i)
    return a + xt * (b - a) / (np.power(2, len(x)) - 1)


def select(X, fitness):
    """根据轮盘赌法选择优秀个体"""
    fitness = 1 / fitness
    fitness = fitness / fitness.sum() # 归一化
    idx = np.array(list(range(X.shape[0])))
    X2_idx = np.random.choice(idx, size=X.shape[0], p=fitness) # 根据概率选择
    X2 = X[X2_idx, :]
    return X2


def crossover(X, c):
    """按顺序选择2个个体以概率c进行交叉操作"""
    for i in range(0, X.shape[0], 2):
        xa = X[i, :]
        xb = X[i + 1, :]
        for j in range(X.shape[1]):
            # 产生0-1区间的均匀分布随机数，判断是否需要进行交叉替换
            if np.random.rand() <= c:
```

```
                xa[j], xb[j] = xb[j], xa[j]
        X[i, :] = xa
        X[i + 1, :] = xb
    return X


def mutation(X, m):
    """变异操作"""
    for i in range(X.shape[0]):
        for j in range(X.shape[1]):
            if np.random.rand() <= m:
                X[i, j] = (X[i, j] + 1) % 2
    return X


def ga(c=0.4, m=0.1, iter_num=100, price=100):
    """遗传算法主函数"""

    best_fitness = [] # 记录每次迭代的效果
    best_xy = []
    # iter_num = 500 # 最大迭代次数
    X0 = np.random.randint(0, 2, (50, 40)) # 随机初始化种群，为50*40的0-1矩阵
    import tqdm
    for i in tqdm.tqdm(range(iter_num)):
        # print('第{}次迭代'.format(i))
        X1 = decode_X(X0) # 染色体解码
        fitness = fitness_func(X1, price) # 计算个体适应度
        X2 = select(X0, fitness) # 选择操作
        X3 = crossover(X2, c) # 交叉操作
        X4 = mutation(X3, m) # 变异操作
        # 计算一轮迭代的效果
        X5 = decode_X(X4)
        fitness = fitness_func(X5, price)
        best_fitness.append(fitness.min())
        x, y = X5[fitness.argmin()]
        best_xy.append((x, y))
        X0 = X4
    # 多次迭代后的最终效果
    print("最优值是: %.5f" % best_fitness[-1])

    print("最优解是: x=%.5f, y=%.5f" % best_xy[-1])

    title = "最优值是: %.5f" % best_fitness[-1]
    title += "。最优解是: x=%.5f, y=%.5f" % best_xy[-1]
    # 最优值是: 0.00000
    # 最优解是: x=0.00000, y=-0.00000
    # 打印效果
```

```
    plt.figure(figsize=(7, 3), dpi=800)
    plt.plot(best_fitness, color='r')
    # plt.grid()
    # 图片名称
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.title(title)
    plt.xlabel('迭代次数(次)')
    plt.ylabel('系统供电成本(元/KWh)')
    picName = '交叉：{}，变异：{}，迭代次数：{}，碳捕集成本：{}.png'.\
        format(c, m, iter_num, price)
    plt.tight_layout()
    plt.savefig(picName)
    plt.show()


if __name__ == '__main__':
    ga(c=0.5, m=0.05, iter_num=5000, price=0)
```