回焊炉机理模型研究

摘要

为解决回流焊过程中，回焊炉各温区的温度设定和传送带的速度设定问题，本文由相关物理规律入手，建立了炉内空气温度分布和炉温曲线的机理模型，并且可以根据制程界限和特定的优化目标，选取最优的温度、速度设定值。

针对问题 1，本文根据传热学中的一维稳态传热模型，得出已知回焊炉内两点温度，且其中间区域达到稳态时，温度随空间的分布。进而通过定性分析相关文献中回焊炉内空气温度的实验数据，确定整个区域内温度随空间的分布。再根据传热学中的集总热容系统模型，得出焊接区域温度变化速率，与焊接区域和外界温差之间的联系，并通过附件中的数据验证模型曲线的变化趋势，确定模型中的比例系数。最终通过程序模拟，可以根据设定温度和设定速度计算出炉温曲线的数据，并绘制出炉温曲线。

针对问题 2，考虑到速度设定范围不大，本文选用以较高精度遍历可能的速度值，并且代入问题 1 中模型进行验证的方法，选择出温度给定时，满足制程界限的最大传送带速度。

针对问题 3，主要目标是多个参数的共同优化问题，并且参数到优化目标值的函数关系相对复杂，不利于求出解析解。本文使用遗传算法选取最优参数。

针对问题 4，本文首先引入面积的交并比来评价图形的对称程度，然后建立了一个综合问题 3 中面积和问题 4 中交并比的新优化目标，进而使用遗传算法对参数进行了优化。

关键词： 回流焊 一维稳态传热 集总热容系统 遗传算法

一、 问题重述

在集成电路板等电子产品生产中，需要将安装有各种电子元件的电路板放置于传送带上，匀速穿过回焊炉。通过回焊炉的加热作用，电子元件被焊接到电路板上。

回焊炉内部包含若干个小温区，每个小温区的温度可以单独调控。小温区从功能上可以分为四个大温区，分别为预热区、恒温区、回流区、冷却区。

本题中所研究的回焊炉，内有 11 个小温区及炉前区域和炉后区域，每个小温区长度为 30.5cm，相邻小温区之间有 5cm 的间隙，炉前区域和炉后区域长度均为 25cm。如图 1 所示：

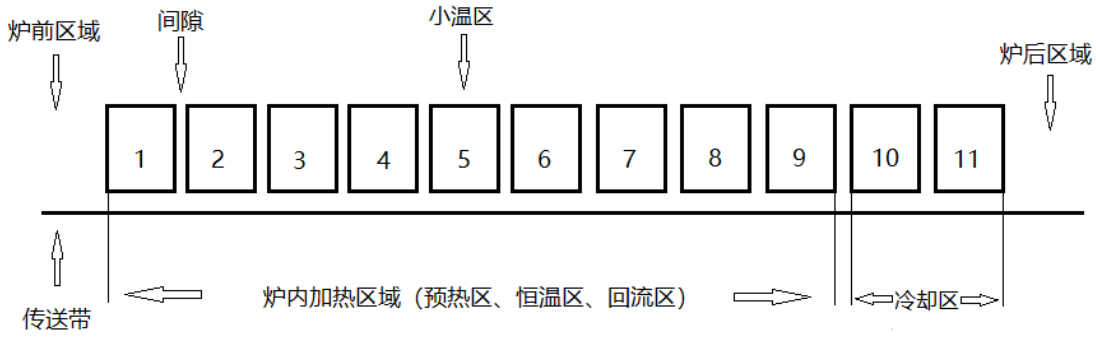

图 1 回焊炉示意图

在整个过程中，回焊炉各部分的温度对于产品质量有着显著的影响，回焊炉内部只对各小温区进行温度控制。炉前区域、炉后区域和小温区间隙的温度都与相邻温区温度有关，各温区边界附近的温度也可能受到相邻温区温度的影响。车间温度保持在 25℃。回焊炉内的空气会在启动后的短时间内达到稳定，然后即可开始进行焊接。

本文将建立合适的模型和适当的算法，解决以下四个问题：

1.在给定传送带过炉速度和各个小温区温度的条件下，根据焊接区域中心温度变化规律建立数学模型，列出焊接区域中心到达指定位置时的温度，并画出炉温曲线。

2.在给定各个小温区温度的条件下，确定满足制程界限的最大过炉速度。

3.为避免焊接中心区域超过 217℃的时间过长和峰值温度太高，应使得炉温曲线中超过 217℃到峰值温度的部分与 217℃水平线所围成的面积最小，在此条

件下确定炉温曲线、每个小温区的温度设定和过炉速度，并给出所围成的面积。

4.在实际的焊接中，不仅要满足制程限制，还希望尽可能使得炉温曲线中以峰值为中心线的两侧超过217℃的部分尽量对称，并结合问题3给出进一步优化的炉温曲线和相应指标。

二、 问题分析

本文所研究的问题主要围绕焊接区域中心温度的变化而展开。

首先，回焊炉启动后炉内空气在短时间内达到稳定并保持不变，因此可根据一维稳态导热相关知识用每个小温区的设定温度计算出整个回焊炉内各个位置的空气温度。

其次，分析传送带上电子元件的温度变化情况，由于回焊炉内不同位置有着不同的温度，所以在回焊炉工作过程中电子元件接触到的空气温度一直发生变化，可根据一维非稳态导热相关知识计算出焊接区域中心的温度，然后可对四个问题进行具体分析。

2.1 问题 1

首先对于回焊炉内各个温区中空气温度分布建立一维稳态导热模型，利用导热方程，根据炉内空气的导热系数和各个小温区的设定温度计算出整个回焊炉内各个位置的空气温度，然后根据电子元件导热情况计算出焊接区域中心温度随时间或路程的变化，进而求出经过指定位置时焊接区域中心的温度以及炉温曲线。

2.2 问题 2

根据问题1求出的焊接区域中心温度随小温区温度的变化规律，画出在给定条件下的炉温曲线，并计算出制程界限表格中提到的温度上升斜率、温度下降斜率、峰值温度等5个指标，由于传送带过炉速度可调节范围不大，所以可编辑程序按照0.01cm/min的步长，在合理的时间复杂度下，遍历整个65~100cm/min范围内的过炉速度，并检验制程界限中的五个指标是否满足要求，从而求出符合要求的最大过炉速度，精确到0.01cm/min。

2.3 问题 3

根据问题 1 中分析得到的焊接中心随温度、过炉速度的变化规律，用程序求解出焊接中心温度随各个温区的设定温度和过炉速度的变化函数，然后采用 python 遗传算法库 scikit-opt，求解出阴影面积最小时的各个温区设定温度和传送带过炉速度，并画出相应的炉温曲线。

2.4 问题 4

以交并比为指标，衡量炉温曲线中，以峰值温度为中心线的两侧超过 217℃部分的对称情况，将炉温曲线中超过 217℃的部分以峰值作为中心线，把中心线一侧的曲线对折，使得炉温曲线与直线 T = 217℃围成两个图形，接着计算出两个图像交的面积和并的面积，并由此计算出可以用来衡量对称情况的交并比。接着需进一步综合衡量阴影面积和对称情况，由于阴影面积越小越好，交并比越大越好，因此可将阴影面积和交并比的倒数相乘，根据乘积与各个温区温度设定和传送带过炉速度的关系，采用 python 遗传算法库 scikit-opt，求解出使得这个乘积最小时的各个指标，并画出炉温曲线。

三、 基本假设

1.工件在过炉过程中对炉内空气温度分布没有影响，并且不考虑气流移动。

2.不考虑回焊炉内空气的横截面积大小对于空气导热的影响，将回焊炉内不同温区中的空气之间的导热视为平壁导热。

3.忽略回焊炉壁与外界环境之间的传热。

4.当工件进入回焊炉内时，回焊炉内的空气温度已经达到了稳定状态。

四、 符号说明

符号	意义	单位
T_{15}	小温区 1~5 的设定温度	℃
T_6	小温区 6 的设定温度	℃
T_7	小温区 7 的设定温度	℃
T_{89}	小温区 8~9 的设定温度	℃

v	过炉速度	cm/min
Bi	毕奥数	无量纲
h	流体的对流换热系数	W/(m^2•℃)
l	物体特征长度	m
λ	物体导热系数	W/(m•℃)
ΔT	炉内空气与焊接区域温度之差	℃
ρ	物体密度	kg/m^3
c	物体比热容	kJ/(kg•℃)
S_{high}	问题 3 中阴影部分面积	℃•s

五、 模型建立与求解

5.1 问题 1

5.1.1 一维稳态导热模型

依据题意，电路板两侧搭在传送带上匀速通过回焊炉，传送带为直线型，电路板的路径也因此为一条固定的笔直轨迹，回焊炉启动后，炉内空气温度会在短时间内达到稳定状态，因此可以以传送带方向为 X 轴，传送带各个位置达到稳定后的空气温度（包括炉前、炉内、炉后以及间隙）为 Y 轴，建立空气温度随位置的变化曲线，对于每一个固定的位置，其温度都与其邻近温区的温度有关，并且在其影响下短时间内达到稳态。由以上分析，为分析炉内空气温度分布，可以建立沿传送带方向上的一维稳态导热模型。

由传热学[1]中的导热微分方程可知，对于三维空间下一个长宽高分别为 dx、dy、dz 的微元体，在单位时间 dt 其内部的净热流量 Δ Φ 满足：

$$\Delta \Phi = -\left(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z}\right) dxdydzdt \tag{1}$$

其中q_x、q_y、q_z分别为三维方向上的热流密度，再根据傅里叶定律，有

$$\begin{cases} q_x = -\lambda \dfrac{\partial T}{\partial x} \\ q_y = -\lambda \dfrac{\partial T}{\partial y} \\ q_z = -\lambda \dfrac{\partial T}{\partial z} \end{cases} \tag{2}$$

其中 T 为温度，λ为导热系数，将(2)式代入(1)，得：

$$\Delta \Phi = \left[\frac{\partial}{\partial x}(\lambda \frac{\partial T}{\partial x}) + \frac{\partial}{\partial y}(\lambda \frac{\partial T}{\partial y}) + \frac{\partial}{\partial z}(\lambda \frac{\partial T}{\partial z})\right] dxdydzdt \tag{3}$$

因为在稳态下，$\Delta \Phi$ 恒为 0，因此有：

$$\frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\lambda \frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\lambda \frac{\partial T}{\partial z}\right) = 0 \tag{4}$$

而在本例的一维空间下，则为

$$\frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right) = 0 \tag{5}$$

对其积分，得：

$$\lambda \frac{\partial T}{\partial x} = C_1 \tag{6}$$

空气的导热系数λ在一定范围内可以认为与温度成正比例关系，可表示为：

$$\lambda = \lambda_0(1 + bT) \tag{7}$$

中λ_0、b均为常数，T 单位为℃. 将(7)式代入(6)式，得：

$$\lambda_0(1 + bT)\frac{\partial T}{\partial x} = C_1 \tag{8}$$

对(8)式积分，得：

$$\mathrm{T} + \frac{1}{2}bT^2 = \frac{C_1}{\lambda_0}x + C_2 \tag{9}$$

对于下图中的具体情况，

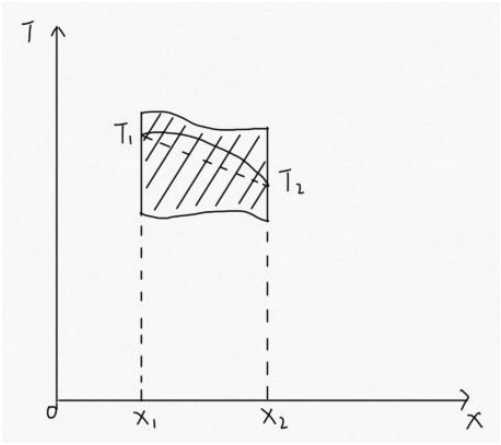

图 2 一维热稳态示意图

阴影部分为一个达到稳态的介质（本题中为空气），相同 x 处的所有部位温度都相等，其左右两侧 X1、X2 处的温度分别为 T1、T2。因为是稳态，故在 X 轴一维方向上，任何一处的温度都满足上述的(9)式，因此将(x1, T1)、(x2, T2)分别代入(9)式，得到方程组：

$$\begin{cases} T_1 + \frac{1}{2}bT_1{}^2 = \frac{C_1}{\lambda_0}x_1 + C_2 \\ T_2 + \frac{1}{2}bT_2{}^2 = \frac{C_1}{\lambda_0}x_2 + C_2 \end{cases} \tag{10}$$

解二元一次方程组得：

$$\begin{cases} C_1 = \frac{\lambda_0(T_2-T_1)\left[\frac{b}{2}(T_1+T_2)+1\right]}{x_2-x_1} \\ C_2 = \frac{x_2T_1\left(\frac{b}{2}T_1+1\right)-x_1T_2\left(\frac{b}{2}T_2+1\right)}{x_2-x_1} \end{cases} \tag{11}$$

同时由(9)能够解得$x_1 \sim x_2$之间任意位置处的温度 T 为：

$$\mathrm{T} = \frac{\sqrt{1+2b\left(\frac{C_1x}{\lambda_0}+C_2\right)}-1}{b} \tag{12}$$

5.1.2 回焊炉内空气温度分布模型

相关文献[2]某次回焊炉实验中，设定加热温度与实际测量空气温度的对比图如图 3。

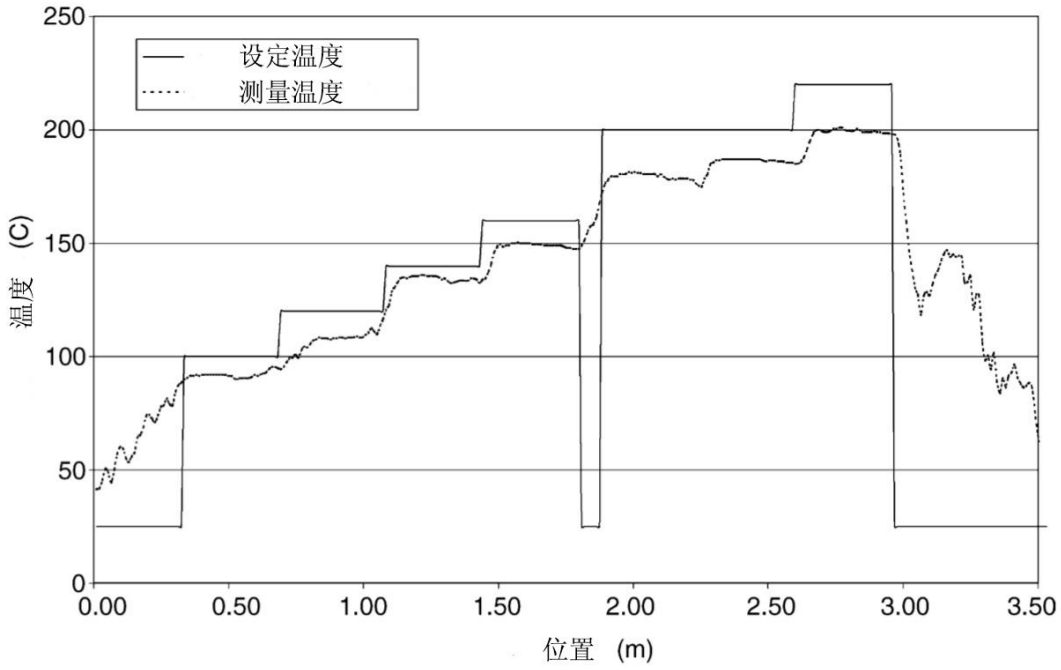

图 3 设定加热温度与实际测量空气温度对比图[2]

对上图进行定性分析，将沿传送带方向上的炉内空间，分成若干个分区，对于每一个分区，利用上述一维稳态导热模型，得到每一个分区的空气温度分布，进而得到整个炉内的空气温度分布模型（图 4）：

炉前区域起点与温区 1 中点建立热稳态；

温区 5 中点与温区 6 中点建立热稳态；

温区 6 中点与温区 7 中点建立热稳态；

温区 7 中点与温区 8 中点建立热稳态；

温区 9 终点与炉后区域终点建立热稳态。

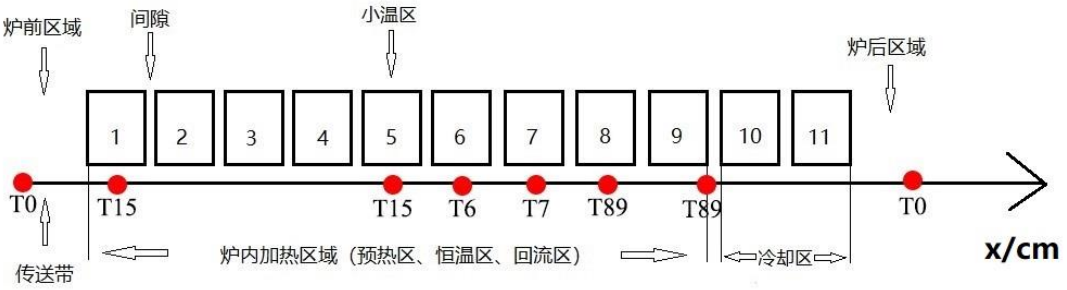

图 4 设定温度与空气温度分布之间的关系

5.1.3 焊接区域在回焊炉内加热（冷却）模型

焊接区域与回焊炉内空气之间是非稳态导热过程。非稳态导热过程中，表征物体内部导热热阻，与物体表面对流换热热阻相对大小的无量纲数称为毕奥数：

$$\mathrm{Bi} = \frac{hl}{\lambda} \tag{13}$$

当 Bi≤0.1 时，可以近似认为物体内部的导热热阻与表面对流换热热阻相比可以忽略不计，这种内部温度梯度小得可以忽略的导热体称为集总热容系统。对于集总热容系统，物体温度对时间的变化率满足：[1]

$$\frac{dT}{dt} = \frac{h\Delta T}{\rho c l} \tag{14}$$

定义比例系数 C_T，表示焊接区域温度对时间的变化率$\frac{dT}{dt}$，与物体与炉内空气温差ΔT之间的比值：

$$C_T = \frac{h}{\rho c l} \tag{15}$$

根据附件中的实验数据，由于小温区 2 到小温区 4 的空间范围内，空气温度可近似看作恒定为设定的温度。对于这一范围内的每个数据点，焊接区域温度对时间的变化率，与物体与外界温差作比，可以得到 C_T 的估计值，再对数据点取

平均得到整个过程中 C_T 的估计值，进而解出 Bi，代入检验符合集总热容系统的条件。然后对 C_T 附近的值进行二分查找，选择出 C_T 的最优取值，使得在附件对应的条件下，计算出的炉温曲线最高温度与附件数据中的最高温度之差足够小。利用 C_T 的值和回焊炉内的空气温度分布，可以计算出焊接区域温度与时间的关系，并绘制出炉温曲线。

5.1.4 问题 1 求解

通过估算，本题中空气温度变化范围大致在 300K~500K，通过查询表格得知，空气温度为 300K 时热传导系数为 0.0263W/(m•K)，空气温度为 500K 时热传导系数为 0.0407W/(m•K)，将这两个数值带入上述公式(7)

$$\lambda = \lambda_0(1 + bT)$$

式中 T 的单位为℃.

求得：

$$\begin{cases} \lambda_0 = 0.024356 \\ b = \frac{0.0072}{2.4356} \end{cases} \tag{16}$$

从上述分析得知，将整个回焊炉分成若干个热稳态分区，每个分区中空气的变化符合以下模型(即上述公式 11 和公式 12)：

$$\begin{cases} C_1 = \dfrac{\lambda_0(T_2 - T_1)\left[\frac{b}{2}(T_1 + T_2) + 1\right]}{x_2 - x_1} \\ C_2 = \dfrac{x_2 T_1\left(\frac{b}{2}T_1 + 1\right) - x_1 T_2\left(\frac{b}{2}T_2 + 1\right)}{x_2 - x_1} \end{cases}$$

$$\mathrm{T} = \frac{\sqrt{1 + 2b(\frac{C_1 x}{\lambda_0} + C_2)} - 1}{b}$$

其中 T_1、T_2 分别为每个分区左、右侧边缘的温度，x1、x2 分别为分区左、右边缘的坐标，其他参数均已知。整个回焊炉内空气温度随位置的变化关系 T(x) 由所有分区的曲线合并而来。

根据各个分区的温度设定和位置运行程序 1，分段画出回焊炉内完整的空气温度随位置变化的曲线（图 5），并且将结果保存在列表中。

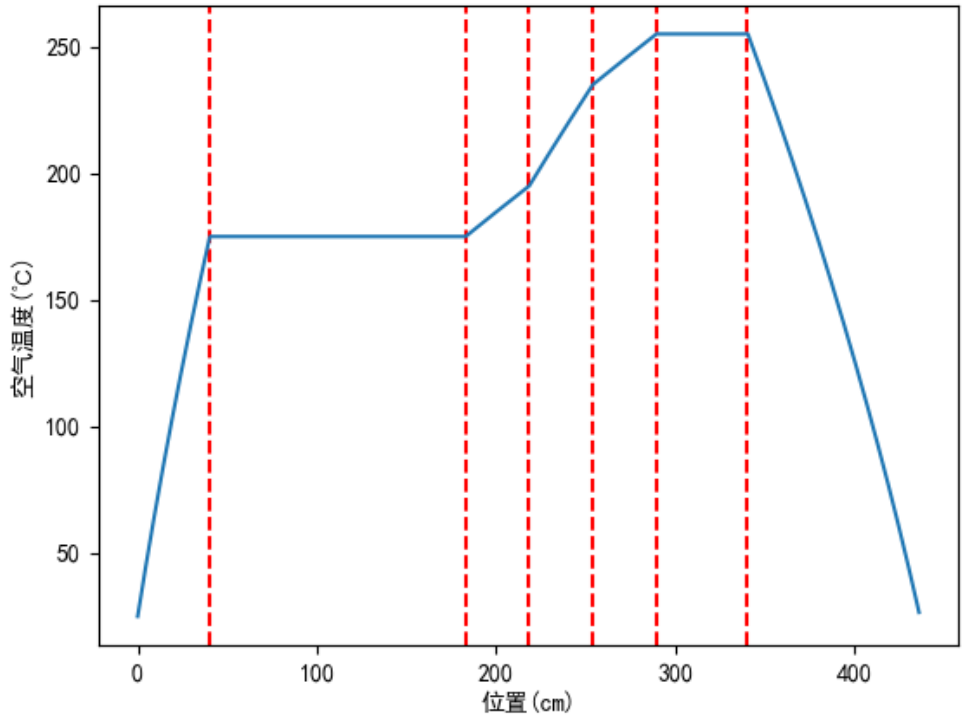

图 5 空气温度曲线

根据附件中的实验数据，取小温区 2 到小温区 4 的空间范围内，空气温度为 175℃。对于这一范围内的每个数据点，焊接区域温度对时间的变化率，与物体与外界温差作比，再对数据点取平均得到整个过程中 C_T 的估计值为 0.0201。代入公式 (13) 和公式 (15) 解出 Bi = 2.692e-6，Bi ≤ 0.1，符合集总热容系统的条件。然后运行程序 2 对[0.015, 0.025]区间上进行二分查找，选择出 C_T 的最优取值为 0.0216。

之后，由公式(14)和公式(15)得：

$$\frac{dT}{dt} = C_T \Delta T \tag{17}$$

利用 C_T 的值和回焊炉内的空气温度分布，可以利用程序 3 积分计算出焊接区域温度与时间的关系，并绘制出炉温曲线如下：

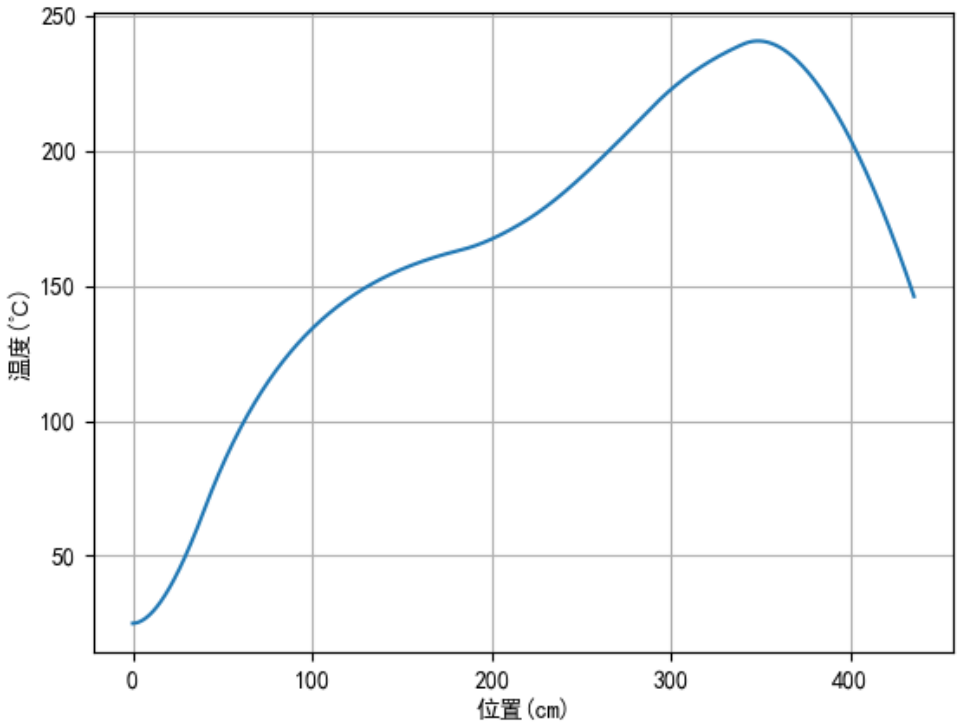

图 6 问题 1 中炉温曲线

同时，程序 3 会输出所求位置处的温度如下：

小温区 3 中点温度为：140.78℃

小温区 6 中点温度为：173.57℃

小温区 7 中点温度为：192.21℃

小温区 8 结束处温度：224.90℃

通过程序 4 将横坐标的位置换算为时间，计算出每隔 0.5 s 焊接区域中心的温度，并存放在提供的 result.csv 中，详见支撑材料。

5.2 问题 2

5.2.1 制程界限中的指标与传送带过炉速度的关系

首先，利用问题 1 分析得到的模型，输入过炉速度 v，由于各个小温区的设定温度已经给定，所以可以画出炉温曲线。然后根据炉温曲线的数据可以计算特定的过炉速度下，温度上升斜率、温度下降斜率、温度上升过程中在 150℃~190℃的时间等五个指标。从而得到这五个指标与过炉速度 v 的关系。

5.2.2 求出满足制程界限的最大传送带过炉速度

由于过炉速度未知，并且五个制程界限指标随过炉速度变化，所以需要在给定过炉速度时对五个制程界限指标进行检查，求出五个指标均满足要求时的最大过炉速度。使用 python 程序编写函数，函数参数为过炉速度，函数返回值为是否满足制程界限，在函数内部求解出五个制程界限指标并进行合理性检查。由于速度变化范围不大，对过炉速度 v 的所有可能取值在一定精度下进行遍历，求出满足制程要求的最大过炉速度。

5.2.3 问题 2 求解

编写 python 函数求出五个制程界限指标与过炉速度 v 的关系：

(1)温度上升斜率，温度下降斜率

由于二者的绝对值均需要满足不大于 3，所以只需要判断整个炉温曲线中斜率绝对值的最大值是否小于等于 3。本题采用与问题 1 同样的处理方式，将炉温曲线上的数据离散化，斜率 k 近似为数据中相邻两个点之间的温度变化量与时间变化量的比值。为满足制程要求，$\max(|k|)\leq 3$.

(2)温度上升过程中在 150℃~190℃的时间

用 t_{150}，t_{190} 分别表示上升过程中温度达到 150℃和 190℃时的时间。

先令 t_{150} 为 0，当当前温度 $T_{now}\geq 150℃$并且 $t_{150}=0$，即 t_{150} 没有更新过，使得 t_{150} 为当前时间。同理当 $T_{now}\leq 190℃$，并且 $k>0$ 时，使得 t_{190} 为此时时间。用 $t_{190} - t_{150}$ 即可得到温度上升过程中在 150℃~190℃的时间。为满足制程要求，$60s \leq t_{190} - t_{150} \leq 120s$。

(3)温度大于 217℃的时间

用 t_{217_1} 表示上升过程中温度达到 217℃时的时间，用 t_{217_2} 表示下降过程中温度达到 217℃时的时间。

当前一时刻温度 $T_{prev}\leq 217℃$并且 $T_{now}\geq 217℃$时，表示此时温度刚好超过 217℃，将此时的时间记录为 t_{217_1}。同理当 $T_{prev}\geq 217℃$ 并且 $T_{now}\leq 217℃$时，表示此时温度刚好低于 217℃，将此时的时间记录为 t_{217_2}。所以 $t_{217\_2} - t_{217\_1}$ 则表示温度大于 217℃的时间。为满足制程要求，$40s \leq t_{217\_2} - t_{217\_1} \leq 90s$。

(4)峰值温度

由于已经将炉温曲线上所有数据点存在于列表中，所以可以使用 max()方法求出温度列表中的最大值，即峰值温度。为满足制程要求，$240℃ \leq T_{max} \leq 250℃$.

利用程序 5 对 v 进行遍历，从 100cm/min 开始，步长取(-0.01)cm/min。当得

到一个满足要求的 v 值时跳出循环并输出结果。运行后求得的最大过炉速度为 79.16cm/min.

5.3 问题 3

5.3.1 遗传算法模型

遗传算法是模拟自然选择和进化过程的计算模型，是一种通过模拟自然进化过程寻找最优解的方法。它直接对结构对象进行操作，避免了求导和函数连续性的限定，并能够自适应地调整搜索方向。

根据问题 1 中的模型，输入各温区设定温度和传送带速度，可以得到炉温曲线，进而计算出 S_{high}。因此对于问题 3 可以建立输入为设定温度和传送带速度，优化目标为 S_{high} 最小，输入到 S_{high} 函数关系已知的遗传算法模型，搜索设定温度和传送带速度的最优组合。特别地，在输入到 S_{high} 的函数中，输入使得炉温曲线不满足制程界限时返回无穷大，使得不满足制程界限的输入在遗传算法模型中不被保留。

5.3.2 问题 3 求解

用 t217_1 表示上升过程中温度达到 217℃时的时刻，用 t_{max} 表示到达最高温度时的时刻，对两个时刻之间的区域进行细分，将所有小的细分单元的面积累加，得到最终的阴影部分的面积，即 S_{high}。

将 scikit-opt 库中的 GA 类实例化，将初始群体个体数设置为 200，迭代次数设置为 10，并且设置小温区温度和传送带速度的可调节范围，精度设置为 1e-7, 然后调用 ga 实例的 run 方法，得到最优的小温区温度和传送带过炉速度以及相应的面积。

最终根据程序运行结果求得：

T_{15} = 170.060℃

T_6 = 186.089℃

T_7 = 239.017℃

T_{89} = 264.924℃

v = 94.355cm/min

S_{high}=473.426℃•s

并利用程序 3 画出炉温曲线如下：

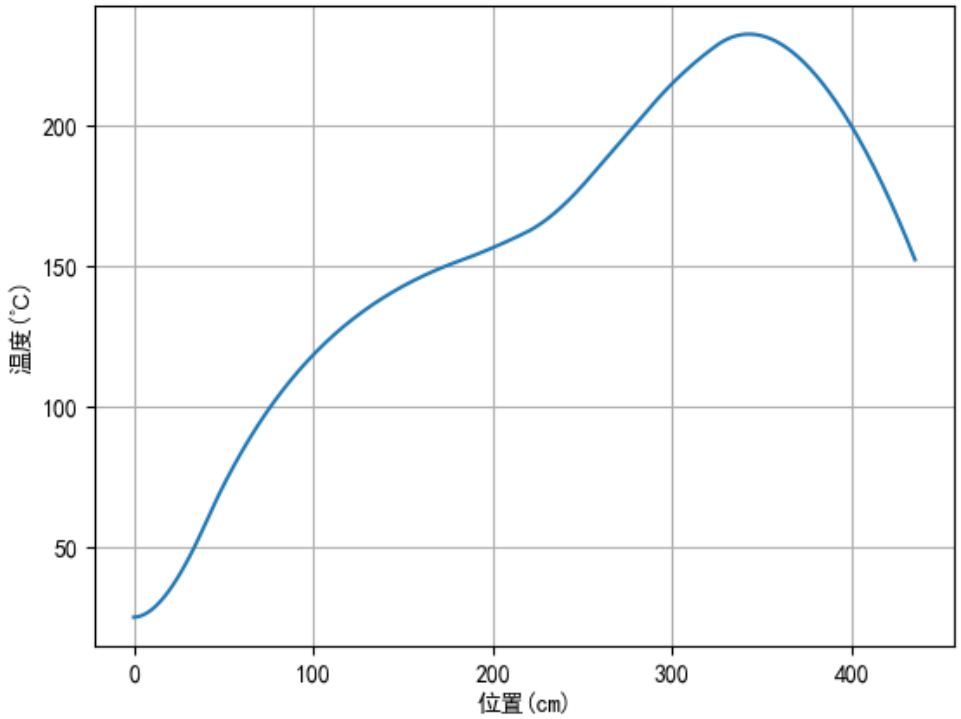

图 7 问题 3 中炉温曲线

5.4 问题 4

5.4.1 双目标共同优化

问题 4 与问题 3 同为优化问题，同样采用输入为设定温度和传送带速度的遗传算法模型，而优化目标变为 S_{high} 最小，和以峰值温度为中心线的两侧超过 217ºC 的炉温曲线对称程度最大的综合。

上述对称程度可以等价为峰值温度中心线一侧超过 217ºC 的区域，沿中心线对称后，与另一侧区域的重合程度。为评价重合程度，引入两区域交并比的概念：

$$IOU = S_{\cap}/S_{\cup} \qquad (18)$$

$S_{\cap}$ 表示两区域交的面积，$S_{\cup}$ 表示两区域并的面积。交并比的可能范围是[0 ,1]，0 表示完全不相交，1 表示完全重合。

为了综合考虑 S_{high} 和 IOU，设定优化目标为 $\frac{S_{high}}{IOU}$ 最小，将设定温度和传送带速度与 $\frac{S_{high}}{IOU}$ 的函数关系代入遗传算法模型，可以搜索得到设定温度和传送带速度的最优组合。特别地，在计算 $\frac{S_{high}}{IOU}$ 的函数中，输入使得炉温曲线不满足

制程界限时返回无穷大，使得不满足制程界限的输入在遗传算法模型中不被保留。

5.4.2 问题 4 求解

用 t_{left} 表示上升过程中温度达到 217℃时的时刻，用 t_{right} 表示下降过程中温度达到 217℃时的时刻，用 t_{mid} 表示到达最高温度时的时刻。

下面通过手工图示，介绍交并比 IOU 的计算原理：

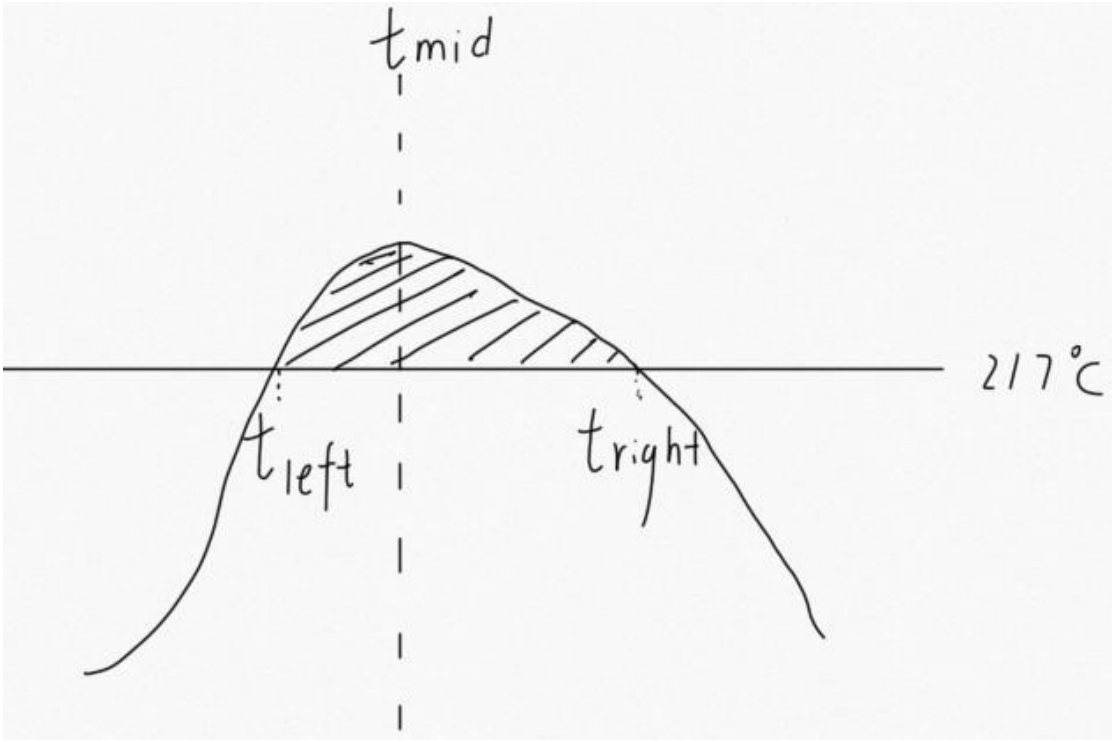

图 8 翻折前

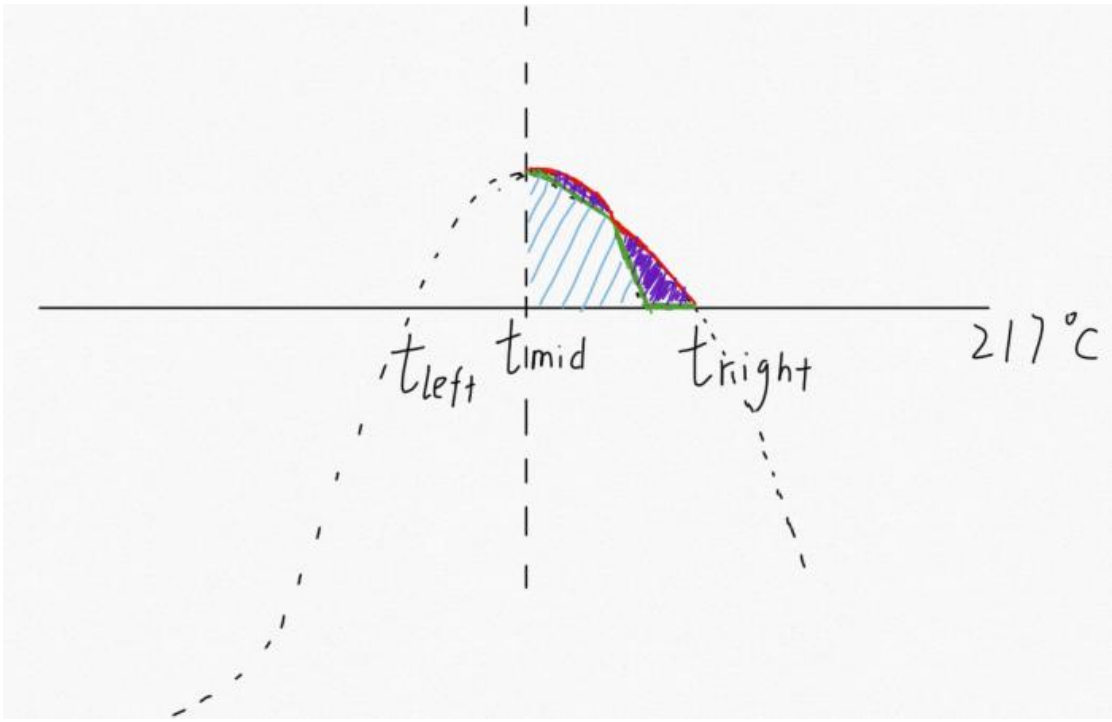

图 9 翻折后

如图 8、图 9，将炉温曲线以 t_{mid} 为轴，将左侧翻折至右侧，则 $S_{\cap}$ 即为图中蓝色涂线部分，亦即绿色线与 217℃水平线所围成的区域面积；$S_{\cup}$ 即为图中蓝色涂线部分与紫色部分面积之和，亦即红色线与 217℃水平线所围成的区域面积。利用程序 7 中的函数分别对绿色线和红色线相对于 217℃水平线进行积分，求得 $S_{\cap}$ 和 $S_{\cup}$，进而求交并比 IOU = $\frac{S\cap}{S\cup}$ ，可见通过交并比能够很好的衡量炉温曲线高于 217℃部分的对称情况。

面积 S_{high} 仍然按照上一题的方式求解。

将 scikit-opt 库中的 GA 类实例化，将初始群体个体数设置为 200，迭代次数设置为 10，并且设置小温区温度和传送带速度的可调节范围，精度设置为 1e-7, 然后调用 ga 实例的 run 方法，得到最优的小温区温度和传送带过炉速度以及相应的 $\frac{S_{\mathrm{high}}}{\mathrm{IOU}}$。

最终根据程序运行结果求得：

T_{15} = 171.210℃

T_6 = 185.815℃

T_7 = 230.136℃

T_{89} = 264.952℃

v = 91.629cm/min

S_{high}=476.398℃•s

IOU=0.873

并利用程序 3 画出炉温曲线如下：

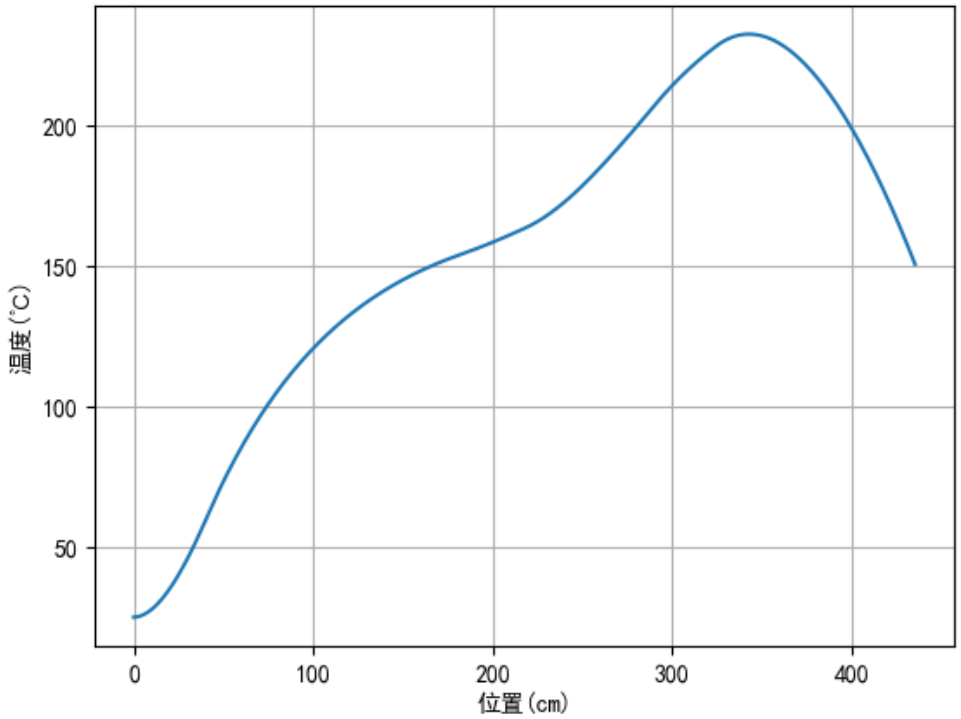

图 10 问题 4 中炉温曲线

六、 模型评价与推广

本文主要描述了回焊炉内电子元件焊接中心的温度随着炉内小温区温度设定和过炉速度的变化规律，并用此规律解决四个问题。

整个模型建立与求解过程中可分为两部分，第一部分是对于炉内空气温度的计算，这里采用一维稳态传热模型，对于空气温度进行计算，在计算结束后与题中给定的实验数据进行对比，发现在相同条件下所拟合的温度曲线与题中所给数据画出的曲线非常接近，表明以此模型计算炉内空气温度计算较为准确。（如下图）

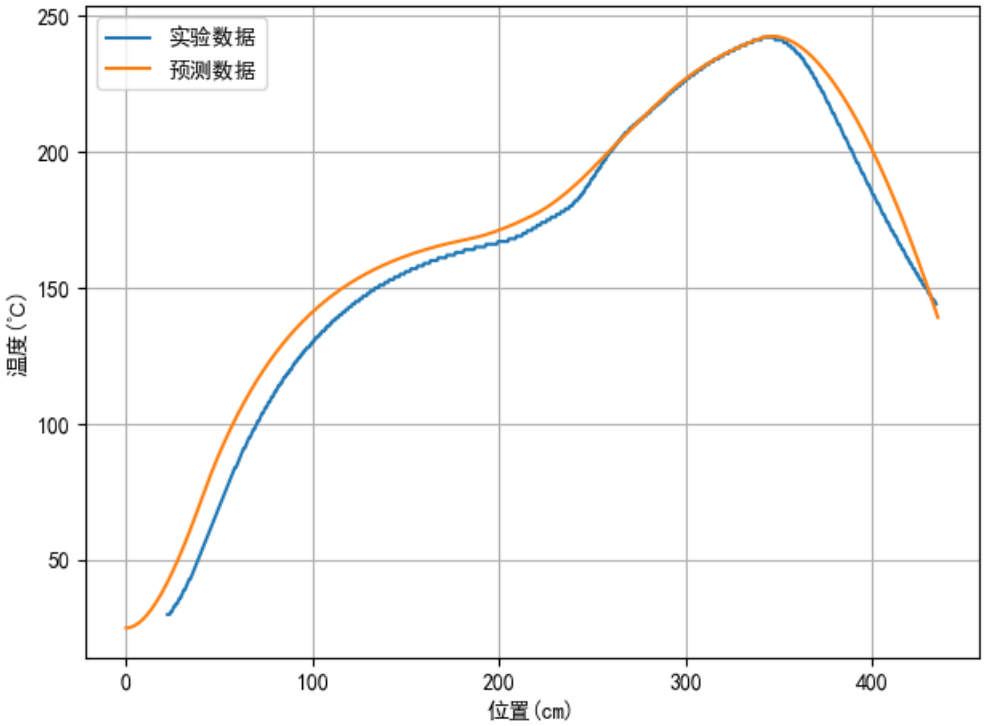

图 11 预测与实验数据对比图

由于在解决问题时，计算图形面积采用了微元法，将面积划分成多个小部分然后累加，导致理论上会存在误差，但在实际验证中发现误差比较微小，结果比较准确。

本文的分析与求解过程可应用在实际工业生产中，在设定小温区的温度之后可以更加准确的预测炉内空气温度并对温度进行调节，以获得更高的产品质量。

七、 参考文献

[1] 张靖周，常海萍，谭晓茗. 传热学（第三版）[M]. 北京: 科学出版社, 2019: 57-61.

[2] D.C Whalley. A simplified reflow soldering process model[J]. Journal of Materials Processing Technology, 2004, 150(1):134-144.

[3] 汤宗健,谢炳堂,梁革英.回流焊炉温曲线的管控分析[J]. 电子质量, 2020(08): 15-19+23.

八、 附录

注：程序源文件同时也保存在支撑材料中。

程序 1：求空气温度随位置变化的曲线

```
import matplotlib.pyplot as plt
import numpy as npy
import math

def func(t1,t2,x1,x2):
    l = 0.024356
    b = 0.0072 / (100 * l)
    c1 = l * (t1 - t2) * (b * (t1 + t2) / 2 + 1) / (x1 - x2)
    c2 = (x2 * t1 * (b * t1 / 2 + 1) - x1 * t2 * (b * t2 / 2 + 1)) / (x2 - x1)
    t = []
    x = []
    tmp = 0
    while tmp < x2 - x1:
        x.append(tmp)
        tmp += 0.005
    for i in x:
        root = (-1 + math.sqrt(1 + 2 * b * c1 * i / l + 2 * b * c2)) / b
        t.append(root)
    return t


dist = 0
x = []
t = []

tmp = 0
while tmp < 40.25:
```

```
        x.append(dist)
        dist += 0.005
        tmp += 0.5
t.extend(func(25,175,0,0.4025))

tmp = 0
while tmp < 142.5:
        x.append(dist)
        dist += 0.005
        tmp += 0.5
t.extend(func(175,175,0,1.42))

#5-6
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += 0.005
        tmp += 0.5
t.extend(func(175,195,0,0.355))

#6-7
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += 0.005
        tmp += 0.5
t.extend(func(195,235,0,0.355))

#7-8
tmp = 0
```

```
while tmp < 35.5:
        x.append(dist)
        dist += 0.005
        tmp += 0.5
t.extend(func(235,255,0,0.355))

#8-9
tmp = 0
while tmp < 50.75:
        x.append(dist)
        dist += 0.005
        tmp += 0.5
t.extend(func(255,255,0,0.5075))

tmp = 0
while tmp < 96:
        x.append(dist)
        dist += 0.005
        tmp += 0.5
t.extend(func(255,25,0,0.96))



tt = npy.array(t)
xx = npy.array(x) * 100

plt.axvline(40.25, color='red', linestyle='--')
plt.axvline(182.75, color='red', linestyle='--')
plt.axvline(218.25, color='red', linestyle='--')
plt.axvline(253.75, color='red', linestyle='--')
plt.axvline(289.25, color='red', linestyle='--')
```

```
plt.axvline(340, color='red', linestyle='--')

plt.plot(xx,tt)
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.xlabel("位置(cm)")
plt.ylabel("空气温度(℃)")
plt.show()
```

程序 2：求系数 CT

```
import matplotlib.pyplot as plt
import numpy as npy
import math

T0 = 25
T15 = 175
T6 = 195
T7 = 235
T89 = 255
v = 70
deltaS = 13 / 400
deltat = deltaS / (v / 60)

def func(t1,t2,x1,x2):
    l = 0.024356
    b = 0.0072 / (100 * l)
    c1 = l * (t1 - t2) * (b * (t1 + t2) / 2 + 1) / (x1 - x2)
    c2 = (x2 * t1 * (b * t1 / 2 + 1) - x1 * t2 * (b * t2 / 2 + 1)) / (x2 - x1)
    t = []
    x = []
```

```
    tmp = 0
    while tmp < x2 - x1:
        x.append(tmp)
        tmp += deltaS / 100
    for i in x:
        root = (-1 + math.sqrt(1 + 2 * b * c1 * i / l + 2 * b * c2)) / b
        t.append(root)
    return t


def calculate_max_t(l, r):
    h = (l + r) / 2

    dist = 0
    x = []
    t_air = []

    tmp = 0
    while tmp < 40.25:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
    t_air.extend(func(T0, T15, 0, 0.4025))

    tmp = 0
    while tmp < 142:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
    t_air.extend(func(T15, T15, 0, 1.42))
```

```
# 5-6
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T15, T6, 0, 0.355))

# 6-7
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T6, T7, 0, 0.355))

# 7-8
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T7, T89, 0, 0.355))

tmp = 0
while tmp < 50.75:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T89, T89, 0, 0.5075))
```

```
        tmp = 0
        while tmp < 96:
            x.append(dist)
            dist += deltaS * 0.01
            tmp += deltaS
        t_air.extend(func(T89, T0, 0, 0.96))

        tt = npy.array(t_air)
        xx = npy.array(x) * 100

        T_now = 25
        Ts = []

        for i in range(len(x)):
            deltaT = t_air[i] - T_now
            T_now += (deltaS / (v / 60)) * h * deltaT
            Ts.append(T_now)
        return max(Ts)

maxT = 0
l = 0.015
r = 0.025
while abs(maxT - 242.28) > 0.01:
        maxT = calculate_max_t(l, r)
        if maxT - 242.28 > 0:
            r = (l + r) / 2
        else:
            l = (l + r) / 2
print((l+r) / 2)
```

程序 3：求炉温曲线

```
import matplotlib.pyplot as plt
import numpy as npy
import math

def func(t1,t2,x1,x2):
    l = 0.024356
    b = 0.0072 / (100 * l)
    c1 = l * (t1 - t2) * (b * (t1 + t2) / 2 + 1) / (x1 - x2)
    c2 = (x2 * t1 * (b * t1 / 2 + 1) - x1 * t2 * (b * t2 / 2 + 1)) / (x2 - x1)
    t = []
    x = []
    tmp = 0
    while tmp < x2 - x1:
        x.append(tmp)
        tmp += 0.005
    for i in x:
        root = (-1 + math.sqrt(1 + 2 * b * c1 * i / l + 2 * b * c2)) / b
        t.append(root)
    return t

dist = 0
x = []
t = []

T0 = 25
T15 = 173
T6 = 198
T7 = 230
```

```
T89 = 257

v = 78

tmp = 0
while tmp < 40.25:
    x.append(dist)
    dist += 0.005
    tmp += 0.5
t.extend(func(T0,T15,0,0.4025))

tmp = 0
while tmp < 142:
    x.append(dist)
    dist += 0.005
    tmp += 0.5
t.extend(func(T15,T15,0,1.42))

#5-6
tmp = 0
while tmp < 35.5:
    x.append(dist)
    dist += 0.005
    tmp += 0.5
t.extend(func(T15,T6,0,0.355))

#6-7
tmp = 0
while tmp < 35.5:
    x.append(dist)
```

```
        dist += 0.005
        tmp += 0.5
t.extend(func(T6,T7,0,0.355))

#7-8
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += 0.005
        tmp += 0.5
t.extend(func(T7,T89,0,0.355))

#8-9
tmp = 0
while tmp < 50.75:
        x.append(dist)
        dist += 0.005
        tmp += 0.5
t.extend(func(T89,T89,0,0.5075))

tmp = 0
while tmp < 96:
        x.append(dist)
        dist += 0.005
        tmp += 0.5
t.extend(func(T89,T0,0,0.96))

tt = npy.array(t)
xx = npy.array(x) * 100
```

```
h = 0.0216
T_now = 25
Ts = []
for i in range(len(x)):
    deltaT = t[i] - T_now
    T_now += (0.5 / (v / 60)) * h * deltaT
    Ts.append(T_now)

a = 0
b = 0
for i in range(0,len(xx)):
    if xx[i] - 111.0 < 0.000001:
        a = Ts[i]
    if xx[i] - 111.5 < 0.000001:
        b = Ts[i]
print("小温区 3 中点温度为：")
print("{:.2f}".format((a + b) / 2))

a = 0
b = 0
for i in range(0,len(xx)):
    if xx[i] - 217.5 < 0.000001:
        a = Ts[i]
    if xx[i] - 218 < 0.000001:
        b = Ts[i]
print("小温区 6 中点温度为：")
print("{:.2f}".format((a + b) / 2))

a = 0
b = 0
```

```
for i in range(0,len(xx)):
    if xx[i] - 253 < 0.000001:
        a = Ts[i]
    if xx[i] - 253.5 < 0.000001:
        b = Ts[i]
print("小温区 7 中点温度为：")
print("{:.2f}".format((a + b) / 2))

a = 0
for i in range(0,len(xx)):
    if xx[i] - 304 < 0.000001:
        a = Ts[i]
print("小温区 8 结束处温度为：")
print("{:.2f}".format(a))

plt.plot(xx,npy.array(Ts))
plt.xlabel("位置(cm)")
plt.ylabel("温度(℃)")
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False
plt.grid('true')
plt.show()
```

程序 4：将输出结果粘贴到 CSV 文件中

```
import numpy as npy
import math

dist = 0

T0 = 25
```

```
T15 = 173
T6 = 198
T7 = 230
T89 = 257

v = 78

deltaS = 13 / 400

def func(t1,t2,x1,x2):
    l = 0.024356
    b = 0.0072 / (100 * l)
    c1 = l * (t1 - t2) * (b * (t1 + t2) / 2 + 1) / (x1 - x2)
    c2 = (x2 * t1 * (b * t1 / 2 + 1) - x1 * t2 * (b * t2 / 2 + 1)) / (x2 - x1)
    t = []
    x = []
    tmp = 0
    while tmp < x2 - x1:
        x.append(tmp)
        tmp += deltaS / 100
    for i in x:
        root = (-1 + math.sqrt(1 + 2 * b * c1 * i / l + 2 * b * c2)) / b
        t.append(root)
    return t


x = []
t_air = []

tmp = 0
```

```
while tmp < 40.25:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T0, T15, 0, 0.4025))

tmp = 0
while tmp < 142:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T15, T15, 0, 1.42))

#5-6
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T15, T6, 0, 0.355))

#6-7
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T6, T7, 0, 0.355))

#7-8
```

```
tmp = 0
while tmp < 35.5:
    x.append(dist)
    dist += deltaS * 0.01
    tmp += deltaS
t_air.extend(func(T7, T89, 0, 0.355))

#8-9
tmp = 0
while tmp < 50.75:
    x.append(dist)
    dist += deltaS * 0.01
    tmp += deltaS
t_air.extend(func(T89, T89, 0, 0.5075))

tmp = 0
while tmp < 96:
    x.append(dist)
    dist += deltaS * 0.01
    tmp += deltaS
t_air.extend(func(T89, T0, 0, 0.96))

tt = npy.array(t_air)
xx = npy.array(x) * 100

h = 0.0216
T_now = 25
Ts = []
for i in range(len(x)):
    deltaT = t_air[i] - T_now
```

```
    T_now += (deltaS / (v / 60)) * h * deltaT
    Ts.append(T_now)

cnt = 0

for i in Ts:
    if cnt % 20 == 0:
        print("{:.6f}".format(i))
    cnt += 1
```

程序 5：求最大过炉速度

```
import numpy as npy
import math

T0 = 25
T15 = 182
T6 = 203
T7 = 237
T89 = 254

deltaS = 13 / 400

def func(t1,t2,x1,x2):
    l = 0.024356
    b = 0.0072 / (100 * l)
    c1 = l * (t1 - t2) * (b * (t1 + t2) / 2 + 1) / (x1 - x2)
    c2 = (x2 * t1 * (b * t1 / 2 + 1) - x1 * t2 * (b * t2 / 2 + 1)) / (x2 - x1)
    t = []
    x = []
    tmp = 0
```

```
    while tmp < x2 - x1:
        x.append(tmp)
        tmp += deltaS / 100
    for i in x:
        root = (-1 + math.sqrt(1 + 2 * b * c1 * i / 1 + 2 * b * c2)) / b
        t.append(root)
    return t


def ask_if_true(v):
    #print(v)
    x = []
    t_air = []
    dist = 0
    deltat = deltaS / (v / 60)

    tmp = 0
    while tmp < 40.25:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
    t_air.extend(func(T0, T15, 0, 0.4025))

    tmp = 0
    while tmp < 142:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
    t_air.extend(func(T15, T15, 0, 1.42))

    #5-6
```

```
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T15, T6, 0, 0.355))

#6-7
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T6, T7, 0, 0.355))

#7-8
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T7, T89, 0, 0.355))

tmp = 0
while tmp < 50.75:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T89, T89, 0, 0.5075))
```

```
tmp = 0
while tmp < 96:
    x.append(dist)
    dist += deltaS * 0.01
    tmp += deltaS
t_air.extend(func(T89, T0, 0, 0.96))

tt = npy.array(t_air)
xx = npy.array(x) * 100

h = 0.0216
T_now = 25
Ts = []
maxk = 0

t_150 = 0
t_190 = 0

t_217_1 = 0
t_217_2 = 0

for i in range(len(x)):
    deltaT = t_air[i] - T_now
    T_prev = T_now
    T_now += (deltaS / (v / 60)) * h * deltaT
    if T_now >= 150 and t_150 == 0:
        t_150 = xx[i] / (v / 60)
    if T_now <= 190 and (deltaS / (v / 60)) * h * deltaT > 0:
        t_190 = xx[i] / (v / 60)
    k = abs((deltaS / (v / 60)) * h * deltaT) / deltat
```

```
            if T_prev <= 217 and T_now >= 217:
                t_217_1 = xx[i] / (v / 60)
            if T_prev >= 217 and T_now <= 217:
                t_217_2 = xx[i] / (v / 60)
            if k > maxk:
                maxk = k
            Ts.append(T_now)

        cnt = 0
        '''print("max k:")
        print(maxk)
        print("delta t:")
        print(t_190 - t_150)
        print("217 time:")
        print(t_217_2 - t_217_1)
        print("max t:")
        print(max(Ts))'''

        if maxk <= 3 and 120 >= t_190 - t_150 >= 60 and 40 <= t_217_2 - t_217_1
<= 90 and 240 <= max(Ts) <= 250:
            return True
        else :
            return False

    max_v = 0
    tmp_v = 65
    while tmp_v <= 100:
        if ask_if_true(tmp_v):
            if tmp_v > max_v:
                max_v = tmp_v
```

```
            print(tmp_v)
        tmp_v += 0.01
print(max_v)
```

程序 6：求最小面积

```
import numpy as npy
import math
from sko.GA import GA

T0 = 25

deltaS = 13 / 400

def func(t1,t2,x1,x2):
    l = 0.024356
    b = 0.0072 / (100 * l)
    c1 = l * (t1 - t2) * (b * (t1 + t2) / 2 + 1) / (x1 - x2)
    c2 = (x2 * t1 * (b * t1 / 2 + 1) - x1 * t2 * (b * t2 / 2 + 1)) / (x2 - x1)
    t = []
    x = []
    tmp = 0
    while tmp < x2 - x1:
        x.append(tmp)
        tmp += deltaS / 100
    for i in x:
        root = (-1 + math.sqrt(1 + 2 * b * c1 * i / l + 2 * b * c2)) / b
        t.append(root)
    return t

def get_area(T15, T6, T7, T89, v):
```

```
dist = 0
x = []
t_air = []

deltat = deltaS / (v / 60)

tmp = 0
while tmp < 40.25:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T0, T15, 0, 0.4025))

tmp = 0
while tmp < 142:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T15, T15, 0, 1.42))

#5-6
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T15, T6, 0, 0.355))

#6-7
tmp = 0
```

```
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T6, T7, 0, 0.355))

#7-8
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T7, T89, 0, 0.355))

tmp = 0
while tmp < 50.75:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T89, T89, 0, 0.5075))

tmp = 0
while tmp < 96:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T89, T0, 0, 0.96))


tt = npy.array(t_air)
```

```
xx = npy.array(x) * 100


h = 0.0216
T_now = 25
Ts = []
maxk = 0


t_150 = 0
t_190 = 0


t_217_1 = 0
t_217_2 = 0


integrate = False
area = 0


for i in range(len(x)):
    deltaT = t_air[i] - T_now
    T_prev = T_now
    T_now += (deltaS / (v / 60)) * h * deltaT
    if T_now >= 150 and t_150 == 0:
        t_150 = xx[i] / (v / 60)
    if T_now <= 190 and (deltaS / (v / 60)) * h * deltaT > 0:
        t_190 = xx[i] / (v / 60)
    k = abs((deltaS / (v / 60)) * h * deltaT) / deltat
    if T_prev <= 217 and T_now >= 217:
        t_217_1 = xx[i] / (v / 60)
        integrate = True
    if T_prev > T_now:
        integrate = False
```

```
            if T_prev >= 217 and T_now <= 217:
                t_217_2 = xx[i] / (v / 60)
            if k > maxk:
                maxk = k
            if integrate:
                area += (T_now - 217) * deltat
            Ts.append(T_now)

        if maxk <= 3 and 120 >= t_190 - t_150 >= 60 and 40 <= t_217_2 - t_217_1
<= 90 and 240 <= max(Ts) <= 250:
            return area
        else:
            return 99999

    ga = GA(func=get_area, n_dim=5, size_pop=200, max_iter=10, lb=[165, 185,
225, 245, 65], ub=[185, 205, 245, 265, 100], precision=1e-7)
    x, y = ga.run()

    print("小温区 1~5 设定温度：",x[0])
    print("小温区 6 设定温度：",x[1])
    print("小温区 7 设定温度：",x[2])
    print("小温区 8~9 设定温度：",x[3])
    print("传送带过炉速度为：",x[4])
    print("面积为：",y)
```

程序 7：双目标优化

```
import numpy as npy
import math
from sko.GA import GA
```

```
T0 = 25

deltaS = 13 / 400


def func(t1,t2,x1,x2):
    l = 0.024356
    b = 0.0072 / (100 * l)
    c1 = l * (t1 - t2) * (b * (t1 + t2) / 2 + 1) / (x1 - x2)
    c2 = (x2 * t1 * (b * t1 / 2 + 1) - x1 * t2 * (b * t2 / 2 + 1)) / (x2 - x1)
    t = []
    x = []
    tmp = 0
    while tmp < x2 - x1:
        x.append(tmp)
        tmp += deltaS / 100
    for i in x:
        root = (-1 + math.sqrt(1 + 2 * b * c1 * i / l + 2 * b * c2)) / b
        t.append(root)
    return t


def get_iou(T15, T6, T7, T89, v):
    dist = 0
    x = []
    t_air = []

    deltat = deltaS / (v / 60)

    tmp = 0
    while tmp < 40.25:
```

```
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T0, T15, 0, 0.4025))

tmp = 0
while tmp < 142:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T15, T15, 0, 1.42))

#5-6
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T15, T6, 0, 0.355))

#6-7
tmp = 0
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T6, T7, 0, 0.355))

#7-8
tmp = 0
```

```
while tmp < 35.5:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T7, T89, 0, 0.355))

tmp = 0
while tmp < 50.75:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T89, T89, 0, 0.5075))

tmp = 0
while tmp < 96:
        x.append(dist)
        dist += deltaS * 0.01
        tmp += deltaS
t_air.extend(func(T89, T0, 0, 0.96))



tt = npy.array(t_air)
xx = npy.array(x) * 100

h = 0.0216
T_now = 25
Ts = []
maxk = 0

t_150 = 0
```

```
t_190 = 0


t_217_1 = 0
t_217_2 = 0

integrate = False
area = 0

left = 0
mid = 0
right = 0

for i in range(len(x)):
    deltaT = t_air[i] - T_now
    T_prev = T_now
    T_now += (deltaS / (v / 60)) * h * deltaT
    if T_now >= 150 and t_150 == 0:
        t_150 = xx[i] / (v / 60)
    if T_now <= 190 and (deltaS / (v / 60)) * h * deltaT > 0:
        t_190 = xx[i] / (v / 60)
    k = abs((deltaS / (v / 60)) * h * deltaT) / deltat
    if T_prev <= 217 and T_now >= 217:
        t_217_1 = xx[i] / (v / 60)
        integrate = True
    if T_prev > T_now:
        integrate = False
    if T_prev >= 217 and T_now <= 217:
        t_217_2 = xx[i] / (v / 60)
    if k > maxk:
        maxk = k
```

```
            if integrate:
                area += (T_now - 217) * deltat
            if T_now >= 217 and T_now > T_prev and left == 0 :
                left = i
            if T_now >= 217 and T_now < T_prev and mid == 0 :
                mid = i - 1
            if T_now <= 217 and T_now < T_prev and right == 0:
                right = i
            Ts.append(T_now)

        l = r = 0
        big = small = 0
        bigSum = smallSum = 0
        for i in range(0, max(mid - left, right - mid)) :
            l = mid - i
            r = mid + i
            big = max(max(Ts[l], Ts[r]) - 217, 0)
            small = max(min(Ts[l], Ts[r]) - 217, 0)
            bigSum += big * deltat
            smallSum += small * deltat

        if maxk <= 3 and 120 >= t_190 - t_150 >= 60 and 40 <= t_217_2 - t_217_1
<= 90 and 240 <= max(Ts) <= 250:
            return area * bigSum / smallSum
        else:
            return 99999


    ga = GA(func=get_iou, n_dim=5, size_pop=200, max_iter=10, lb=[165, 185,
225, 245, 65], ub=[185, 205, 245, 265, 100], precision=1e-7)
```

```
x, y = ga.run()

print("小温区 1~5 设定温度：",x[0])
print("小温区 6 设定温度：",x[1])
print("小温区 7 设定温度：",x[2])
print("小温区 8~9 设定温度：",x[3])
print("传送带过炉速度为：",x[4])

print("面积为：",y)
```