接触式轮廓仪的自动标注

朱宇豪、喻健凯、黄萱

8/2025

摘要

本文针对工件 1 轮廓线参数确定问题，建立了一套基于数据拟合与几何分析的参数求解方法。

首先，基于附件一 sheet1 的测量数据，通过 MATLAB 绘制散点图并采用交互式方法初步识别轮廓线转折点。考虑到轮廓线由直线段和圆弧构成的特点，采用分段最小二乘拟合法，在初步识别区间内进行精细化拟合，建立轮廓线的精确数学模型，进而通过解析几何方法确定最终转折点坐标及相关参数。

针对 sheet2 中工件在倾斜状态下的 17 组测量数据，我们提出基于基准线校正的数据处理方法：以水平方向为基准，计算各测量数据的倾斜角度并进行水平校正。通过整合校正后的全部数据点，采用最小二乘法进行整体拟合，可以显著提高参数估计的精度。

对于不同倾斜角度下的 10 次重复测量数据，我们提出以最长直线段为基准的标准化处理方法：计算各次测量的相对倾斜角，通过求取平均值消除系统误差，建立标准参考角度体系，从而获得更为可靠的参数估计值。

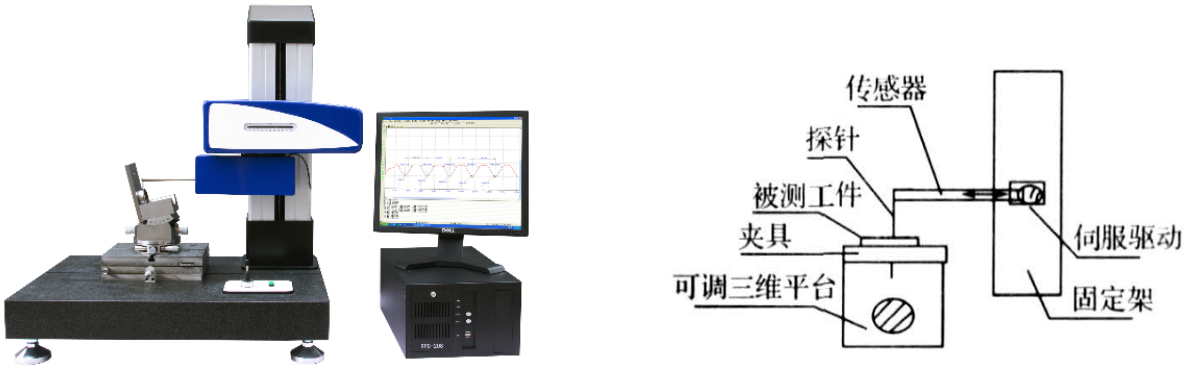
此外，针对 9 组局部测量数据，我们开发了基于特征点匹配的精度提升算法：通过选取关键特征点并进行平移配准，实现多组数据的空间对齐，在此基础上进行融合拟合，进一步提高了参数确定的准确性。该方法为类似工件的几何参数测量提供了可靠的技术方案。

关键词：轮廓线拟合　最小二乘法　几何参数测量　数据校正　特征点

1 问题重述与分析

1.1 问题重述

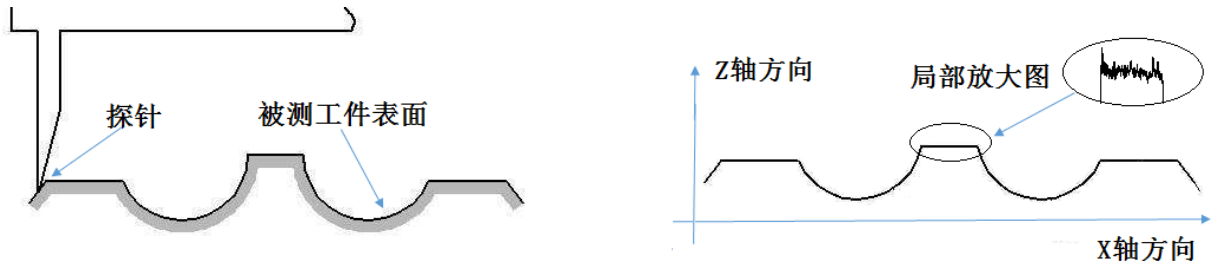
轮廓仪是一种两坐标测量仪器（见图 1(a)），它由工作平台、夹具、被测工件、探针、传感器和伺服驱动等部件组成（见图 1(b)）。



(a) 某种型号的接触式轮廓仪　　**(b)** 接触式轮廓仪的示意图

图 1

接触式轮廓仪的工作原理是，探针接触到被测工件表面并匀速滑行，传感器感受到被测表面的几何变化，在 X 和 Z 方向分别采样，并转换成电信号。该电信号经放大等处理，转换成数字信号储存在数据文件中（见图 2）。



(a) 接触式轮廓仪测量示意图　　**(b)** 数据文件中的数字信号

图 2: 接触式轮廓仪的工作原理

在理想状况下，轮廓曲线应该是光滑的，但由于接触式轮廓仪存在探针沾污、探针缺陷、扫描位置不准等问题，检测到的轮廓曲线呈现出粗糙不平的情况（见图 2(b) 中的局部放大图），这给工件形状的准确标注带来影响。为了简化问题，假设被测工件的轮廓线是由直线和圆弧构成的平面曲线（见图 3）。请建立数学模型，并根据附件 1（工件 1 的水平和倾斜测量数据）、附件 2 ~ 附件 4（工件 2 的多次测量数据）所提供的轮廓仪测量数据，研究下列问题：

问题 1.　附件 1 中的表 level 是工件 1 在水平状态下的测量数据，其轮廓线如图 4 所示，请标注出轮廓线的各项参数值：槽口宽度（如 x_1,x_3 等）、圆弧半径（如 R_1,R_3 等）、

圆心之间的距离（如 c_1, c_2 等）、圆弧的长度、水平线段的长度（如 x_2, x_4 等）、斜线线段的长度、斜线与水平线之间的夹角（如 $\angle 1$,$\angle 2$ 等）和人字形线的高度（z_1）。

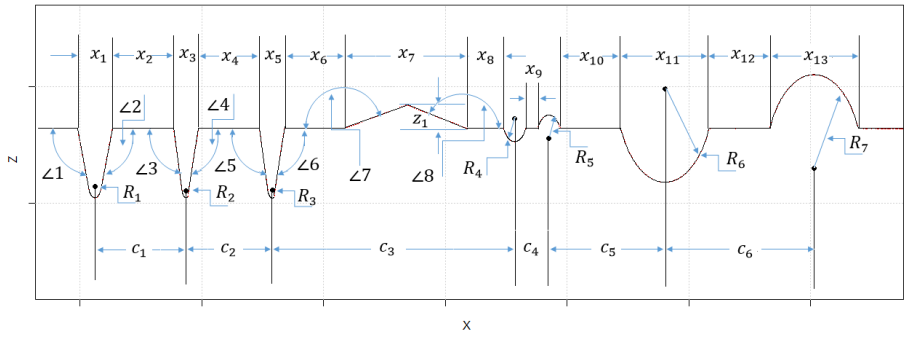


图 3: 工件 1 在水平状态下测量的轮廓线

问题 2. 同一工件在不同次测量时，由于工件放置的角度和位置不同，轮廓线参数的计算值也会存在差异。附件 1 中的表 down 给出了工件 1 在倾斜一个角度和有一些水平位移状态下轮廓线的测量数据。请计算该工件测量时的倾斜角度，并作水平校正。在数据校正后，完成问题 1 的任务，并比较两种测量状态下工件 1 各项参数计算值之间的差异。

问题 3. 在对工件作多次检测时，工件每次放置的角度、测量的起点和终点都会有偏差，这导致了每次测量实际是对整个工件中的某一部分进行检测。附件 2 提供了对工件 2 的 10 次测量数据，请基于这些数据完成：(1) 每次测量时工件 2 的倾斜角度；(2) 标注出工件 2 轮廓线的各项参数值（同问题 1）；(3) 画出工件 2 的完整轮廓线。

问题 4. 为了更准确地标注出工件 2 的各项参数值，附件 3 和附件 4 分别提供了工件 2 关于圆和角的 9 次局部测量数据，请利用这些数据修正问题 3 的结论，并对该工件的完整轮廓线作进一步修正。

1.2 问题分析

问题一: 已经假设工件一轮廓线只由直线段与圆弧构成，于是我们主要采用曲线拟合的方法，利用 matlab 画出附件一水平的散点图，人工初步选取大区间的 17 个分段点，分区间利用最小二乘法以及 LM 算法进行曲线的拟合，得到工件一在水平情况下的分段表达式，为了弥补人工选取分段点的缺陷，我们将 17 个相邻拟合曲线的交点作为正式的间段点 (特殊点)，再次基础上利用解析几何方法求得工件一水平情况下的参数

问题二: 为了确定工件一的偏转角度与水平位移，我们采用两种方法。

方法一: 利用 matlab 画出附件一 sheet2 的散点图，同问题一一样人工选取 17 个初步间段点，以此为区间端点分区间进行曲线的拟合，最终以曲线的 17 个交点作为正式的间段点。对于水平状态以及位移后状态的工件一，我们将两组正式的间段点作为工件一进

行钢体位移的代表, 以对应特殊点之间的距离为基础，通过最小二乘法确立最优的偏转角度 θ 与水平位移 Δx

方法二: 利用 matlab 画出附件一 sheet2 的散点图，同方法一一样人工选取 17 个初步间段点，以此为区间端点进行分段曲线的拟合，以工件一对应的几组直线作为工件一钢体运动的代表，分别求出其对应的偏转角，最终以平均偏转角作为整个工件的偏转角，将倾斜的工件一水平校正后，通过分段点的水平位移之差求得整个工件一的水平位移。

问题三: 我们主要利用问题二中方法二的思想对问题三进行处理。对十组数据先画出它们的散点图，再使用问题二的方法对它们分别进行分段曲线的拟合，选取它们相对应的直线段计算偏转角与水平位移，在水平校正以及水平平移后将十组分段函数求和取平均作为对真实的附件一水平状态的数学表示。利用问题一的模型进一步求得其参数。

问题四：针对多次局部测量数据融合问题，我们需解决测量误差和坐标系不统一的影响。通过提取特征点进行数据配准，结合最小二乘平差和加权融合方法，在保持几何约束的条件下优化参数估计，从而提高轮廓线关键参数的测量精度。

2 模型假设与符号说明

2.1 模型假设

1. **轮廓分段性假设**：假设工件轮廓线可以分解为有限个直线段和圆弧段的组合，且各段之间在转折点处连续。

2. **刚体运动假设**：假设工件在测量过程中仅发生平移和旋转，不产生形变，满足刚体运动特性：

$$\begin{cases} x' = x\cos\theta - z\sin\theta + \Delta x \\ z' = x\sin\theta + z\cos\theta + \Delta z \end{cases}$$

3. **数据可靠性假设**：假设测量数据中噪声服从高斯分布，异常值占比不超过 5%，可通过标准差准则剔除：

$$|z_i - \bar{z}| > 3\sigma \Rightarrow \text{剔除}$$

4. **局部线性化假设**：对于任意足够小的轮廓段 $[s_i, s_{i+1}]$，存在线性近似：

$$\exists a, b \in \mathbb{R}, \max_{x\in[s_i,s_{i+1}]} |z(x) - (ax+b)| < \varepsilon$$

5. **圆弧光滑性假设**：圆弧段曲率 k 满足 Lipschitz 连续条件：

$$|k(s_1) - k(s_2)| \le L|s_1 - s_2|,\ \forall s_1, s_2 \in \text{圆弧段}$$

6. **特征不变性假设**：关键特征点（如曲率极值点）在位移前后保持拓扑关系不变：

$$\text{Neighbor}(p_i) \leftrightarrow \text{Neighbor}(q_i)$$

7. **小角度近似假设**：当偏转角 $\theta < 5^\circ$ 时，允许采用近似计算：

$$\sin\theta \approx \theta,\ \cos\theta \approx 1 - \frac{\theta^2}{2}$$

2.2 符号说明

表 1: 自定义符号说明表

符号	含义
a_i	转折点编号 i 的横坐标，$i \in [1,17] \cap \mathbb{Z}$
(x_i, z_i)	第 i 个特征点在工件坐标系中的坐标
θ	工件绕 y 轴的刚性偏转角度（弧度制）
$(\Delta x, \Delta z)$	工件在 x 和 z 方向的平移量
k_i	离散点 p_i 的近似曲率，计算式为 $\frac{\lvert(z_{i+1}-z_i)(x_{i-1}-x_i)-(x_{i+1}-x_i)(z_{i-1}-z_i)\rvert}{[(x_{i+1}-x_i)^2+(z_{i+1}-z_i)^2]^{3/2}}$
R_j	第 j 段圆弧的拟合半径，$j \in \{1,3,5,7\}$ 对应不同圆弧段
(a_j, b_j)	第 j 段圆弧的圆心坐标，通过 LM 算法优化得到
$\angle_k$	第 k 段直线的倾斜角，$\angle_k = \pi - \arctan(a_k)$（$a_k$ 为斜率）
$[a_m, a_n]$	从第 m 个到第 n 个转折点之间的轮廓段
p_i, q_i	分别为基准状态和偏移状态下的匹配特征点对
$\mathcal{E}(\theta, \Delta x, \Delta z)$	刚体变换的最小二乘误差函数：$\sum \lVert q_i - T(p_i)\rVert^2$
$T(\cdot)$	刚体变换算子：$T(p) = \begin{pmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{pmatrix} p + \begin{pmatrix}\Delta x \\ \Delta z\end{pmatrix}$
κ_{th}	曲率特征点检测阈值，取 $\kappa_{\text{th}} = 1.5\bar{\kappa}$
$\mathcal{P}_L, \mathcal{P}_C$	分别表示直线段和圆弧段的点集划分

3 模型建立与求解

3.1 工件轮廓线的分段拟合解析模型

3.1.1 初步转折点选取

将附件一经过 matlab 的处理后其散点图如下:

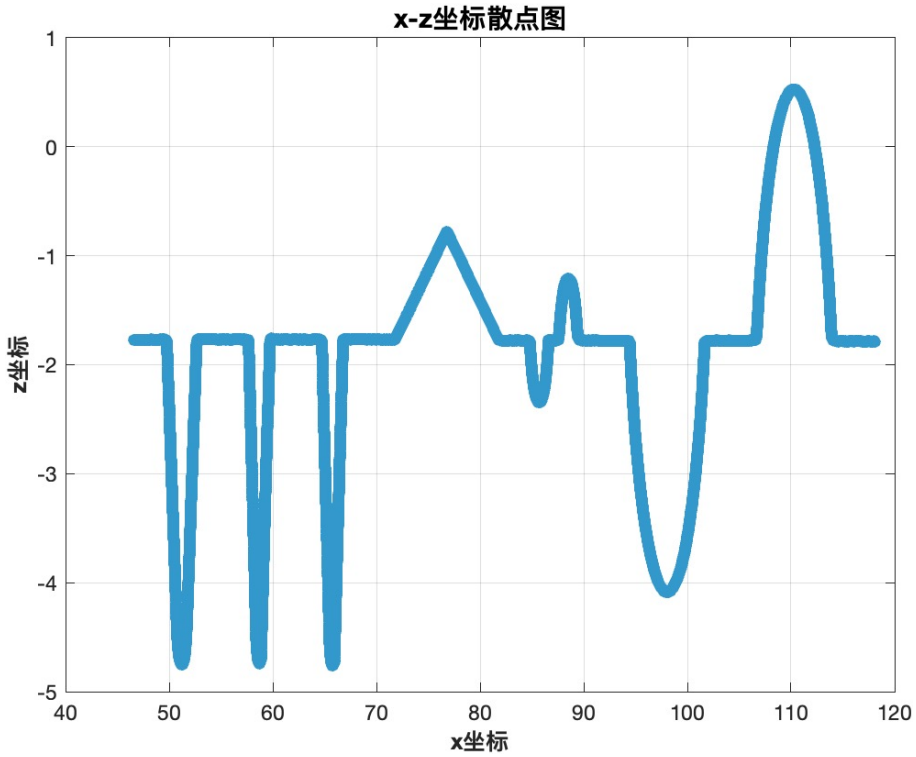


图 4: 水平散点图

检测转折点方式：首先通过绘图手动选取明显的转折点，然后去除掉其 x 轴周围 0.2 的区域，采用区间内部的点进行拟合，最后根据拟合方程计算交点，得到最终转折点。

表 2: 手动粗选的 17 个明细转折点

No.	Position	No.	Position
1	[114.1814, -1.7795]	10	[76.8240, -0.7815]
2	[106.5504, -1.7766]	11	[71.6264, -1.7676]
3	[101.7909, -1.7828]	12	[66.8444, -1.7679]
4	[94.3925, -1.7840]	13	[64.5999, -1.7690]
5	[89.5424, -1.7685]	14	[59.7754, -1.7659]
6	[87.4959, -1.7757]	15	[57.5009, -1.7676]
7	[86.6494, -1.7737]	16	[52.7223, -1.7699]
8	[84.6165, -1.7795]	17	[49.7154, -1.7739]
9	[81.8819, -1.7691]		

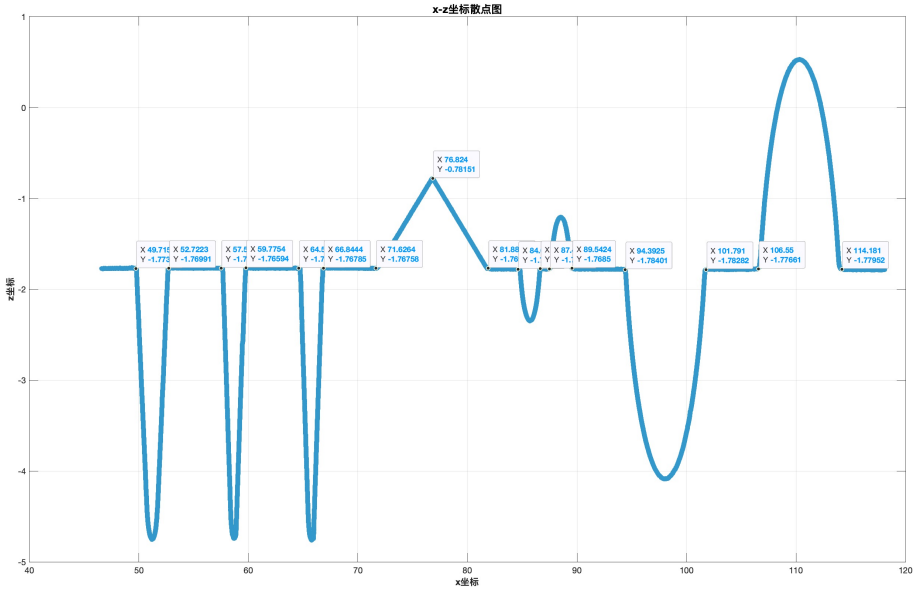


图 5: 初步选取了转折点

直线间交点计算:

$A_i x + B_i z + C_i = 0$,

$\delta = A_1 * B_2 - A_2 * B_1$,

$x = (B_1 * C_2 - B_2 * C_1)/\delta$,

$z = (A_2 * C_1 - A_1 * C_2)/\delta$

线圆间交点计算:

$z = mx + c$, 代入圆方程: $(x-h)^2 + (mx + c - k)^2 = r^2$ 求解二次方程, 并设置 δ 的阈值 0.5, 若 δ 的绝对值 <0.5, 则视为线圆之间仅有一个切点。最终, 我们选取的转折点如下, 我们依次序从左到右对转折点进行编号 (横坐标 $a_i, i \in [1, 17] \cap \mathbf{Z}$):

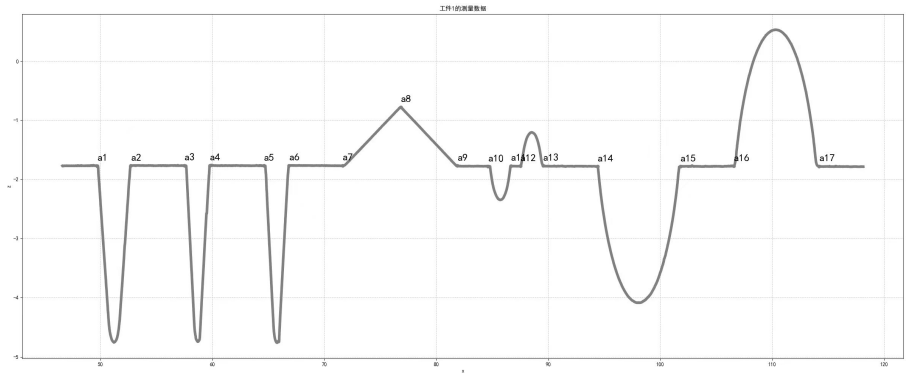


图 6

直线拟合: 根据转折点确定的区间读取数据, 采用 numpy 库的 stats.linregress 函数进行线性拟合。

弧线拟合： 使用数据的质心作为圆心初始估计, 计算到质心的平均距离作为半径初始估计，采用 numpy 库的 $least\text{ - }squares$ 中的 'lm' 方法进行迭代。

混合拟合 ：第三类拟合中的过渡区间选取办法：在对应区间对 z 轴以 0.1 为间隔统计点的个数，选择点的个数急剧变多的区间拟合圆弧，点的个数变化不大的区间拟合直线，再计算交点得到转折点。

经多次区间判断、实际测试及绘图之后，选取得到最合适的拟合区间如下：

i.[a_1,a_2]**：** 选取 z 为 [-4.3,-4.8] 的区域拟合圆弧，取 [-1.9,-3.8] 的区域拟合两端直线。

表 4: Z 值区间分布统计表

区间	点数	区间	点数	区间	点数
[-4.8, -4.7)	888	[-3.5, -3.4)	140	[-2.2, -2.1)	150
[-4.7, -4.6)	528	[-3.4, -3.3)	144	[-2.1, -2.0)	165
[-4.6, -4.5)	302	[-3.3, -3.2)	130	[-2.0, -1.9)	150
[-4.5, -4.4)	229	[-3.2, -3.1)	160	[-1.9, -1.8)	128
[-4.4, -4.3)	152	[-3.1, -3.0)	153	[-1.8, -1.7)	295
[-4.3, -4.2)	156	[-3.0, -2.9)	146	[-1.7, -1.6)	0
[-4.2, -4.1)	125	[-2.9, -2.8)	142		
[-4.1, -4.0)	138	[-2.8, -2.7)	145		
[-4.0, -3.9)	157	[-2.7, -2.6)	144		
[-3.9, -3.8)	161	[-2.6, -2.5)	141		
[-3.8, -3.7)	126	[-2.5, -2.4)	143		
[-3.7, -3.6)	158	[-2.4, -2.3)	140		
[-3.6, -3.5)	146	[-2.3, -2.2)	133		

ii.[a_3,a_4]**：** 选取 z 为 [-4.3,-4.8] 的区域拟合圆弧，取 [-2.1,-4.2] 的区域拟合两端直线。

表 5: Z 值区间分布统计表

z 值区间	点数	z 值区间	点数
[-4.8, -4.7)	638	[-3.6, -3.5)	119
[-4.7, -4.6)	248	[-3.5, -3.4)	111
[-4.6, -4.5)	163	[-3.4, -3.3)	114
[-4.5, -4.4)	119	[-3.3, -3.2)	108

接下页

表 5 Z 值区间分布统计表（续）

z 值区间	点数	z 值区间	点数
[-4.4, -4.3)	104	[-3.2, -3.1)	106
[-4.3, -4.2)	105	[-3.1, -3.0)	113
[-4.2, -4.1)	109	[-3.0, -2.9)	117
[-4.1, -4.0)	113	[-2.9, -2.8)	106
[-4.0, -3.9)	108	[-2.8, -2.7)	106
[-3.9, -3.8)	112	[-2.7, -2.6)	110
[-3.8, -3.7)	112	[-2.6, -2.5)	179
[-3.7, -3.6)	98	[-2.5, -2.4)	115

iii.[a_5,a_6]**：** 选取 z 为 [-4.2,-4.8] 的区域拟合圆弧，取 [-2.1,-4.0] 的区域拟合两端直线。

表 6

z 值区间	点数	z 值区间	点数
[-4.8, -4.7)	741	[-3.6, -3.5)	112
[-4.7, -4.6)	195	[-3.5, -3.4)	111
[-4.6, -4.5)	146	[-3.4, -3.3)	113
[-4.5, -4.4)	115	[-3.3, -3.2)	105
[-4.4, -4.3)	102	[-3.2, -3.1)	113
[-4.3, -4.2)	107	[-3.1, -3.0)	110
[-4.2, -4.1)	110	[-3.0, -2.9)	115
[-4.1, -4.0)	110	[-2.9, -2.8)	110
[-4.0, -3.9)	110	[-2.8, -2.7)	115
[-3.9, -3.8)	107	[-2.7, -2.6)	115
[-3.8, -3.7)	113	[-2.6, -2.5)	104
[-3.7, -3.6)	104	[-2.5, -2.4)	120
[-2.4, -2.3)	87	[-1.9, -1.8)	165
[-2.3, -2.2)	102	[-1.8, -1.7)	367
[-2.2, -2.1)	127	[-1.7, -1.6)	0
[-2.1, -2.0)	93		
[-2.0, -1.9)	146		

在确立了转折点的坐标后，我们以转折点为界分区间对轮廓线进行曲线拟合。

3.1.2 曲线拟合

这里将不同区间内拟合的曲线 (不考虑水平直线) 类型分为三类:
第一类: 直线段，有 $[a_7, a_8], [a_8, a_9]$
第二类: 圆周，有 $[a_{10}, a_{11}], [a_{12}, a_{13}], [a_{14}, a_{15}], [a_{16}, a_{17}]$
第三类: 直线段与圆周，有 $[a_1, a_2], [a_3, a_4], [a_5, a_6]$
这里就三类情况分别进行处理:

第一类:水平直线拟合 以区间 $[a_7, a_8]$ 为例。a_7, a_8 的坐标分别为:(71.6264,-1.7679),(76.8240,-0.7815).
于是区间具体数值为:[71.6264,76.8240]
采用最小二乘法拟合直线: 设直线方程为 $ax + b$.
目标函数为 $minf(a, b) = \sum_j (c_j - ab_j - b)^2$
其中 c_j 为区间 $[a_7, a_8]$ 中附件一 sheet1 内点对应的纵坐标，b_j 为其对应的横坐标。

表 7: 拟合直线方程

区间	拟合直线方程
$(-\infty, a_1]$	$z = 0.001316x - 1.833238$
$[a_2, a_3]$	$z = -0.000223x - 1.755124$
$[a_4, a_5]$	$z = -0.000190x - 1.756219$
$[a_6, a_7]$	$z = -0.000877x - 1.706856$
$[a_7, a_8]$	$z = 0.198677x - 16.033946$
$[a_8, a_9]$	$z = -0.200582x + 14.632147$
$[a_9, a_{10}]$	$z = -0.000371x - 1.746974$
$[a_{11}, a_{12}]$	$z = -0.008780x - 1.013106$
$[a_{13}, a_{14}]$	$z = -0.000470x - 1.735078$
$[a_{15}, a_{16}]$	$z = 0.000252x - 1.806660$
$[a_{17}, \infty)$	$z = -0.000948x - 1.674516$

表 8: 拟合统计指标

区间	R^2	p 值	斜率误差	MAE	RMSE	角度 (°)	弧度
$(-\infty, a_1]$	0.299	0.000	0.000027	0.001244	0.001583	179.925	3.1403
$[a_2, a_3]$	0.032	0.000	0.000013	0.001257	0.001563	179.987	3.1414
$[a_4, a_5]$	0.021	0.000	0.000014	0.001277	0.001646	179.989	3.1414
$[a_6, a_7]$	0.396	0.000	0.000012	0.001129	0.001371	179.950	3.1407
$[a_7, a_8]$	0.999972	0.000	0.000011	0.001204	0.001457	168.763	2.9455
$[a_8, a_9]$	0.999982	0.000	0.000009	0.000897	0.001130	168.658	2.9436
$[a_9, a_{10}]$	0.038	0.000	0.000027	0.001047	0.001251	179.979	3.1412
$[a_{11}, a_{12}]$	0.582	0.000	0.000249	0.000807	0.000958	179.497	3.1328
$[a_{13}, a_{14}]$	0.118	0.000	0.000014	0.001263	0.001649	179.973	3.1411
$[a_{15}, a_{16}]$	0.017	0.000	0.000020	0.001727	0.002394	179.986	3.1413
$[a_{17}, \infty)$	0.249	0.000	0.000020	0.001291	0.001681	179.946	3.1406

第二类:圆弧拟合 以区间 $[a_{10}, a_{11}]$ 为例, 采用 LM 算法. 其对应的区间为 [84.6165,86.6494]
设其满足圆方程:

$$(x-a)^2 + (y-b)^2 = r^2$$

其中 (a,b) 是圆心，r 是半径.
定义残差:

$$f_j(a,b,r) = \sqrt{(b_j-a)^2 + (c_j-b)^2} - r$$

目标函数为:

$$minE(a,b,r) = min \sum_j f_j^2$$

表 9: 圆弧拟合参数

区间	圆心坐标 (h,k)	半径 r	残差平方和 (RSS)	R^2
$[a_{10}, a_{11}]$	$(85.706, -1.350)$	0.996	0.016075	0.999716
$[a_{12}, a_{13}]$	$(88.517, -2.211)$	1.006	0.013345	0.999727
$[a_{14}, a_{15}]$	$(98.050, -0.082)$	4.003	0.092802	0.999980
$[a_{16}, a_{17}]$	$(110.303, -3.478)$	4.005	0.250571	0.999957

表 10: 圆弧拟合统计指标

区间	MAE	RMSE	拟合类型
$[a_{10}, a_{11}]$	0.001707	0.002219	arc_1
$[a_{12}, a_{13}]$	0.001694	0.002013	arc_2
$[a_{14}, a_{15}]$	0.001908	0.002575	arc_3
$[a_{16}, a_{17}]$	0.001980	0.004163	arc_4

第三类：混合区间的拟合 以区间 $[a_1, a_2]$ 为例，其具体数值为 [49.7154,52.7223].
将其分为三部分：第一部分为左侧直线段，第二部分为中间圆弧，第三部分为右侧直线段。
考虑到直线段与圆弧的交点即过渡点难以确认 (工件表面粗糙)，于是在已有其散点图的情况下将三个部分分别在更小的区间内进行拟合，左侧直线选取的拟合区间为 [49.8,50.6], 中间圆弧选取的拟合区间为 [50.9,51.4], 右侧直线选取的拟合区间为 [52.1,52.7] 由此将第三类情形归类为第一类与第二类情形.

表 11: 混合拟合参数（直线 + 圆弧）

区间	拟合类型	方程/圆心	半径/角度	R^2
$[a_1, a_2]$	左侧直线	$z = -2.736517x + 134.435414$	110.074°	0.999818
	右侧直线	$z = 2.618811x - 139.741554$	110.900°	0.994157
	中间圆弧	$(51.218, -4.257)$, $r = 0.495$	-	0.998260
$[a_3, a_4]$	左侧直线	$z = -3.741219x + 213.957262$	104.965°	0.999450
	右侧直线	$z = 3.514363x - 211.547270$	105.884°	0.999947
	中间圆弧	$(58.702, -4.425)$, $r = 0.315$	-	0.931672
$[a_5, a_6]$	左侧直线	$z = -3.642133x + 233.954398$	105.353°	0.997912
	右侧直线	$z = 3.461037x - 232.894114$	106.116°	0.998295
	中间圆弧	$(65.773, -4.467)$, $r = 0.293$	-	0.989821

表 12: 混合拟合统计指标

区间	拟合类型	MAE	RMSE	其他
$[a_1, a_2]$	左侧直线	0.001623	0.002130	斜率误差: 0.001882
	右侧直线	0.009615	0.011024	斜率误差: 0.010179
	中间圆弧	0.001104	0.001480	RSS: 0.002848
$[a_3, a_4]$	左侧直线	0.003380	0.004222	斜率误差: 0.004920
	右侧直线	0.000997	0.001278	斜率误差: 0.001356
	中间圆弧	0.002962	0.003335	RSS: 0.005248
$[a_5, a_6]$	左侧直线	0.004201	0.005151	斜率误差: 0.011666
	右侧直线	0.003522	0.004296	斜率误差: 0.009776
	中间圆弧	0.001419	0.001849	RSS: 0.002072

3.1.3 正式转折点的确定

由于初步确认的转折点依赖于人工干预，无法精确地反映实际情况，于是在分段拟合出曲线方程的基础上，我们将拟合出的相邻的轮廓曲线联立得到交点作为模型的转折点（包括水平直线），从而得到对工件一水平轮廓线一般的数学模型。

选取的转折点具体坐标如下，我们依次序从左到右对转折点进行编号。

表 13: 转折点坐标列表

点编号	a	b	点编号	a	b
1	49.772445	-1.767736	13	89.425324	-1.777153
2	52.685998	-1.766888	14	94.425123	-1.779506
3	57.661756	-1.767999	15	101.674392	-1.780994
4	59.692100	-1.767577	16	106.675598	-1.779732
5	64.721130	-1.768534	17	113.931704	-1.782479
6	66.780180	-1.765454	18	50.745317	-4.430017
7	71.795270	-1.769855	19	51.669307	-4.429409
8	76.807367	-0.774065	20	58.393964	-4.507349
9	81.809127	-1.777330	21	58.858080	-4.698625
10	84.806444	-1.778443	22	65.483778	-4.546202
11	86.608338	-1.773530	23	65.927967	-4.714993
12	87.606839	-1.782297			

3.1.4 参数值的确定

x_i**:** 对于水平线段 x_i，直接利用选取的断点之差确定。

$\angle 1$: 对于角度 $\angle 1$，设第一条斜线的方程为 $z = ax + b$，则

$$\angle 1 = \pi - \arctan a$$

R_1: 对于半径 R_1 则直接由圆的方程得出。

z_1: 高度 z_1 由选取断点的高度之差确立。

c_1: 圆心距离 $\mathbf{c}_1$ 由对应圆的方程确立。

表 14: 圆参数列表

圆编号	**圆心坐标 (h,k)**	**半径 r**	**圆编号**	**圆心坐标 (h,k)**	**半径 r**
O_1	$(51.218281, -4.257183)$	0.495084	O_5	$(88.517301, -2.210691)$	1.006212
O_2	$(58.702350, -4.424920)$	0.314907	O_6	$(98.050106, -0.081529)$	4.002952
O_3	$(65.772786, -4.466850)$	0.292671	O_7	$(110.303008, -3.477862)$	4.005216
O_4	$(85.706230, -1.350278)$	0.996464			

表 15: 转折点横坐标差值

区间	**差值 Δx**	**区间**	**差值 Δx**
(a_1, a_2)	2.914	(a_7, a_9)	10.014
(a_2, a_3)	4.976	(a_9, a_{10})	2.997
(a_3, a_4)	2.030	(a_{11}, a_{12})	0.999
(a_4, a_5)	5.029	(a_{13}, a_{14})	5.000
(a_5, a_6)	2.059	(a_{14}, a_{15})	7.249
(a_6, a_7)	5.015	(a_{15}, a_{16})	5.001
		(a_{16}, a_{17})	7.256

表 16: 圆心横坐标差值

区间	**差值 Δc**	**区间**	**差值 Δc**
(O_1, O_2)	7.484	(O_4, O_5)	2.811
(O_2, O_3)	7.070	(O_5, O_6)	9.533
(O_3, O_4)	19.933	(O_6, O_7)	12.253

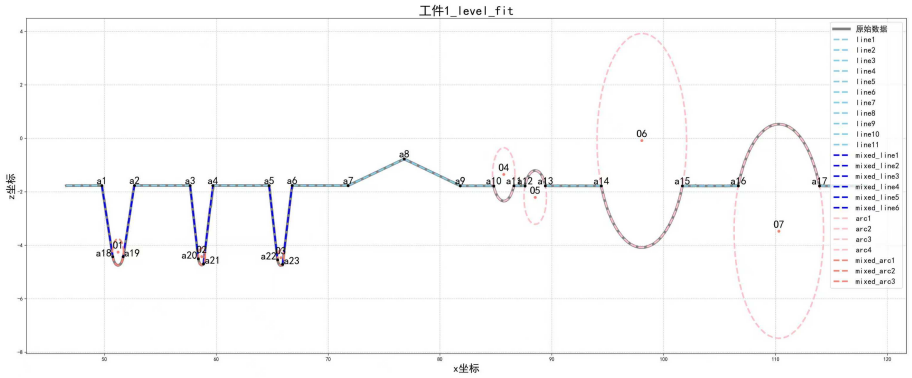


图 7: 拟合总图

3.2 倾角位移校正模型

我们将测量的数据做成图，如下:

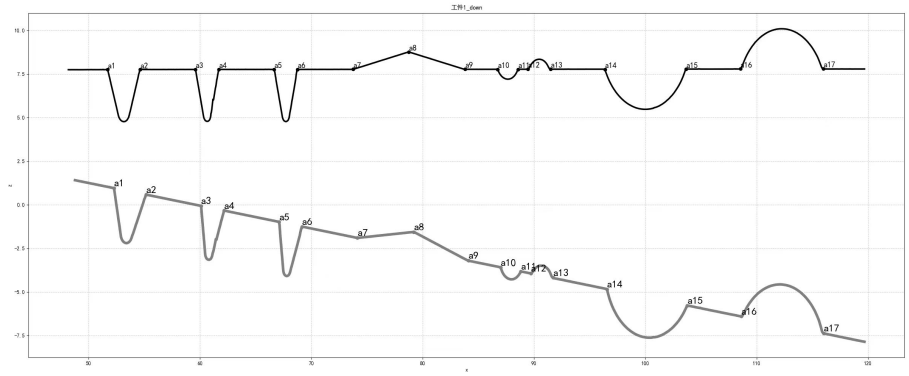


图 8

3.2.1 水平位移和偏转角度的确定

已知刚体的变换公式如下:

$$\begin{pmatrix} x' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta z \end{pmatrix}$$

其中 θ 为旋转角度，$(\Delta x, \Delta z)$ 为位移量
所以接下来的目标即为确定最优的 $\theta, \Delta x, \Delta z$, 即从轮廓线坐标数据反推出工件的刚体运动参数，即找到两组点云之间最优空间变换关系。

我们采用人工干预，利用 Matlab 画出附件一 sheet2 的散点图，选择与第一问中附件一 sheet1 对应的 17 个断点作为与其对应的初步特征点

表 17: 选取的 17 个转折点

No.	Position	No.	Position
1	[74.1437, -1.9026]	10	[84.1239, -3.2049]
2	[116.0250, -7.3731]	11	[79.2127, -1.5718]
3	[108.6285, -6.4093]	12	[69.1762, -1.2660]
4	[103.7796, -5.7907]	13	[67.1081, -0.9914]
5	[96.5639, -4.8494]	14	[62.1717, -0.3424]
6	[91.7218, -4.1919]	15	[60.0920, -0.0733]
7	[89.7228, -3.9415]	16	[55.1646, 0.5689]
8	[88.8237, -3.8364]	17	[52.2842, 0.9468]
9	[87.0161, -3.5982]		

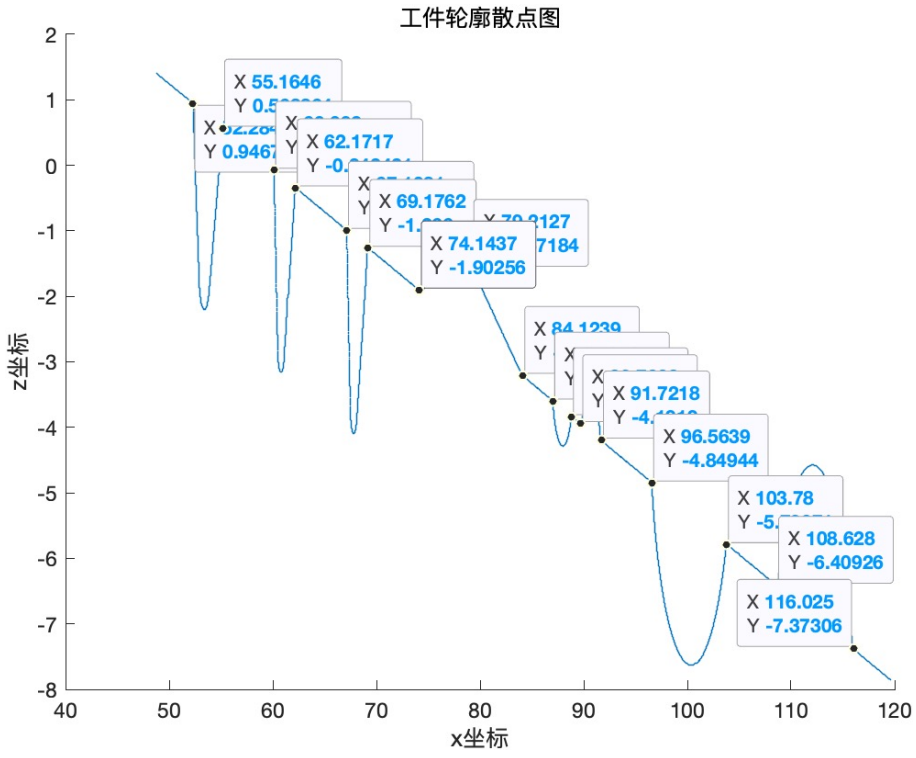


图 9

采用人工选择的特征点进行局部拟合，而后以拟合曲线的交点为最终特征点的方法进行偏转角度和水平位移的确认

参数确定 在两组数据中初步选取 17 个特征点后
利用与问题一相同的方法进行分区间拟合，将相邻拟合曲线的 17 个交点作为正式的特

征点，利用最小二乘法，对两组正式的倾斜点进行偏转角和水平位移的计算

$$\begin{cases} x' = x\cos\theta - z\sin\theta + \Delta x \\ z' = x\sin\theta + z\cos\theta + \Delta z \end{cases}$$

在确立了 sheet1 的特征点集 $\{p_i\}$，sheet2 的特征点集 $\{q_i\} i \in \{1...n\}, n = 17$ 后，进行参数的确定

利用最小二乘法，目标函数为:

$$min_{\theta,\Delta x} \sum_{i=1}^{n} [(x_i' - (x\cos\theta - z\sin\theta + \Delta x))^2 + (z_i' - (x\sin\theta + z\cos\theta + \Delta z)^2)]$$

其中 x_i, x_i', z_i, z_i' 为对应特征点的横纵坐标

利用 python 求得最优的变位参数

3.2.2 水平校正

在求得变位参数 $\theta, \Delta x$ 后，利用反演变换:

$$\begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}^{-1} \begin{pmatrix} x' \\ z' \end{pmatrix} - \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}^{-1} \begin{pmatrix} \Delta x \\ 0 \end{pmatrix}$$

求得附件一 sheet2 的水平校正数据集，带入问题一的模型即求得其对应的参数

3.3 问题二第二种模型建立

3.3.1 偏转位移与角度确定

根据实验数据，首先确定工件的水平位移和偏转角度。以原始数据为基础，绘制带有偏转的散点图，直观展示工件的初始状态。

3.3.2 基准线拟合与倾斜角计算

将水平方向作为基准线，对采集的 17 组直线段数据进行线性拟合。采用一般式直线方程表示, 从而得到倾斜角度。

$ax + by + c = 0$

对全部 17 组直线夹角求算术平均值，得到工件的整体倾斜角度:

$\bar{\theta} = \frac{1}{17} \sum_{i=1}^{17} \theta_i$

3.3.3 坐标旋转校正

通过坐标旋转将测量数据转换至水平基准系。设 (x_1, y_1) 为旋转后的观测坐标，(x, y) 为水平状态下的标准坐标，转换关系为：

$$\begin{cases} x = x_1 \cos\bar{\theta} - y_1 \sin\bar{\theta} \\ y = y_1 \cos\bar{\theta} + x_1 \sin\bar{\theta} \end{cases}$$

3.3.4 水平状态参数求解

将校正后的坐标数据代入问题一的处理方法，即可获得工件在水平状态下的精确几何参数。该转换过程消除了初始偏转带来的系统误差，保证了后续分析的准确性。

问题二结果

全局参数

$$\text{偏转角度} = -7.4809^{(1)}$$

直线段拟合结果

区间	拟合方程	R^2	p 值	角度	MAE	RMSE
$[, a_1]$	$z = 0.001524x + 7.668300$	0.4389	< 0.000001	179.91°	0.001072	0.001558
$[a_2, a_3]$	$z = 0.000595x + 7.720503$	0.3813	< 0.000001	179.97°	0.000778	0.001000
$[a_4, a_5]$	$z = 0.000059x + 7.754136$	0.0028	0.00000037	179.99°	0.001044	0.001460
$[a_6, a_7]$	$z = 0.000558x + 7.720063$	0.3167	< 0.000001	179.97°	0.000860	0.001090
$[a_7, a_8]$	$z = 0.200737x - 7.044889$	0.99998	< 0.000001	168.65°	0.001061	0.001296
$[a_8, a_9]$	$z = -0.199602x + 24.482357$	0.99998	< 0.000001	168.71°	0.000804	0.001097
$[a_9, a_{10}]$	$z = -0.000043x + 7.768713$	0.0010	0.027890	179.99°	0.000813	0.001005
$[a_{11}, a_{12}]$	$z = -0.003363x + 8.068299$	0.2613	< 0.000001	179.81°	0.000676	0.000825
$[a_{13}, a_{14}]$	$z = 0.000202x + 7.752312$	0.0398	< 0.000001	179.99°	0.000951	0.001284
$[a_{15}, a_{16}]$	$z = 0.000772x + 7.697689$	0.1738	< 0.000001	179.96°	0.001394	0.002182
$[a_{17},]$	$z = -0.000278x + 7.815536$	0.0413	< 0.000001	179.98°	0.001050	0.001274

圆弧拟合结果

区间	圆心坐标 (h, k)	半径 r	RSS	R^2	RMSE
$[a_{10}, a_{11}]$	$(87.6578, 8.2007)$	1.00231	0.005217	0.999749	0.001361
$[a_{12}, a_{13}]$	$(90.4691, 7.3425)$	0.99982	0.008891	0.999795	0.001670

续表：圆弧拟合结果

区间	圆心坐标 (h,k)	半径 r	RSS	R^2	RMSE
$[a_{14},a_{15}]$	$(100.0005, 9.4796)$	4.00894	0.076664	0.999981	0.002373
$[a_{16},a_{17}]$	$(112.2545, 6.0926)$	3.99841	0.033301	0.999993	0.001546
$[a_1,a_2]_{\text{arc}}$	$(53.1671, 5.2548)$	0.49214	0.002494	0.998209	0.001413
$[a_3,a_4]_{\text{arc}}$	$(60.6584, 5.1092)$	0.32889	0.002253	0.973322	0.001954
$[a_5,a_6]_{\text{arc}}$	$(67.7194, 5.0613)$	0.29333	0.001828	0.991450	0.001616

混合区间拟合结果

区间	左侧直线	右侧直线	圆弧
	$z=-2.7269x+148.7728$	$z=2.7300x-141.3504$	$O=(53.1671,5.2548)$
$[a_1,a_2]$	$\theta=110.14^\circ$	$\theta=110.12^\circ$	$r=0.49214$
	$R^2=0.99991$	$R^2=0.99958$	$R^2=0.99821$
	$z=-3.5927x+221.8481$	$z=3.9843x-237.9242$	$O=(60.6584,5.1092)$
$[a_3,a_4]$	$\theta=105.55^\circ$	$\theta=104.09^\circ$	$r=0.32889$
	$R^2=0.99669$	$R^2=0.99204$	$R^2=0.97332$
	$z=-3.7400x+257.1253$	$z=3.6072x-240.3185$	$O=(67.7194,5.0613)$
$[a_5,a_6]$	$\theta=104.97^\circ$	$\theta=105.49^\circ$	$r=0.29333$
	$R^2=0.99608$	$R^2=0.98866$	$R^2=0.99145$

完整转折点坐标

表 22: 全部转折点坐标

转折点	坐标 (a_i,b_i)
a_1	$(51.71612752, 7.74713400)$
a_2	$(54.61731276, 7.75299316)$
a_3	$(59.59042966, 7.75595150)$
a_4	$(61.66283071, 7.75777330)$
a_5	$(66.67629630, 7.75806905)$
a_6	$(68.77197985, 7.75842272)$
a_7	$(73.75861804, 7.76120415)$
a_8	$(78.75129234, 8.76341921)$

续表：全部转折点坐标

转折点	坐标 (a_i, b_i)
a_9	$(83.75279007, 7.76510882)$
a_{10}	$(86.75512121, 7.76497963)$
a_{11}	$(88.56307010, 7.77048567)$
a_{12}	$(89.56389253, 7.76712019)$
a_{13}	$(91.37253129, 7.77076049)$
a_{14}	$(96.37349704, 7.77177023)$
a_{15}	$(103.63023566, 7.77772688)$
a_{16}	$(108.63030918, 7.78158866)$
a_{17}	$(115.87780736, 7.78333038)$
a_{18}	$(52.69376049, 5.08120920)$
a_{19}	$(53.63891076, 5.08198905)$
a_{20}	$(60.35094653, 5.02362222)$
a_{21}	$(60.97794371, 5.02899191)$
a_{22}	$(67.41885266, 4.98093164)$
a_{23}	$(68.00251866, 4.98279241)$

完整几何参数

表 23: 全部关键几何参数

参数	值
$x_1 = a_2 - a_1$	2.901 185 24
$x_2 = a_3 - a_2$	4.973 116 90
$x_3 = a_4 - a_3$	2.072 401 05
$x_4 = a_5 - a_4$	5.013 465 59
$x_5 = a_6 - a_5$	2.095 683 55
$x_6 = a_7 - a_6$	4.986 638 19
$x_7 = a_8 - a_7$	9.994 172 03
$x_8 = a_{10} - a_9$	3.002 331 14
$x_9 = a_{12} - a_{11}$	1.000 822 43
$x_{10} = a_{14} - a_{13}$	5.000 965 75
$x_{11} = a_{15} - a_{14}$	7.256 738 62
$x_{12} = a_{16} - a_{15}$	5.000 073 52
$x_{13} = a_{17} - a_{16}$	7.247 498 18

续表：全部关键几何参数

参数	值
$c_1 = O_2^x - O_1^x$	7.491 253 45
$c_2 = O_3^x - O_2^x$	7.060 985 49
$c_3 = O_4^x - O_3^x$	19.938 402 63
$c_4 = O_5^x - O_4^x$	2.811 293 06
$c_5 = O_6^x - O_5^x$	9.531 396 77
$c_6 = O_7^x - O_6^x$	12.253 997 44
z_1	0.998 310 39

完整圆心参数

表 24: 全部圆心参数

圆心	坐标 (h, k)	半径 r
O_1	(53.16713554, 5.25480264)	0.492 139 69
O_2	(60.65838899, 5.10919580)	0.328 891 47
O_3	(67.71937448, 5.06128574)	0.293 333 78
O_4	(87.65777711, 8.20068677)	1.002 311 52
O_5	(90.46907017, 7.34252163)	0.999 815 23
O_6	(100.00046694, 9.47959528)	4.008 937 14
O_7	(112.25446438, 6.09258206)	3.998 405 21

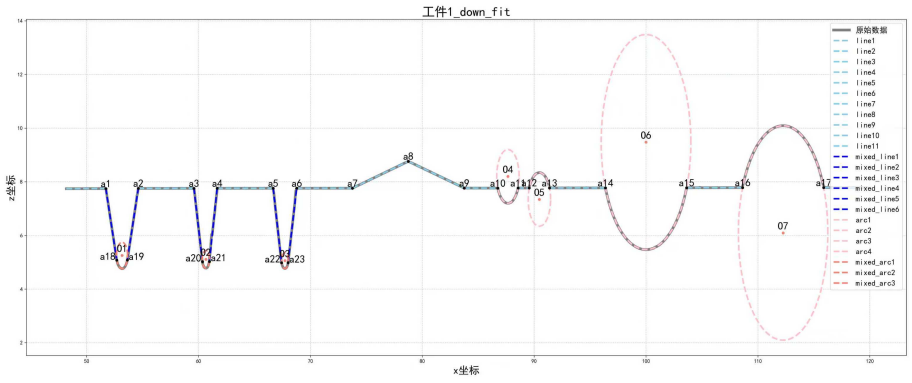


图 10: 拟合总图

3.4 问题三求解

3.4.1 基准线选择与测量原理

我们将十次测量的数据做成图如下：

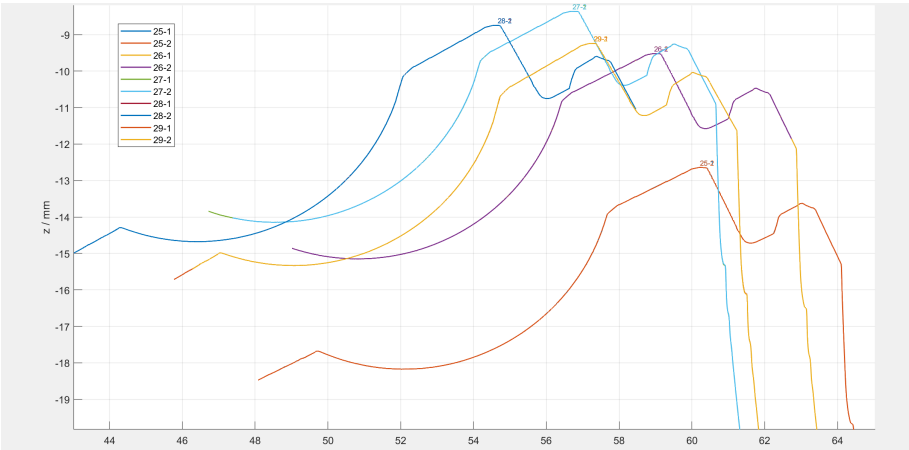


图 11

在工件 2 的轮廓测量中，我们选择第三段直线作为工艺基准面进行倾斜角度计算。该基准面的选取基于以下考虑：

- 几何特性：该段为工件轮廓中最长的直线段。
- 稳定性：直线特征明显，受测量噪声影响较小。
- 代表性：能够反映工件的整体倾斜状态。

3.4.2 单次测量数学模型

对于第 i 次测量获得的轮廓数据，定义第三段直线的点集为：

$$\mathcal{P}_i = \{(x_{ij}, z_{ij})\}_{j=1}^{N_i}$$

其中 N_i 表示第 i 次测量的数据点数量。

采用最小二乘法拟合直线方程：

$$z = k_i x + b_i$$

拟合参数计算如下：

$$k_i = \frac{N_i \sum_{j=1}^{N_i} x_{ij} z_{ij} - \left(\sum_{j=1}^{N_i} x_{ij}\right)\left(\sum_{j=1}^{N_i} z_{ij}\right)}{N_i \sum_{j=1}^{N_i} x_{ij}^2 - \left(\sum_{j=1}^{N_i} x_{ij}\right)^2}, \quad b_i = \frac{\sum_{j=1}^{N_i} z_{ij} - k_i \sum_{j=1}^{N_i} x_{ij}}{N_i}$$

据此得到第 i 次测量的倾斜角度：

$$\alpha_i = \arctan(k_i) \times \frac{180}{\pi}$$

3.4.3 基准角度建立与偏差分析

为消除测量系统误差，建立标准参考角度：

$$\alpha_{\text{ref}} = \frac{1}{10}\sum_{i=1}^{10}\alpha_i$$

定义第 i 次测量的相对角度偏差：

$$\Delta\alpha_i = \alpha_{\text{ref}} - \alpha_i$$

基准选择合理性 采用多次测量平均值作为基准角度，可有效消除以下影响：

- 测量平台的水平度误差
- 单次测量的随机误差
- 工件放置的微小偏差

3.5 问题四

3.5.1 圆的修正

附件 3 提供了工件 2 关于圆 9 次局部测量数据，我们将这些数据画成图如下，再线性拟合，与前面的做法类似。

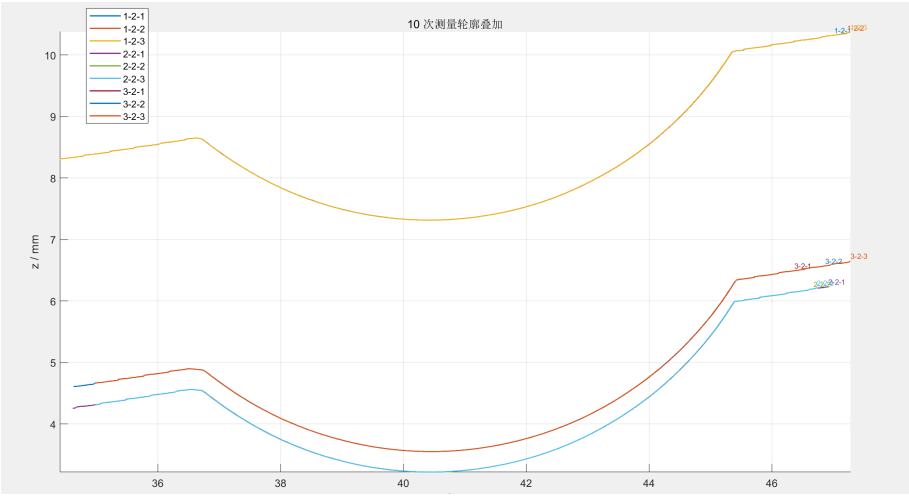


图 12

我们可以找到最低点，从而把最低点记成特征点。由于只需要对工件 2 的轮廓校正，即我们需要计算工件 2 圆角部分的参数即可：即圆半径和弧长。将附件三中的数据经过上下左右平移，使特征点重合。

得到结果。

3.5.2 角的修正

附件 4 提供了工件 2 关于角的 9 次局部测量数据，我们将这些数据画成图如下。

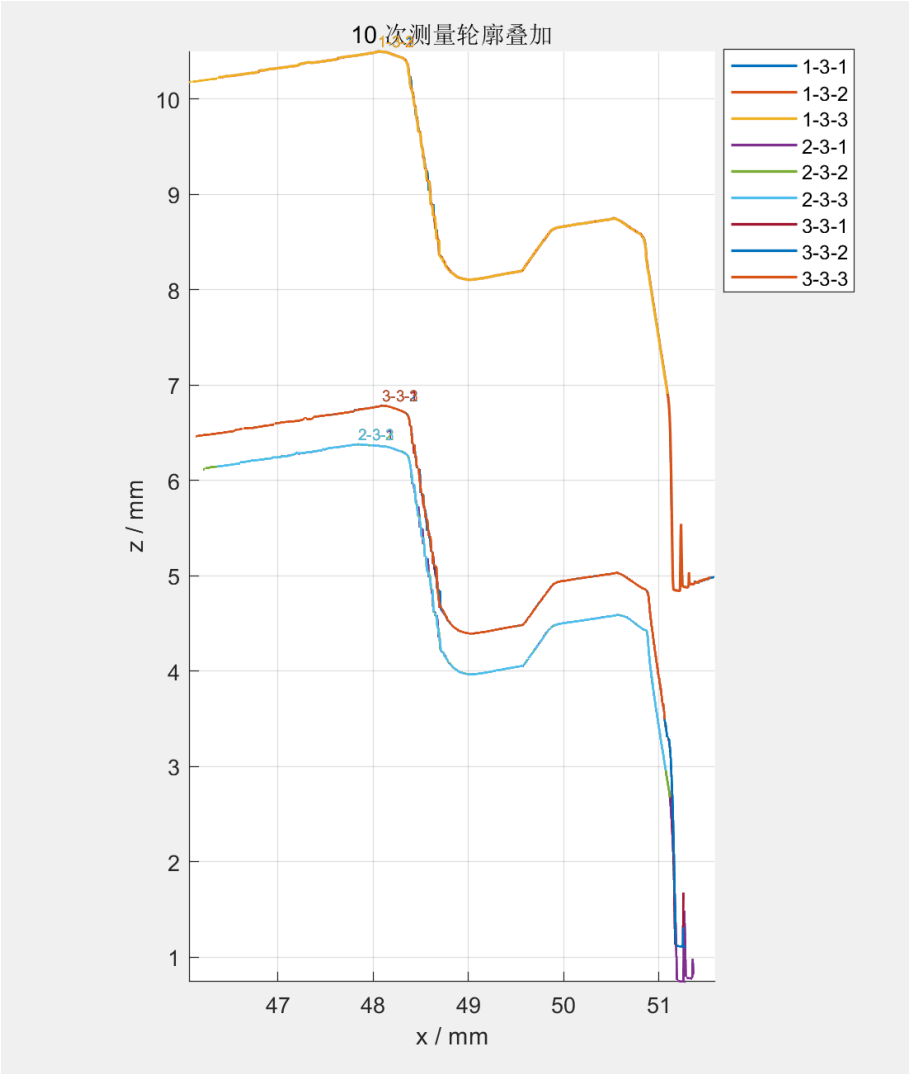


图 13

令最高点为特征点，通过平移令特征点重合，再线性拟合，与前面的做法类似。得到结果。

4 总结分析及应用

本文建立的轮廓线参数计算模型通过多阶段处理方法实现了精确的几何特征提取，其主要特点包括：

- **分阶段建模框架**：
  - 采用曲率分析法自动定位断点，避免人工选取的主观性

  - 通过预处理去除孤立离群点，提高数据质量

- **鲁棒拟合算法**：
  - 直线段采用 RANSAC 方法进行拟合
  - 强制直线与圆弧在过渡点保持连续

- **实际应用效果**：
  - 为轮廓配准和误差分析建立统一坐标系
  - 通过相对角度转换消除系统误差

5 模型评价与推广

5.1 模型评价

我们的方法在数据拟合、误差校正、标准化处理和数据融合等方面均取得了良好的效果，具有较强的鲁棒性和实用性。

我们在分段最小二乘拟合与解析几何结合的方法下，成功建立了工件 1 轮廓线的精确数学模型，准确识别了转折点坐标及相关参数，确保了轮廓线拟合的精度。

我们在倾斜数据校正的基准线处理方法下，通过水平校正和整体最小二乘拟合，显著提高了工件在倾斜状态下的参数估计精度，有效降低了测量误差。

我们在以最长直线段为基准的标准化处理策略下，通过计算相对倾斜角并求取平均值，消除了系统误差，建立了稳定的参考角度体系，使参数估计更加可靠。

我们在基于特征点匹配的精度提升算法下，通过多组数据的空间对齐与融合拟合，进一步提高了局部测量数据的准确性，为类似工件的几何参数测量提供了可推广的技术方案。

5.2 推广潜力

本方法体系具有以下可扩展性：

- **数据维度扩展**：
  - 给定更高采样频率 f_s，可改进曲率计算为：

$$k(t) = \frac{\|\dot{r}(t) \times \ddot{r}(t)\|}{\|\dot{r}(t)\|^3} \tag{2}$$

  - 增加正交视角数据后，可构建三维曲率场：

$$\kappa(x, y, z) = \nabla \times (\nabla \times \mathbf{r}) \tag{3}$$

- **实时处理**：若配备 $\Delta t \leq 10ms$ 的实时数据流，可建立增量式更新：

$$\hat{\theta}_t = \alpha\theta_{t-1} + (1-\alpha)\theta_{new} \tag{4}$$

- **约束增强**：引入 C^2 连续性约束后，可处理更复杂的过渡特征：

$$\min \sum \|r_i - f(s_i)\|^2 + \lambda\|\nabla^2 f\|^2 \tag{5}$$

参考文献

[1] 戴静兰, 陈志杨, 叶修梓. ICP 算法在点云配准中的应用 [J]. 中国图象图形学报, 2007, 12(3): 517–521.

[2] 仇谷烽, 郭培基, 谢滨, 等. 接触式非球面轮廓测量的数据处理模型 [J]. 光学精密工程, 2007, 15(4): 492–498.

[3] 韩志国, 李锁印, 冯亚南, 等. 接触式轮廓仪探针状态检查图形样块的研制 [J]. 微纳电子技术, 2019, 56(9): 761–765.

[4] 颜鲁林. 接触式轮廓仪自动标注方法研究 [J]. 甘肃高师学报, 2021, 26(2): 36–40.

[5] Supaporn Jaturunruangsri. Evaluation of Material Surface Profiling Methods: Contact versus Non-contact[D]. Brunel University London, 2014.

[6] 李美霞. 接触式轮廓仪的自动标注问题分析 [J]. 延安职业技术学院学报, 2023, 37(1): 103–108.

[7] 王阿利. 工件轮廓线参数校正方法研究 [J]. 内燃机与配件, 2021(10): 58–59.

[8] 付嵌波. 触针式轮廓仪几何误差补偿算法研究与实验 [D]. 西安: 西安工业大学, 2023.

[9] 薛毅. ” 接触式轮廓仪的自动标注” 的问题解析 [J]. 数学建模及其应用, 2021, 10(1): 92–100.

[10] 高彤, 陈鸿, 张亮, 等. 接触式轮廓扫描仪系统标定算法 [J]. 电子技术应用, 2021, 47(3): 65–70+74.

[11] 蒋培军. 接触式轮廓仪测量的自动标注问题探究——基于最小二乘法 [J]. 南通职业大学学报, 2023, 37(3): 66–70.

[12] 张穗, 刘楠. 接触式轮廓仪的测量精度问题 [J]. 理论算法, 2022(8): 55–57.

[13] 喻雪. 接触式轮廓仪的工件形状自动标注数学模型研究 [J]. 兰州石化职业技术学院学报, 2021, 21(2): 15–17.

[14] 乔旭安, 陈丽莎, 李华, 等. 接触式轮廓仪的自动标注参数及分析 [J]. 科技与创新, 2021(7): 97–100 .

[15] 王云庆, 李庆祥, 周兆英. 接触式轮廓测量中触针测量力的分析 [J]. 现代计量测试, 1996(01): 18–21+17.

[16] 伍春兰. 直线方程、倾斜角与斜率的教学实践与反思 [J]. 数学通报, 2017, 56(05): 30–33+63.

附录

第一问代码

```
1
2  #This is a file for Q1.
3  import numpy as np
4  from numpy import argmin
5  import pandas as pd
6  import matplotlib.pyplot as plt
7  import math
8  from scipy import stats
9  from scipy.optimize import least_squares
10
11 import os
12
13 #设置字体
14 plt.rcParams['font.sans-serif'] = ['SimHei']
15 #设置负号
16 plt.rcParams['axes.unicode_minus'] = False
17
18 os.makedirs('Figures', exist_ok=True)
19
20 #区间数据读取器
21 def data_reader(start,end,df):
22     # 读取数据
23     x = df['x']
24     #选取区间内部的点
25     start_idx = np.searchsorted(x, start+0.2, side='left')
26     end_idx = np.searchsorted(x, end-0.2, side='right')
27     x = df['x'].iloc[start_idx:end_idx + 1].values
28     z = df['z'].iloc[start_idx:end_idx + 1].values
29     return x,z
30
31 #角度计算器（大角）
32 def angle_cal(slope):
33     if slope >= 0:
34         angle_rad = np.pi-math.atan(slope)  # 弧度
35         angle_deg = math.degrees(angle_rad)  # 角度
36     else:
```

```
37          angle_rad = np.pi+math.atan(slope)
38          angle_deg = math.degrees(angle_rad)
39      return angle_rad,angle_deg
40
41  #交点计算器
42  def line_intersection(line1, line2, tol=1e-6):
43      """
44      计算直线间的交点
45      :param: 直线参数 (Ai, Bi, Ci)=(slope,-1,intercept) 表示 Ai*x +
            Bi*z + Ci = 0
46      """
47      A1, B1, C1 = line1
48      A2, B2, C2 = line2
49      # 计算行列式
50      det = A1 * B2 - A2 * B1
51      # 处理特殊情况
52      if abs(det) < tol:
53          # 检查是否重合
54          if abs(A1 * C2 - A2 * C1) < tol and abs(B1 * C2 - B2 * C1) <
                tol:
55              return "直线重合"
56          else:
57              return "直线平行"
58      # 计算交点坐标
59      x = (B1 * C2 - B2 * C1) / det
60      z = (A2 * C1 - A1 * C2) / det
61      return (x, z)
62
63  def line_circle_intersection(line, circle, tol=0.7):
64      """
65      计算直线与圆的交点
66      :param line: 直线参数 (A, B, C)=(slope,-1,intercept) 表示 Ax +
            Bz + C = 0
67      :param circle: 圆参数 (h, k, r) 圆心(h,k)和半径r
68      """
69      A, B, C = line
70      h, k, r = circle
71      # 特殊情况处理: 垂直线 (B=0)
72      if abs(B) < tol:
73          # 垂直线方程: Ax + C = 0 → x = -C/A
```

```
        x0 = -C / A
        # 代入圆方程：(x0-h)^2 + (z-k)^2 = r^2
        d_sq = r ** 2 - (x0 - h) ** 2
        if d_sq < -tol:
            print("无交点")
            return ()  # 无交点
        elif abs(d_sq) < tol:
            print("相切")
            return (x0, k)  # 相切
        else:
            y1 = k + np.sqrt(d_sq)
            y2 = k - np.sqrt(d_sq)
            print("两个交点")
            return (x0, y1), (x0, y2)
    # 将直线表示为 z = mx + c
    m = -A / B
    c = -C / B
    # 展开为二次方程：ax^2 + bx + c = 0
    a = 1 + m ** 2
    b = 2 * (m * (c - k) - h)
    c_eq = h ** 2 + (c - k) ** 2 - r ** 2
    # 计算判别式
    delta = b ** 2 - 4 * a * c_eq
    print('delta=',delta)
    if delta < -tol:
        print("无交点")
        return ()  # 无交点
    elif abs(delta) < tol:
        print("相切")
        x = -b / (2 * a)
        y = m * x + c
        return (x, y)
    else:
        # 两个交点
        print("两个交点")
        sqrt_disc = np.sqrt(delta)
        x2 = (-b + sqrt_disc) / (2 * a)
        x1 = (-b - sqrt_disc) / (2 * a)
        return (x1, m * x1 + c), (x2, m * x2 + c)

```

```
#直线拟合器
def linear_fit(start,end,df):
    # 读取数据
    x,z=data_reader(start,end,df)

    # 计算线性拟合参数
    slope, intercept, r_value, p_value, std_err =
       stats.linregress(x, z)
    angle_rad,angle_deg=angle_cal(slope)

    # 计算评价指标
    y_pred = slope * x + intercept
    residuals = z - y_pred
    mae = np.mean(np.abs(residuals))  # 平均绝对误差
    rmse = np.sqrt(np.mean(residuals ** 2))  # 均方根误差

    print(f"拟合直线方程: z = {slope:.8f}x + {intercept:.8f}\n"
          f"R^2: {r_value ** 2:.8f}\n"
          f"p值: {p_value:.8f}\n"
          f"斜率标准误差: {std_err:.8f}\n"
          f"MAE: {mae:.8f}\n"
          f"RMSE: {rmse:.8f}\n"
          f"直线角度: {angle_deg:.4f}° (弧度: {angle_rad:.4f})\n"
          )
    return (slope,-1,intercept)

#弧线拟合器
def arc_fit(start,end,df):
    # 读取数据
    x,z=data_reader(start,end,df)
    # 初始参数估计
    # 使用数据的质心作为圆心初始估计
    h0 = np.mean(x)
    k0 = np.mean(z)
    # 计算到质心的平均距离作为半径初始估计
    r0 = np.sqrt(np.mean((x - h0) ** 2 + (z - k0) ** 2))

    #定义残差
    def residuals(params):
        h, k, r = params
```

```
        return np.sqrt((x - h) ** 2 + (z - k) ** 2) - r

    # 使用LM算法最小化残差平方和
    result = least_squares(
        fun=residuals,  # 残差函数
        x0=[h0, k0, r0],  # 初始参数
        method='lm',  # Levenberg-Marquardt方法
        ftol=1e-8,  # 函数容忍度
        xtol=1e-8,  # 参数容忍度
        max_nfev=100  # 最大函数评估次数
    )
    h_fit, k_fit, r_fit = result.x
    # 确保半径为正值
    r_fit = abs(r_fit)

    # 计算评价指标
    res = residuals((h_fit, k_fit, r_fit))
    rss = np.sum(res ** 2)  # 残差平方和
    sst = np.sum((z - np.mean(z)) ** 2)  # 总平方和
    r_squared = 1 - (rss / sst)  # 决定系数
    mae = np.mean(np.abs(res))  # 平均绝对误差
    rmse = np.sqrt(np.mean(res ** 2))  # 均方根误差

    print(f"拟合参数(h, k, r)={(h_fit, k_fit, r_fit)}\n"
          f"残差平方和(RSS): {rss:.8f}\n"
          f"R^2: {r_squared:.8f}\n"
          f"MAE: {mae:.8f}\n"
          f"RMSE: {rmse:.8f}\n")
    return (h_fit, k_fit, r_fit)

#混合拟合器
def mixed_fit(start,end,target_z1,target_z2,target_z3,df):
    #获取区间
    x,z=data_reader(start,(start+end)/2,df)
    # 获取原始数据的全局索引
    start_idx = np.searchsorted(df['x'], start+0.2, side='left')
    idx1 = start_idx + np.argmin(np.abs(z - target_z1))
    idx2 = start_idx + np.argmin(np.abs(z - target_z2))
    idx3 = start_idx + np.argmin(np.abs(z - target_z3))

```

```
    x, z = data_reader((start + end) / 2,end,df)
    mid_idx = np.searchsorted(df['x'], (start+end)/2+0.2,
        side='left')
    idx6 = mid_idx + np.argmin(np.abs(z - target_z1))
    idx5 = mid_idx + np.argmin(np.abs(z - target_z2))
    idx4 = mid_idx + np.argmin(np.abs(z - target_z3))

    #左侧直线拟合
    print(f"左侧直线区间的拟合结果: ")
    mixed_line1=linear_fit(df['x'][idx1],df['x'][idx2],df)

    #右侧直线拟合
    print(f"右侧直线区间的拟合结果: ")
    mixed_line2=linear_fit(df['x'][idx5],df['x'][idx6],df)

    #中间弧线拟合
    print(f"中间弧线区间的拟合结果: ")
    mixed_arc=arc_fit(df['x'][idx3],df['x'][idx4],df)

    return mixed_line1,mixed_line2,mixed_arc

def plot_all_results(df):
    plt.figure(figsize=(24, 10))

    # 1. 绘制原始数据
    plt.plot(df['x'], df['z'], color='gray', linewidth=5,
        label='原始数据')

    # 2. 绘制所有直线段拟合曲线
    def plot_fitted_line(line,start,end,line_name,color='skyblue'):
        slope, _, intercept = line
        x_fit = np.linspace(start, end, 100)
        y_fit = slope * x_fit + intercept
        plt.plot(x_fit, y_fit, color=color, linestyle='--',
            linewidth=3, label=line_name)

    plot_fitted_line(line1,df['x'][0],turning_dict['a1'],'line1')
    plot_fitted_line(line2,turning_dict['a2'],turning_dict['a3'],'line2')
    plot_fitted_line(line3,turning_dict['a4'],turning_dict['a5'],'line3')
    plot_fitted_line(line4,turning_dict['a6'],turning_dict['a7'],'line4')
```

```
    plot_fitted_line(line5,turning_dict['a7'],turning_dict['a8'],'line5')
    plot_fitted_line(line6,turning_dict['a8'],turning_dict['a9'],'line6')
    plot_fitted_line(line7,turning_dict['a9'],turning_dict['a10'],'line7')
    plot_fitted_line(line8,turning_dict['a11'],turning_dict['a12'],'line8')
    plot_fitted_line(line9,turning_dict['a13'],turning_dict['a14'],'line9')
    plot_fitted_line(line10,turning_dict['a15'],turning_dict['a16'],'line10')
    plot_fitted_line(line11,turning_dict['a17'],df['x'].iloc[-1],'line11')

    plot_fitted_line(mixed_line1,turning_dict['a1'],turning_dict['a18'],'mixed_l
    plot_fitted_line(mixed_line2,turning_dict['a19'],turning_dict['a2'],'mixed_l
    plot_fitted_line(mixed_line3,turning_dict['a3'],turning_dict['a20'],'mixed_l
    plot_fitted_line(mixed_line4,turning_dict['a21'],turning_dict['a4'],'mixed_l
    plot_fitted_line(mixed_line5,turning_dict['a5'],turning_dict['a22'],'mixed_l
    plot_fitted_line(mixed_line6,turning_dict['a23'],turning_dict['a6'],'mixed_l

    # 3. 绘制所有完整圆
    def plot_full_circle(arc_params, color, label, name):
        h, k, r = arc_params
        theta = np.linspace(0, 2 * np.pi, 100)
        x = h + r * np.cos(theta)
        y = k + r * np.sin(theta)
        plt.plot(x, y, color=color, linestyle='--', linewidth=3,
           label=label)
        plt.scatter(h, k, color='salmon', s=20)
        plt.text(h, k+0.12, name, fontsize=20, ha='center')

    # 绘制所有弧线
    plot_full_circle(arc1, 'pink', 'arc1','O4')
    plot_full_circle(arc2, 'pink', 'arc2','O5')
    plot_full_circle(arc3, 'pink', 'arc3','O6')
    plot_full_circle(arc4, 'pink', 'arc4','O7')

    plot_full_circle(mixed_arc1, 'salmon', 'mixed_arc1','O1')
    plot_full_circle(mixed_arc2, 'salmon', 'mixed_arc2','O2')
    plot_full_circle(mixed_arc3, 'salmon', 'mixed_arc3','O3')

     # 4. 绘制所有转折点
    for i in range(1, 24):
        if f'a{i}' in turning_dict:
            plt.scatter(turning_dict[f'a{i}'],
```

```
                turning_dict[f'b{i}'], color='black', s=20, zorder=10)
269             if i in [18,20,22]:  # 左侧点向左偏移
270                 plt.text(turning_dict[f'a{i}']-0.1,
                        turning_dict[f'b{i}'],
271                             f'a{i}', fontsize=20, ha='right')
272             elif i in [19,21,23]:  # 右侧点向右偏移
273                 plt.text(turning_dict[f'a{i}']+0.1,
                        turning_dict[f'b{i}'],
274                             f'a{i}', fontsize=20, ha='left')
275             else:  # 其他点保持原样
276                 plt.text(turning_dict[f'a{i}'],
                        turning_dict[f'b{i}'] + 0.05,
277                             f'a{i}', fontsize=20, ha='center')
278
279
280     # 5. 添加图例和标签
281     plt.legend(fontsize=15, loc='upper right')
282     plt.title('工件1_level_fit', fontsize=24)
283     plt.xlabel('x坐标', fontsize=20)
284     plt.ylabel('z坐标', fontsize=20)
285     plt.grid(True, linestyle='--', alpha=0.7)
286     plt.tight_layout()
287
288     # 6. 保存和显示图像
289     plt.savefig('Figures/工件1_level_fit.png', dpi=300,
            bbox_inches='tight')
290     plt.show()
291
292
293 if __name__ == '__main__':
294     #读取数据
295     df=pd.read_excel('第2周C题: 接触式轮廓仪的自动标注(2020年D题)/附件1_工件1的测
296     #标记转折点
297     turning_points=['(49.7154,-1.7739)','(52.7223,-1.7699)','(57.5009,-1.7676)',
298                     '(64.5999,-1.7690)','(66.8444,-1.7679)','(71.6264,-1.7676)',
299                     '(81.8819,-1.7691)','(84.6165,-1.7795)','(86.6494,-1.7737)',
300                     '(89.5424,-1.7685)','(94.3925,-1.7840)','(101.7909,-1.7828)'
301                     '(106.5504,-1.7766)','(114.1814,-1.7795)'
302                     ]
303     turning_points_x=[]
```

```
    turning_points_z=[]
    for item in turning_points:
        coord=item.strip('()').split(',')
        turning_points_x.append(float(coord[0]))
        turning_points_z.append(float(coord[1]))
    #转折点命名
    turning_dict = {}
    for i in range(len(turning_points)):
        turning_dict[f'a{i+1}'] = turning_points_x[i]
        turning_dict[f'b{i+1}'] = turning_points_z[i]

    #绘图
    plt.figure(figsize=(24,10))
    plt.plot(df['x'],df['z'],color='gray',linewidth=5)
    #添加转折点
    plt.scatter(turning_points_x, turning_points_z, color='gray')
    for i in range(len(turning_points)):
        plt.text(turning_points_x[i],turning_points_z[i]+0.08,f'a{i+1}',fontsize
    plt.title('工件1的测量数据')
    plt.xlabel('x')
    plt.ylabel('z')
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.savefig('Figures/工件1_level.png')
    plt.show()
    plt.close()

    #直线区段拟合
    print("区间[,a1]的拟合结果")
    line1=linear_fit(df['x'][0],turning_dict['a1'],df)
    print('区间[a2,a3]的拟合结果: ')
    line2=linear_fit(turning_dict['a2'],turning_dict['a3'],df)
    print('区间[a4,a5]的拟合结果: ')
    line3=linear_fit(turning_dict['a4'],turning_dict['a5'],df)
    print('区间[a6,a7]的拟合结果: ')
    line4=linear_fit(turning_dict['a6'],turning_dict['a7'],df)
    print('区间[a7,a8]的拟合结果: ')
    line5=linear_fit(turning_dict['a7'],turning_dict['a8'],df)
    print('区间[a8,a9]的拟合结果: ')
    line6=linear_fit(turning_dict['a8'],turning_dict['a9'],df)
```

```
    print('区间[a9,a10]的拟合结果: ')
    line7=linear_fit(turning_dict['a9'],turning_dict['a10'],df)
    print('区间[a11,a12]的拟合结果: ')
    line8=linear_fit(turning_dict['a11'],turning_dict['a12'],df)
    print('区间[a13,a14]的拟合结果: ')
    line9=linear_fit(turning_dict['a13'],turning_dict['a14'],df)
    print('区间[a15,a16]的拟合结果: ')
    line10=linear_fit(turning_dict['a15'],turning_dict['a16'],df)
    print("区间[a17,]的拟合结果: ")
    line11=linear_fit(turning_dict['a17'],df['x'].iloc[-1],df)

    #更新转折点坐标
    point=line_intersection(line4, line5)
    turning_dict['a7']=point[0]
    turning_dict['b7']=point[1]
    point=line_intersection(line5, line6)
    turning_dict['a8']=point[0]
    turning_dict['b8']=point[1]
    point = line_intersection(line6, line7)
    turning_dict['a9'] = point[0]
    turning_dict['b9'] = point[1]
    #计算z1
    z1=turning_dict['b8']-turning_dict['b9']
    print(f'z1={z1:.8f}\n')

    #弧线区段拟合
    print('区间[a10,a11]的拟合结果: ')
    arc1=arc_fit(turning_dict['a10'],turning_dict['a11'],df)
    print('区间[a12,a13]的拟合结果: ')
    arc2=arc_fit(turning_dict['a12'],turning_dict['a13'],df)
    print('区间[a14,a15]的拟合结果: ')
    arc3=arc_fit(turning_dict['a14'],turning_dict['a15'],df)
    print('区间[a16,a17]的拟合结果: ')
    arc4=arc_fit(turning_dict['a16'],turning_dict['a17'],df)

    #混合区段拟合
    print('区间[a1,a2]的拟合结果: ')
    mixed_line1, mixed_line2,
       mixed_arc1=mixed_fit(turning_dict['a1'],turning_dict['a2'],-1.9,-3.8,-4.3
    print('区间[a3,a4]的拟合结果: ')
```

```
    mixed_line3, mixed_line4,
        mixed_arc2=mixed_fit(turning_dict['a3'],turning_dict['a4'],-2.1,-4.2,-4.3
    print('区间[a5,a6]的拟合结果: ')
    mixed_line5, mixed_line6,
        mixed_arc3=mixed_fit(turning_dict['a5'],turning_dict['a6'],-2.1,-4.0,-4.2

    # 更新转折点坐标
    point=line_intersection(line1,mixed_line1 )
    turning_dict['a1']=point[0]
    turning_dict['b1']=point[1]
    point = line_intersection(line2, mixed_line2)
    turning_dict['a2'] = point[0]
    turning_dict['b2'] = point[1]
    point=line_intersection(line2,mixed_line3)
    turning_dict['a3']=point[0]
    turning_dict['b3']=point[1]
    point=line_intersection(mixed_line4,line3)
    turning_dict['a4']=point[0]
    turning_dict['b4']=point[1]
    point = line_intersection(mixed_line5, line3)
    turning_dict['a5'] = point[0]
    turning_dict['b5'] = point[1]
    point = line_intersection(mixed_line6, line4)
    turning_dict['a6'] = point[0]
    turning_dict['b6'] = point[1]
    point = line_circle_intersection(line7, arc1)[0]
    turning_dict['a10'] = point[0]
    turning_dict['b10'] = point[1]
    point = line_circle_intersection(line8, arc1)[1]
    turning_dict['a11'] = point[0]
    turning_dict['b11'] = point[1]
    point = line_circle_intersection(line8, arc2)[0]
    turning_dict['a12'] = point[0]
    turning_dict['b12'] = point[1]
    point = line_circle_intersection(line9, arc2)[1]
    turning_dict['a13'] = point[0]
    turning_dict['b13'] = point[1]
    point = line_circle_intersection(line9, arc3)[0]
    turning_dict['a14'] = point[0]
    turning_dict['b14'] = point[1]
```

```
    point = line_circle_intersection(line10, arc3)[1]
    turning_dict['a15'] = point[0]
    turning_dict['b15'] = point[1]
    point = line_circle_intersection(line10, arc4)[0]
    turning_dict['a16'] = point[0]
    turning_dict['b16'] = point[1]
    point = line_circle_intersection(line11, arc4)[1]
    turning_dict['a17'] = point[0]
    turning_dict['b17'] = point[1]
    point=line_circle_intersection(mixed_line1,mixed_arc1)
    turning_dict['a18']=point[0]
    turning_dict['b18']=point[1]
    point = line_circle_intersection(mixed_line2, mixed_arc1)
    turning_dict['a19'] = point[0]
    turning_dict['b19'] = point[1]
    point = line_circle_intersection(mixed_line3, mixed_arc2)
    turning_dict['a20'] = point[0]
    turning_dict['b20'] = point[1]
    point = line_circle_intersection(mixed_line4, mixed_arc2)[0]
    turning_dict['a21'] = point[0]
    turning_dict['b21'] = point[1]
    point = line_circle_intersection(mixed_line5, mixed_arc3)
    turning_dict['a22'] = point[0]
    turning_dict['b22'] = point[1]
    point = line_circle_intersection(mixed_line6, mixed_arc3)[0]
    turning_dict['a23'] = point[0]
    turning_dict['b23'] = point[1]

    #打印转折点列表
    print("转折点列表")
    for i in range(23):
        print(f'(a{i+1},b{i+1})=({turning_dict[f'a{i+1}']:.8f},{turning_dict[f'b
    print("圆参数")
    for i, arc in enumerate([mixed_arc1, mixed_arc2, mixed_arc3], 1):
        h, k, r = arc
        print(f'O{i}=({h:.8f},{k:.8f}) 半径={r:.8f}')
    for i, arc in enumerate([arc1, arc2, arc3, arc4], 4):
        h, k, r = arc
        print(f'O{i}=({h:.8f},{k:.8f}) 半径={r:.8f}')

```

```
    #绘制总图
    plot_all_results(df)
```

5.3 第二问代码

```
from Q1_main import line_intersection, line_circle_intersection,
    linear_fit, arc_fit, mixed_fit
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import math

#设置字体
plt.rcParams['font.sans-serif'] = ['SimHei']
#设置负号
plt.rcParams['axes.unicode_minus'] = False

os.makedirs('Figures', exist_ok=True)

def angle_search(*lines):
    slopes = [line[0] for line in lines]
    return np.mean([math.atan(s) for s in slopes])

def coordinate_tranaform(df,angle):
    df['x_new'] = df['x'] * np.cos(angle) - df['z'] * np.sin(angle)
    df['z_new'] = df['x'] * np.sin(angle) + df['z'] * np.cos(angle)
    df['x'] = df['x_new'].round(8)
    df['z'] = df['z_new'].round(8)

    df.drop(columns=['x_new', 'z_new'], inplace=True)
    df.to_excel('Figures/工件1_down_new.xlsx', sheet_name='down',
        index=False)
    return df

def plot_all_results(df):
    plt.figure(figsize=(24, 10))

    # 1. 绘制原始数据
```

```
33      plt.plot(df['x'], df['z'], color='gray', linewidth=5,
            label='原始数据')
34
35      # 2. 绘制所有直线段拟合曲线
36      def plot_fitted_line(line, start, end, line_name,
            color='skyblue'):
37          slope, _, intercept = line
38          x_fit = np.linspace(start, end, 100)
39          y_fit = slope * x_fit + intercept
40          plt.plot(x_fit, y_fit, color=color, linestyle='--',
                linewidth=3, label=line_name)
41
42      plot_fitted_line(line1, df['x'][0], turning_dict['a1'], 'line1')
43      plot_fitted_line(line2, turning_dict['a2'], turning_dict['a3'],
            'line2')
44      plot_fitted_line(line3, turning_dict['a4'], turning_dict['a5'],
            'line3')
45      plot_fitted_line(line4, turning_dict['a6'], turning_dict['a7'],
            'line4')
46      plot_fitted_line(line5, turning_dict['a7'], turning_dict['a8'],
            'line5')
47      plot_fitted_line(line6, turning_dict['a8'], turning_dict['a9'],
            'line6')
48      plot_fitted_line(line7, turning_dict['a9'], turning_dict['a10'],
            'line7')
49      plot_fitted_line(line8, turning_dict['a11'],
            turning_dict['a12'], 'line8')
50      plot_fitted_line(line9, turning_dict['a13'],
            turning_dict['a14'], 'line9')
51      plot_fitted_line(line10, turning_dict['a15'],
            turning_dict['a16'], 'line10')
52      plot_fitted_line(line11, turning_dict['a17'], df['x'].iloc[-1],
            'line11')
53
54      plot_fitted_line(mixed_line1, turning_dict['a1'],
            turning_dict['a18'], 'mixed_line1', color='blue')
55      plot_fitted_line(mixed_line2, turning_dict['a19'],
            turning_dict['a2'], 'mixed_line2', color='blue')
56      plot_fitted_line(mixed_line3, turning_dict['a3'],
            turning_dict['a20'], 'mixed_line3', color='blue')
```

```
    plot_fitted_line(mixed_line4, turning_dict['a21'],
        turning_dict['a4'], 'mixed_line4', color='blue')
    plot_fitted_line(mixed_line5, turning_dict['a5'],
        turning_dict['a22'], 'mixed_line5', color='blue')
    plot_fitted_line(mixed_line6, turning_dict['a23'],
        turning_dict['a6'], 'mixed_line6', color='blue')

    # 3. 绘制所有完整圆
    def plot_full_circle(arc_params, color, label, name):
        h, k, r = arc_params
        theta = np.linspace(0, 2 * np.pi, 100)
        x = h + r * np.cos(theta)
        y = k + r * np.sin(theta)
        plt.plot(x, y, color=color, linestyle='--', linewidth=3,
            label=label)
        plt.scatter(h, k, color='salmon', s=20)
        plt.text(h, k + 0.12, name, fontsize=20, ha='center')

    # 绘制所有弧线
    plot_full_circle(arc1, 'pink', 'arc1', 'O4')
    plot_full_circle(arc2, 'pink', 'arc2', 'O5')
    plot_full_circle(arc3, 'pink', 'arc3', 'O6')
    plot_full_circle(arc4, 'pink', 'arc4', 'O7')

    plot_full_circle(mixed_arc1, 'salmon', 'mixed_arc1', 'O1')
    plot_full_circle(mixed_arc2, 'salmon', 'mixed_arc2', 'O2')
    plot_full_circle(mixed_arc3, 'salmon', 'mixed_arc3', 'O3')

    # 4. 绘制所有转折点
    for i in range(1, 24):
        if f'a{i}' in turning_dict:
            plt.scatter(turning_dict[f'a{i}'],
                turning_dict[f'b{i}'], color='black', s=20, zorder=10)
            if i in [18, 20, 22]:  # 左侧点向左偏移
                plt.text(turning_dict[f'a{i}'] - 0.1,
                    turning_dict[f'b{i}'],
                        f'a{i}', fontsize=20, ha='right')
            elif i in [19, 21, 23]:  # 右侧点向右偏移
                plt.text(turning_dict[f'a{i}'] + 0.1,
                    turning_dict[f'b{i}'],
```

```
                        f'a{i}', fontsize=20, ha='left')
            else:  # 其他点保持原样
                plt.text(turning_dict[f'a{i}'],
                    turning_dict[f'b{i}'] + 0.05,
                        f'a{i}', fontsize=20, ha='center')

    # 5. 添加图例和标签
    plt.legend(fontsize=15, loc='upper right')
    plt.title('工件1_down_fit', fontsize=24)
    plt.xlabel('x坐标', fontsize=20)
    plt.ylabel('z坐标', fontsize=20)
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.tight_layout()

    # 6. 保存和显示图像
    plt.savefig('Figures/工件1_down_fit.png', dpi=300,
        bbox_inches='tight')
    plt.show()


if __name__ == '__main__':
    # 读取数据
    df =
        pd.read_excel('第2周C题: 接触式轮廓仪的自动标注(2020年D题)/附件1_工件1的测
        sheet_name='down')
    # 标记转折点
    turning_points = ['(52.2842,0.9468)', '(55.1646,0.5689)',
        '(60.0920,-0.0733)', '(62.1717,-0.3424)',
                      '(67.1081,-0.9914)', '(69.1762,-1.2660)',
                          '(74.1437,-1.9026)','(79.2127,-1.5718)',
                      '(84.1239,-3.2049)','(87.0161,-3.5982)',
                          '(88.8237,-3.8364)', '(89.7228,-3.9415)',
                      '(91.7218,-4.1919)','(96.5639,-4.8494)',
                          '(103.7796,-5.7907)', '(108.6285,-6.4093)',
                      '(116.0250,-7.3731)'
                      ]
    turning_points_x = []
    turning_points_z = []
    for item in turning_points:
        coord = item.strip('()').split(',')
```

```
        turning_points_x.append(float(coord[0]))
        turning_points_z.append(float(coord[1]))
    # 转折点命名
    turning_dict = {}
    for i in range(len(turning_points)):
        turning_dict[f'a{i + 1}'] = turning_points_x[i]
        turning_dict[f'b{i + 1}'] = turning_points_z[i]

    # 绘图
    plt.figure(figsize=(24, 10))
    plt.plot(df['x'], df['z'], color='gray', linewidth=5)
    # 添加转折点
    plt.scatter(turning_points_x, turning_points_z, color='gray')
    for i in range(len(turning_points)):
        plt.text(turning_points_x[i], turning_points_z[i] + 0.08,
            f'a{i + 1}', fontsize=20)

    # 直线区段拟合
    print("区间[,a1]的拟合结果")
    line1 = linear_fit(df['x'][0], turning_dict['a1'], df)
    print('区间[a2,a3]的拟合结果: ')
    line2 = linear_fit(turning_dict['a2'], turning_dict['a3'], df)
    print('区间[a4,a5]的拟合结果: ')
    line3 = linear_fit(turning_dict['a4'], turning_dict['a5'], df)
    print('区间[a6,a7]的拟合结果: ')
    line4 = linear_fit(turning_dict['a6'], turning_dict['a7'], df)
    print('区间[a9,a10]的拟合结果: ')
    line7 = linear_fit(turning_dict['a9'], turning_dict['a10'], df)
    print('区间[a11,a12]的拟合结果: ')
    line8 = linear_fit(turning_dict['a11'], turning_dict['a12'], df)
    print('区间[a13,a14]的拟合结果: ')
    line9 = linear_fit(turning_dict['a13'], turning_dict['a14'], df)
    print('区间[a15,a16]的拟合结果: ')
    line10 = linear_fit(turning_dict['a15'], turning_dict['a16'], df)
    print("区间[a17,]的拟合结果: ")
    line11 = linear_fit(turning_dict['a17'], df['x'].iloc[-1], df)

    # 计算角度
    angle = -angle_search(line1, line2, line3, line4, line7, line8,
        line9, line10, line11)
```

```
160     print(f'angle= {angle}')
161
162     # 坐标变换
163     df = coordinate_tranaform(df, angle)
164
165     # 计算新的转折点坐标
166     turning_points_x_new = []
167     turning_points_z_new = []
168     for i in range(len(turning_points)):
169         turning_points_x_new.append(turning_points_x[i] *
                np.cos(angle) - turning_points_z[i] * np.sin(angle))
170         turning_points_z_new.append(turning_points_x[i] *
                np.sin(angle) + turning_points_z[i] * np.cos(angle))
171     turning_points_x=turning_points_x_new
172     turning_points_z=turning_points_z_new
173
174     # 转折点更新
175     for i in range(len(turning_points)):
176         turning_dict[f'a{i + 1}'] = turning_points_x[i]
177         turning_dict[f'b{i + 1}'] = turning_points_z[i]
178
179     # 绘图
180     plt.plot(df['x'], df['z'], color='black', linewidth=3)
181     # 添加转折点
182     plt.scatter(turning_points_x, turning_points_z, color='black')
183     for i in range(len(turning_points)):
184         plt.text(turning_points_x[i], turning_points_z[i] + 0.08,
                f'a{i + 1}', fontsize=15)
185     plt.title('工件1_down')
186     plt.xlabel('x')
187     plt.ylabel('z')
188     plt.grid(True, linestyle='--', alpha=0.7)
189     plt.tight_layout()
190     plt.savefig('Figures/工件1_down.png')
191     plt.show()
192     plt.close()
193
194     # 直线区段拟合
195     print("区间[,a1]的拟合结果")
196     line1 = linear_fit(df['x'][0], turning_dict['a1'], df)
```

```
    print('区间[a2,a3]的拟合结果: ')
    line2 = linear_fit(turning_dict['a2'], turning_dict['a3'], df)
    print('区间[a4,a5]的拟合结果: ')
    line3 = linear_fit(turning_dict['a4'], turning_dict['a5'], df)
    print('区间[a6,a7]的拟合结果: ')
    line4 = linear_fit(turning_dict['a6'], turning_dict['a7'],df)
    print('区间[a7,a8]的拟合结果: ')
    line5 = linear_fit(turning_dict['a7'], turning_dict['a8'],df)
    print('区间[a8,a9]的拟合结果: ')
    line6 = linear_fit(turning_dict['a8'], turning_dict['a9'],df)
    print('区间[a9,a10]的拟合结果: ')
    line7 = linear_fit(turning_dict['a9'], turning_dict['a10'],df)
    print('区间[a11,a12]的拟合结果: ')
    line8 = linear_fit(turning_dict['a11'], turning_dict['a12'],df)
    print('区间[a13,a14]的拟合结果: ')
    line9 = linear_fit(turning_dict['a13'], turning_dict['a14'],df)
    print('区间[a15,a16]的拟合结果: ')
    line10 = linear_fit(turning_dict['a15'], turning_dict['a16'],df)
    print("区间[a17,]的拟合结果: ")
    line11 = linear_fit(turning_dict['a17'], df['x'].iloc[-1],df)

    # 更新转折点坐标
    point = line_intersection(line4, line5)
    turning_dict['a7'] = point[0]
    turning_dict['b7'] = point[1]
    point = line_intersection(line5, line6)
    turning_dict['a8'] = point[0]
    turning_dict['b8'] = point[1]
    point = line_intersection(line6, line7)
    turning_dict['a9'] = point[0]
    turning_dict['b9'] = point[1]

    # 计算z1
    z1 = turning_dict['b8'] - turning_dict['b9']
    print(f'z1={z1:.8f}\n')

    # 弧线区段拟合
    print('区间[a10,a11]的拟合结果: ')
    arc1 = arc_fit(turning_dict['a10'], turning_dict['a11'],df)
    print('区间[a12,a13]的拟合结果: ')
```

```
    arc2 = arc_fit(turning_dict['a12'], turning_dict['a13'],df)
    print('区间[a14,a15]的拟合结果: ')
    arc3 = arc_fit(turning_dict['a14'], turning_dict['a15'],df)
    print('区间[a16,a17]的拟合结果: ')
    arc4 = arc_fit(turning_dict['a16'], turning_dict['a17'],df)

    # 混合区段拟合
    print('区间[a1,a2]的拟合结果: ')
    mixed_line1, mixed_line2, mixed_arc1 =
        mixed_fit(turning_dict['a1'], turning_dict['a2'], 7.5, 5.3,
        5.2,df)
    print('区间[a3,a4]的拟合结果: ')
    mixed_line3, mixed_line4, mixed_arc2 =
        mixed_fit(turning_dict['a3'], turning_dict['a4'], 7.7, 5.3,
        5.2,df)
    print('区间[a5,a6]的拟合结果: ')
    mixed_line5, mixed_line6, mixed_arc3 =
        mixed_fit(turning_dict['a5'], turning_dict['a6'], 7.7, 5.3,
        5.3,df)

    # 更新转折点坐标
    point = line_intersection(line1, mixed_line1)
    turning_dict['a1'] = point[0]
    turning_dict['b1'] = point[1]
    point = line_intersection(line2, mixed_line2)
    turning_dict['a2'] = point[0]
    turning_dict['b2'] = point[1]
    point = line_intersection(line2, mixed_line3)
    turning_dict['a3'] = point[0]
    turning_dict['b3'] = point[1]
    point = line_intersection(mixed_line4, line3)
    turning_dict['a4'] = point[0]
    turning_dict['b4'] = point[1]
    point = line_intersection(mixed_line5, line3)
    turning_dict['a5'] = point[0]
    turning_dict['b5'] = point[1]
    point = line_intersection(mixed_line6, line4)
    turning_dict['a6'] = point[0]
    turning_dict['b6'] = point[1]
    point = line_circle_intersection(line7, arc1)[0]
```

```
    turning_dict['a10'] = point[0]
    turning_dict['b10'] = point[1]
    point = line_circle_intersection(line8, arc1)[1]
    turning_dict['a11'] = point[0]
    turning_dict['b11'] = point[1]
    point = line_circle_intersection(line8, arc2)[0]
    turning_dict['a12'] = point[0]
    turning_dict['b12'] = point[1]
    point = line_circle_intersection(line9, arc2)[1]
    turning_dict['a13'] = point[0]
    turning_dict['b13'] = point[1]
    point = line_circle_intersection(line9, arc3)[0]
    turning_dict['a14'] = point[0]
    turning_dict['b14'] = point[1]
    point = line_circle_intersection(line10, arc3)[1]
    turning_dict['a15'] = point[0]
    turning_dict['b15'] = point[1]
    point = line_circle_intersection(line10, arc4)[0]
    turning_dict['a16'] = point[0]
    turning_dict['b16'] = point[1]
    point = line_circle_intersection(line11, arc4)[1]
    turning_dict['a17'] = point[0]
    turning_dict['b17'] = point[1]
    point = line_circle_intersection(mixed_line1, mixed_arc1)
    turning_dict['a18'] = point[0]
    turning_dict['b18'] = point[1]
    point = line_circle_intersection(mixed_line2, mixed_arc1)
    turning_dict['a19'] = point[0]
    turning_dict['b19'] = point[1]
    point = line_circle_intersection(mixed_line3, mixed_arc2)
    turning_dict['a20'] = point[0]
    turning_dict['b20'] = point[1]
    point = line_circle_intersection(mixed_line4, mixed_arc2)
    turning_dict['a21'] = point[0]
    turning_dict['b21'] = point[1]
    point = line_circle_intersection(mixed_line5, mixed_arc3)
    turning_dict['a22'] = point[0]
    turning_dict['b22'] = point[1]
    point = line_circle_intersection(mixed_line6, mixed_arc3)
    turning_dict['a23'] = point[0]
```

```
    turning_dict['b23'] = point[1]

    # 打印转折点列表
    print("转折点列表")
    for i in range(23):
        print(f'(a{i + 1},b{i + 1})=({turning_dict[f'a{i +
            1}']:.8f},{turning_dict[f'b{i + 1}']:.8f})')
    print("圆参数")
    for i, arc in enumerate([mixed_arc1, mixed_arc2, mixed_arc3], 1):
        h, k, r = arc
        print(f'O{i}=({h:.8f},{k:.8f}) 半径={r:.8f}')
    for i, arc in enumerate([arc1, arc2, arc3, arc4], 4):
        h, k, r = arc
        print(f'O{i}=({h:.8f},{k:.8f}) 半径={r:.8f}')

    # 绘制总图
    plot_all_results(df)
```