基于数值分析的“板凳龙”路径仿真

摘要

本模型核心在于通过数学建模方法对传统民俗活动——板凳龙进行仿真模拟，为板凳龙中“盘龙”线路规划提出合理建议以优化其表演路径，提高安全性和观赏性。

对于问题一, 首先，明确螺线的具体类型，确立参数方程的研究方向；其次，由内至外求出给定圈数和总弧长；之后，通过对牛顿迭代和微元法思想的运用，用数值方法求解龙头前把手每个时刻的位置与速度。通过分析相邻节点之间的速度和位置关系，本研究建立了递推模型，描述并求解板凳龙在盘入螺旋路径过程中的动态变化。

对于问题二, 我们系统的考虑可能发生的碰撞情况，在二维空间中考虑直径卡滞和板板碰撞的情况并同时在三维空间中探索当板凳间有高度差时是否有把手碰撞板面可能性。将其转换为可量化的不等式条件，通过第一问获取的具体信息分别进行求解，并对结果进行对比分析，获取使得板凳之间不发生碰撞的最大终止时刻。

对于问题三, 根据题中最小化螺距需求确定优化目标函数，此优化模型的约束条件包括限定的掉头空间以及在第二个问题中通过计算得到的不发生碰撞的条件。据此建立用离散点逼近全局最优解的模型，通过在参数空间内系统地搜索离散点，逐步逼近全局最优解，得到满足条件的最小化螺距值。

对于问题四, 我们对于舞龙队盘入和盘出的过程，寻找使掉头路线最短的可能路径。在此基础上确定临界值，对整体时间范围进行分段拆分和模拟，通过运动学分析得到不同过程中前后板凳间的递推公式，求解每秒舞龙队的位置和速度。

对于问题五, 在前四问基础上进行总体分并在分段情况边界进行建模，求解出各时刻对应龙尾把手的位置，通过逆向工程方法逆推最小龙头速度，基于各把手的速度均不超过 $2m/s$ 的条件下寻求最优速度配置方案。

关键字： 仿真模拟　线性规划　二分法　牛顿迭代　阿基米德螺线

目录

一、 问题重述

1.1 问题背景

板凳龙是起源于浙闽地区的传统民俗活动之一，是当地人迎神、祈福的一项民俗活动，蕴含着深厚的地域文化。自改革开放以来，板凳龙迎来了传承与弘扬的新时期。现代板凳龙不仅保留了其传统特色，还逐渐发展成为一个综合性的艺术形式，巧妙地融合了书法、绘画、剪纸、刻花、雕塑等多种艺术手法，以及扎制、编糊等传统工艺，是重要的国家级非物质文化遗产。

盘龙作为板凳龙表演的一种特定形式，板凳龙队伍整体呈螺旋状排列。在此篇文章中，我们将以盘龙的表演形态为基础，对特定时间段内队伍中各个节点的位置、速度以及整体的螺距等关键指标进行探究。研究的目的是为了确定合理的螺距值，优化盘龙掉头时的行进路径。在确保表演安全性的基础上，提升板凳龙表演的娱乐性和观赏价值，从而增强其在文化推广和宣传方面的潜力。

1.2 问题要求

- 问题一: 固定盘龙的盘入轨迹为螺距为 55cm 的等距螺线, 顺时针从第十六圈开始盘入，盘入过程中行进速度恒定为 $1m/s$，需求出每秒板凳龙队伍所有节点的位置与速度，取 $0-300s$ 中指定位置坐标与速度，将结果存入.xlsx 文件并在文中给出要求关键值
- 问题二：在确保队列沿预定螺线路径盘入且板凳之间不会发生碰撞情况下，确定合适的终止时刻（即临界点）队列中各节点的位置和速度。
- 问题三：盘龙状态由盘入转变成逆时针掉头盘出状态，给定直径为 $r=4.5m$ 的圆作为掉头区域，求解使龙头前把手能够沿着相应的螺线盘入到调头空间的边界成立时的最小螺距。
- 问题四：在问题三给定掉头空间基础上，求解当盘入螺线的螺距为 1.7m，盘出与盘入螺线关于螺线中心呈中心对称，前一段圆弧的半径是后一段的 2 倍，且与盘入、盘出螺线均相切的掉头曲线，使掉头弧线弧长变短。在龙头前把手保持 1 m/s 情况下，求出从 $-100s$-$100s$ 每秒板凳龙各节点的位置和速度。
- 问题五：板凳龙沿问题 4 设定的路径行进且龙头行进速度保持不变，需确定龙头的最大行进速度，使得舞龙队各把手的速度均不超过 2 m/s。

二、 问题分析

2.1 问题一的分析

问题一首先需明确螺线的具体类型，并确立相应的数学参数方程。在此基础上，由内圈至外圈求出给定圈数和螺距下螺线线总弧长。进一步地，根据给定的恒定龙头行进速度，可逆向推出龙头行进距离，并基于极坐标写出每秒龙头位置（直角坐标系原点）。通过对该系统的运动学特性进行分析，可以建立板凳前后位置和速度的递推关系模型。带入计算程序，获取所需数据并写入表格。

2.2 问题二的分析

在题一解答的基础上，通过观察板凳龙运行过程中各节点的位置和速度，判断碰撞可能发生的情况，基于此求出不同可能情况下对应时间。选取更小的 t 值，即选取最先可能发生碰撞的情况作为本题解答。

2.3 问题三的分析

优化模型下，以等距螺线中螺距尽量大作为目标函数，第二问不碰撞条件以及掉头区域半径为约束函数，在满足约束条件的情况下，使得目标函数尽量大。进行逐步寻优，先确认半径，逐步寻找其他决策变量的优解，使得目标函数达到全局最优解。

2.4 问题四的分析

需依据题目所提供的条件，分段构建的 $-100-100s$ 模拟掉头路径。在此基础上，利用基本不等式确定一条弧长相比更短的最优路径。此外，还需对板凳龙在进入掉头路线后，即在题目规定的 $0-100s$ 内固定点的位置和速度进行分段详细分析。通过这一分析优化路径选择，确保在给定时间区间内实现最佳动态表现。

2.5 问题五的分析

基于问题四，总体分析同一时刻板凳龙各节点的速度以及同一节点随时间变化的趋势，在边界建立模型，基于各把手的速度均不超过 $2m/s$ 的条件寻求最优速度配置方案。

三、 模型假设

3.1 基本假设

- 假设板凳龙每一小节在进入题目描述的特定区域前，沿着与题目提供的相等恒定螺距的螺旋轨迹进行等距螺线运动。

- 假设所有把手中心位置都在给定等距螺线上，相邻板凳有两个把手的中心点连接。
- 假设板凳材料不具备受挤压收缩的性质，板凳间距离保持恒定。
- 假设每节板凳的速度连续，板凳间运动具有平滑性，可利用递推求出每个节点（把手）速度。

3.2 问题一附加假设

- 假设板凳穿孔的直径对运动不造成影响，把手中心点可代表把手的运动轨迹。
- 假设不存在碰撞情况，龙头前把手可以盘入至直角坐标系原点。
- 假设相邻的前后板凳不存在高度差且手握把手的“舞龙人”不存在身高差，所有板凳均在同一二维平面，所有运动不涉及竖直方向上的运动。

3.3 问题四附加假设

- 假设掉头轨迹半径以及掉头轨迹模型可根据题意进行简化。
- 假设龙头板凳长度无限接近于其他板凳长度。

四、 符号说明

4.1 问题一符号说明

序号	符号	符号说明
1	a	起点到极坐标原点的距离 (m)
2	b	螺旋线每增加单位角度 r 随之对应增加的数值 (m)
3	p	螺距 (m)
4	γ	把手距螺线原点的弧度 (rad)
5	θ	把手行进弧度 (rad)
6	Q	螺线圈数
7	d	龙头距外圈距离最近的板凳的平面距离 (m)
8	S	螺线总弧长 (m)

序号	符号	符号说明
9	γ_i	第 i 个节点在某固定时刻对应弧度 (rad)
10	h	步长 (m)
11	L_0	龙头有效板长 (2.86m)
12	L_c	除龙头外其他板有效板长 (1.65m)
13	w	模板宽度 (0.3m)
14	D	把手孔径 ($5.5 * 10^{-2}$m)
15	γ_{head}	把手距螺线原点的弧度 (rad)
16	t'	步长，即走入一节板凳的时间周期 (m)

五、 模型的建立与问题的求解

5.1 问题一模型的建立

依据题意给出的等距条件，建立起有关阿基米德螺线的参数方程，在极坐标系中，螺线的方程 [1] 为：

$$r = a + b\gamma$$

其中，a 为起点到极坐标原点的距离，b 为螺旋线每增加单位角度 r 随之对应增加的数值，即 $b = \frac{55}{2\pi}$。此阿基米德螺线参数 $a = 0, b = \frac{55}{2\pi}$，圈数 $Q = 16$，由定积分求弧长公式 [3] 可得 (其中 s 是有关 γ 的函数)：

$$s = \int \sqrt{s^2 + (\frac{ds}{d\gamma})^2}\, d\gamma$$
$$s = \int_0^{32\pi} \sqrt{(\frac{55}{2\pi})^2\gamma^2 + (\frac{55}{2\pi})^2}\, d\gamma$$

求解可得

$$s = a(\frac{\gamma}{2}\sqrt{1+\gamma^2} + \frac{1}{2}ln|\gamma + \sqrt{1+\gamma^2})|_0^{32\pi}$$

即总弧长为

$$s = \frac{55}{2\pi}(16\pi\sqrt{1+(32\pi)^2} + \frac{1}{2}ln(32\pi + \sqrt{1+(32\pi)^2}) = 44259.045643$$

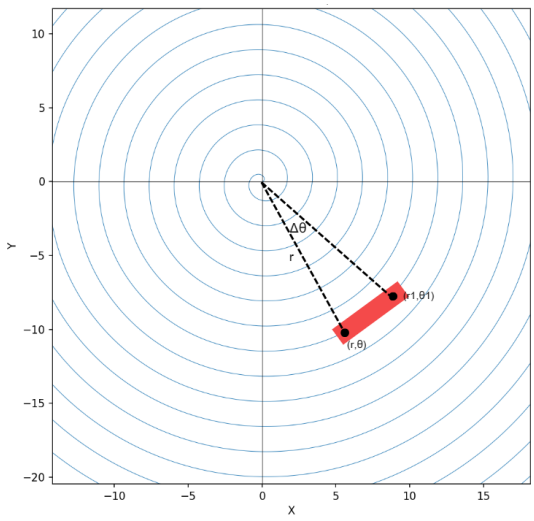
以龙头把手为锚点，已知行进速度为定值 $1m/s$。根据螺线总弧长，反向推导可得龙头行进距离为 $S - v_0t$，令龙头从原点起点所行驶弧度为 γ，$\theta = 32\pi - \gamma$, 为节点从第十六圈向内的行进弧度。那么，对于从第十六圈向内运动的龙头，可列出公式：

$$\frac{s - vt}{b} = \frac{r(\gamma)}{2}\sqrt{1 + r(\gamma)^2} + \frac{1}{2}ln|r(\gamma) + \sqrt{1 + r(\gamma)^2}|$$

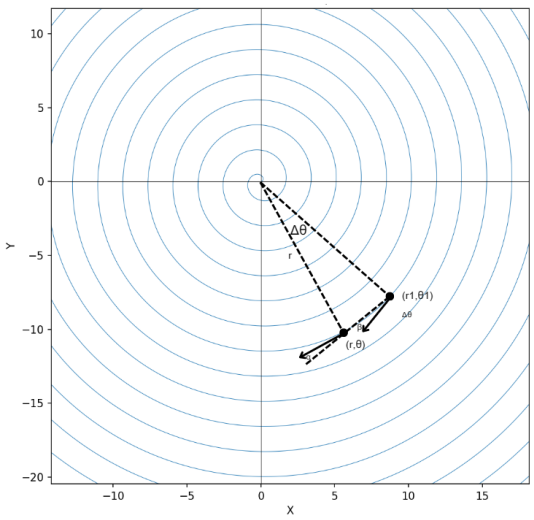
根据极坐标转换可得龙头位置关于行驶时间 t 的表达式:

$$x_{head}(t) = r_{head}(t) \cdot cos(\gamma_{head}(t))$$
$$y_{head}(t) = r_{head}(t) \cdot sin(\gamma_{head}(t))$$

以龙头递推第一节龙身前把手位置为例，第一节龙身前把手和原点连线与龙头把手和



(a) 位置递推示意图（以第一节龙身为例）



(b) 速度递推示意图（以第一节龙身为例

原点连线满足三角函数，其位置如图1a所示

$$L_0^2 = r_0^2 + r_1^2 - 2 * r_0 * r_1 * cos(\triangle\theta)$$

v_1 为 s 关于 t 的导数的同时，也满足递推式

$$v_0 * sin(\pi - \alpha) = v_1 * sin(\pi - \beta)$$

递推可得

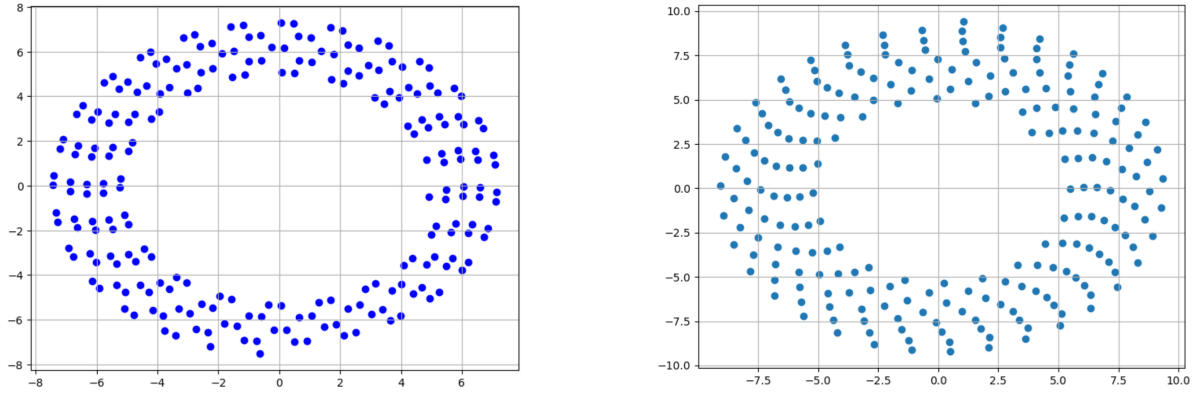
$$v_1 = v_0 * \frac{sin\alpha}{sin\beta} = \frac{r_1}{r_0}$$
$$v_2 = v_1 * \frac{sin\alpha}{sin\beta} = v_0 * \frac{sin\alpha}{sin\beta} * \frac{r_2}{r_1}$$
$$v_i = v_0 * \frac{r_i}{r_0}$$

在每秒通过龙头位置及其固定速度对各节点位置速度进行递推求解。

5.2 问题一模型的求解

将模型一的参数代入计算程序中进行求解，在通过 $L_0^2 = r_0^2 + r_1^2 - 2 * r_0 * r_1 * cos(\triangle\theta)$ 计算各把手行进位置过程中，采用基于 Newton-Raphson 的求数值解方法的 $root$ 函数来确定未知 $\triangle\theta$ 的值。考虑 $cos\theta$ 具有 2π 为周期的周期性特征，设置函数在每次迭代时逼

近 $\theta_0 - \triangle\theta$, 即理论上的 $\theta 1$ 值，但由于所涉及到所有计算数值均较小，$\triangle\theta$ 由于数值方法与实际存在偏差，$cos\theta$ 在给定逼近值左右两边都有较接近值且在循环中难以每次都取到准确初值等多种原因，$root$ 函数计算结果出现部分点位置不精确的情况，如图2a所示。



(a) 牛顿迭代　　**(b)** 微元思想

图 **2**　不同思路下第 $300s$ 各把手位置求解

因此，引入微元法相关思想，逆向对每个节点对应 θ 值进行求解，设定步长 $h = 0.001$，每循环令弧度 $\theta_{i+1} = \theta_i + h$。与此同时，令同在螺线上的两点距离（由欧式距离计算得出）与板凳有效长的差值为 $\triangle L$(板凳有效长 $L_0 = 2.86m$，其余 $L = 1.65m$)，判断 $\triangle L$ 是否 $< 0.01m$。

由于数据精度高，对于部分较难取到合适值的节点，在 $\triangle L_i < 0$ 且 $\triangle L_i > 0$ 时，将 $\triangle L_{i+1}$ 与 $\triangle L_i$ 的平均值作为节点的 θ, 并在此基础上进行位置和速度递推，结果如表3和4所示：

5.2.1 数据结果分析

- 对于同一节点不同时间
  - 同一节点轨迹为等距螺线。
  - 除龙头前把手行进速度保持恒定外，其余把手节点速度随时间增大逐渐增长，且在相同时间差 $\triangle t$ 内，越靠近龙尾的节点，速度增长越快，即加速度越大。
- 对于同一时间不同节点
  - 整体轨迹呈等距螺线。
  - 观察同一时刻所有节点，越靠近龙尾后把手节点速度越快，即从龙头至龙尾节点速度呈上升趋势。

	$0s$	$60s$	$120s$	$180s$	$240s$	$300s$
龙头 $x(m)$	8.800000	5.799209	-4.084887	-2.963609	2.594494	4.420274
龙头 $y(m)$	0.000000	-5.771092	-6.304479	6.09478	-5.356743	2.320429
第 1 节龙身 $x(m)$	8.363824	7.456758	-1.445473	-5.237118	4.821221	2.459489
第 1 节龙身 $y(m)$	2.826544	-3.440399	-7.405883	4.359627	-3.561949	4.402476
第 51 节龙身 $x(m)$	-9.515993	-8.692692	-5.536588	2.882407	5.998319	-6.301432
第 51 节龙身 $y(m)$	1.359255	2.518974	6.383524	7.252598	-3.799774	0.464845
第 101 节龙身 $x(m)$	2.897887	5.703594	5.351534	1.919509	-4.895743	-6.241319
第 101 节龙身 $y(m)$	-9.922878	-7.989866	-7.564863	-8.467164	-6.396863	3.93045
第 151 节龙身 $x(m)$	9.305829	6.667084	2.406079	0.991941	2.941486	7.043423
第 151 节龙身 $y(m)$	1.810861	8.147242	9.72298	9.426274	8.408382	4.388608
第 201 节龙身 $x(m)$	4.578571	-6.625884	-10.62677	-9.278034	-7.442329	-7.45391
第 201 节龙身 $y(m)$	10.714911	9.021079	1.363073	-4.268278	-6.198897	-5.270251
龙尾后 $x(m)$	-5.323877	7.368686	10.974251	7.366113	3.213866	1.783554
龙尾后 $y(m)$	-10.667235	-8.794587	0.844882	7.510148	9.478864	9.301462

图 **3**　$result1$ 各节把手位置

	$0s$	$60s$	$120s$	$180s$	$240s$	$300s$
龙头 (m/s)	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
第 1 节龙身 (m/s)	1.003242	1.003752	1.004453	1.005476	1.00711	1.010133
第 51 节龙身 (m/s)	1.092339	1.106197	1.124846	1.151579	1.192976	1.265656
第 101 节龙身 (m/s)	1.174701	1.199879	1.233517	1.281078	1.353384	1.477432
第 151 节龙身 (m/s)	1.251643	1.286746	1.333338	1.398577	1.496651	1.66231
第 201 节龙身 (m/s)	1.324108	1.368087	1.426196	1.506945	1.627322	1.828585
龙尾后 (m/s)	1.35477	1.402383	1.465185	1.552223	1.681605	1.897099

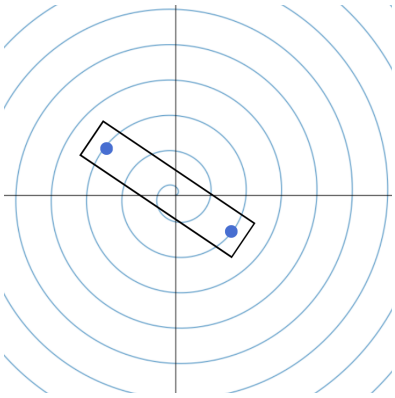
图 **4**　$result1$ 各节把手速度

5.3 问题二模型的建立

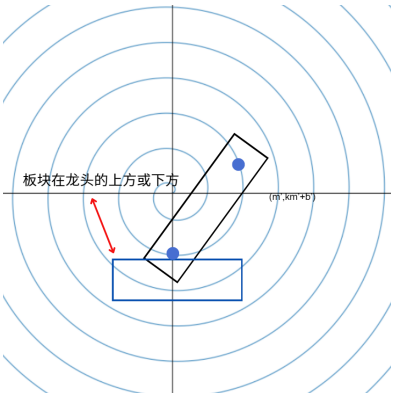
在模拟板凳龙的运行过程中，可能发生碰撞的情况主要包括以下三种情形 [2]：

- 直径卡滞模型：在板凳龙运行过程中，当龙头板凳近似成为螺线直径，龙头部位发生卡滞，无法再向前运行，如图5a所示，此种情况满足：

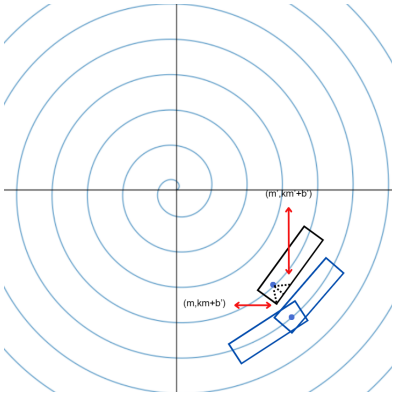
$$b\gamma + b(\gamma + \frac{\pi}{2}) = L_0$$



(a) 情况一示意图



(b) 情况二示意图



(c) 情况三示意图

- 三维把碰板碰撞模型：在运行过程中假设由于相邻前后板凳拼接和手持把手舞龙人身高造成的的高度差，可能会出现龙头前把手与外圈板碰撞的情况，碰撞条件满足：

$$d_2 \leq w + \frac{D}{2}$$

此处 d 指龙头把手中心相邻外圈的板凳的最近边的距离。

- 二维板碰板碰撞模型：在运行过程中假设不存在高度差，所有板凳运行在同一二维平面，可能会出现龙头板与外圈板碰撞的情况，碰撞条件满足以下方程，即

$$d_3 \leq \frac{D}{2} \tag{1}$$

此处 d 指板头对应顶点 A 与相邻外圈的板凳的最近边的距离。

我们将分别计算三种潜在碰撞情况对应的时间，并从中确定最早发生碰撞的时间节点。

5.4 问题二模型的求解

- 对于直径卡滞模型，可以由运算直接得出 γ 为 2.7225, 即 $32\pi - 2.7225$
- 对于三维把碰板碰撞模型：设龙头前把手坐标为 (x_0, y_0)

$$d_2 = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}} = 0.1525$$

	横坐标 $x(m)$	纵坐标 $y(m)$	速度 $(m/s$
龙头 (m/s)	3.299984	0.175616	1.000000
第 1 节龙身 (m/s)	2.003351	2.7248	1.023407
第 51 节龙身 (m/s)	-0.490812	5.072253	1.542051
第 101 节龙身 (m/s)	-5.798	-2.619167	1.925206
第 151 节龙身 (m/s)	-7.32615	1.122784	2.242803
第 201 节龙身 (m/s)	5.026079	6.643861	2.520932
龙尾后 (m/s)	4.102846	-7.677999	2.634303

图 **6**　终止时刻位置与速度

- 对于二维板碰板碰撞模型, 推导运算过程如下，对于与龙头对应板碰撞的板块，存在两条平行的直线 (板的两条长)，设两条线段所在直线方程为

$$kx - y + b = 0$$
$$kx - y + b' = 0$$
$$\frac{|b' - b|}{\sqrt{1 + k^2}} = d_2 = 0.15\,\mathrm{m}$$

计算原点至两条直线分别的距离，始终保留离原点更远的直线，令选取直线为 $kx - y + b' = 0$，在 $d_3 \leq \frac{D}{2}$ 条件下，求解横坐标 m。

$$\begin{aligned}
\frac{D}{2} &= (m - x_0)^2 + (km + b' - y_0)^2 \\
\frac{157}{1600} &= m^2 - 2mx_0 + x_0^2 + k^2m^2 + 2k(b' - y_0) * m + +(b' - y_0)^2 \\
0 &= (1 + k^2)m^2 + (2k(b' - y_0))m + x_0^2 + (b' - y_0)^2 - \frac{157}{1600} \\
\triangle &= (2k(b' - y_0) - 2x_0)^2 - 4(1 + k^2) * (x_0^2 + (b' - y_0)^2) - \frac{157}{1600}
\end{aligned}$$

可解得横坐标

$$m_1 = \frac{2x_0 - 2k(b' - y_0) + \sqrt{\triangle}}{2(1 + k^2)}, m_2 = \frac{2x_0 - 2k(b' - y_0) - \sqrt{\triangle}}{2(1 + k^2)}$$

根据图5c所示，非顶点在模拟过程中不会与其他物体发生碰撞。因此，我们将采用所有计算得到的 m 值，并将其应用于后续每个时间点的模拟计算中，以确定碰撞发生的具体位置和对应的时间点。经过计算得到龙头前把手、龙身第 1、51、101、151、201 条龙身前把手和龙尾后把手的位置和速度如表6所示。

螺距 (cm)	半径 (m)	碰撞时间 (s)
50	1.878907823	380
51	2.10215711	383
52	2.415837767	383
53	2.508608406	389
54	3.056155576	380
55	3.304653981	380
56	3.440907408	384
57	3.524476715	390
58	3.879076511	385
59	4.05479401	387
60	3.671860473	412
61	4.263836273	397
62	4.736351777	385

图 **7**　螺距大小与碰撞关系呈现

5.5 问题三模型的建立与求解

依据题目条件，建立同时满足瞬间把手中心 $r = 4.5m$，螺距最小以及模型不碰撞的优化模型

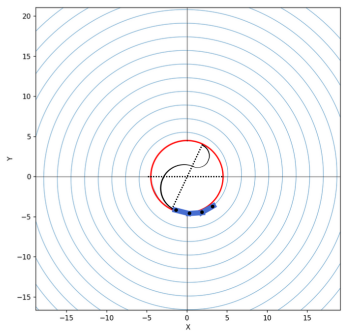
$$\begin{cases} \text{minimize} & p = \dfrac{9\pi}{32\pi - \gamma_0} \\ & d \geq 15 + \dfrac{5.5}{2} \\ & r = 4.5 \end{cases}$$

通过综合分析第二问中得出的各类数据，并对其进行详尽的遍历处理，我们可以得到螺距所对应碰撞时间以及碰撞点所对应半径 r 如下：由于题中给定调头空间是以螺线中心为圆心、直径为 $9m$ 的圆形区域, 可得最小螺距约为 $61.5cm$。

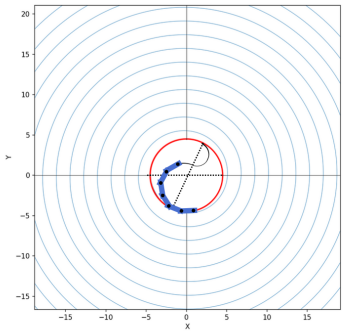
5.6 问题四模型的建立

由题意可得螺距 p 为 1.7 m，龙头前把手的行进速度始终保持 $1m/s$ 盘出螺线与盘入螺线关于螺线中心呈中心对称，设定舞龙队在掉头空间内完成掉头。由基本不等式简单推理可得，当且两段圆弧经过同一条直线时，调头曲线变短，掉头路线如图8a所示。在前几问认知基础上，分三段建立模型对包括掉头过程在内的 $-100 - 100s$ 位置和速度信息进行求解，三段分别如图8a，8b和8c所示。

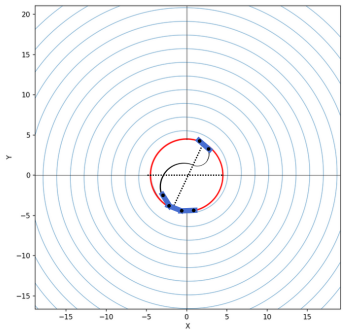
- -100-0s：板凳龙整体沿螺线正常盘入，除龙头前把手外其他节点经历加速过程。
- 0s-龙头前把手出掉头区域时刻：龙头及前几节龙身进入掉头区域内。
- 龙头前把手出掉头区域时刻-100s：龙头部分前把手已超出了指定掉头区域，其余龙身部分继续沿同一路径前进。



(a) 情况一示意图



(b) 情况二示意图



(c) 情况三示意图

5.6.1 阶段一模型的建立

阶段一模型与问题一呈相同趋势，由图观察此时 $a=0$，由螺距基本参数方程 $r=b\gamma$，可得龙头前把手进去掉头区域时刻满足条件 $\gamma=\frac{9\pi}{1.7}$。此时节点仍满足第一问递推关系：

$$L_0^2=r_0^2+r_1^2-2*r_0*r_1*cos(\triangle\theta)$$
$$v_0*sin(\pi-\alpha)=v_1*sin(\pi-\beta)$$

据此可递推出 $-100-0s$ 时任一节点的位置和速度。

5.6.2 阶段二模型的建立

根据最短掉头路径示意图求出掉头路线所需经历弧长为 $\pi*(3+1.5)=4.5\pi\approx 14.14m$，龙头前把手在掉头区域中的各时刻速度递推如下图1所示

由于两段圆弧相切，由三角函数可推得步长：

$$t'=\frac{2\pi arcsin(\frac{L}{2r})}{180^\circ}*r\approx 1.6$$

通过进入掉头区域的最后一节龙身，可由盘入螺线递推出每点的位置和速度。

5.6.3 阶段三模型的建立

当龙头前把手离开掉头区域瞬间，模型进入阶段 2，该阶段时间 t 范围为 $4.5\pi-110s$，板凳龙整体始终有一部分处在盘出螺线，一部分处于掉头区域内，此时假设第 $i-1$ 块板

	0*t'	1*t'	2*t'	3*t'	4*t'	5*t'	6*t'	7*t'	8*t'
龙头	v_0	v_0	v_0	v_0	v_0	v_0	v_0	v_0	v_0
第一节龙身	$\frac{r_1}{r_0} * v_0$	v_0	v_0	v_0	v_0	v_0	v_0	v_0	v_0
第二节龙身	$\frac{r_2}{r_0} * v_0$	$\frac{r_1}{r_0} * v_0$	v_0	v_0	v_0	v_0	v_0	v_0	v_0
第三节龙身	$\frac{r_3}{r_0} * v_0$	$\frac{r_2}{r_0} * v_0$	$\frac{r_1}{r_0} * v_0$	v_0	v_0	v_0	v_0	v_0	v_0
第四节龙身	$\frac{r_4}{r_0} * v_0$	$\frac{r_3}{r_0} * v_0 3$	$\frac{r_2}{r_0} * v_0$	$\frac{r_1}{r_0} * v_0$	v_0	v_0	v_0	v_0	v_0
第五节龙身	$\frac{r_5}{r_0} * v_0$	$\frac{r_4}{r_0} * v_0$	$\frac{r_3}{r_0} * v_0$	$\frac{r_2}{r_0} * v_0$	$\frac{r_1}{r_0} * v_0$	v_0	v_0	v_0	v_0
第六节龙身	$\frac{r_6}{r_0} * v_0$	$\frac{r_5}{r_0} * v_0$	$\frac{r_4}{r_0} * v_0$	$\frac{r_3}{r_0} * v_0$	$\frac{r_2}{r_0} * v_0$	$\frac{r_1}{r_0} * v_0$	v_0	v_0	v_0
第七节龙身	$\frac{r_7}{r_0} * v_0$	$\frac{r_6}{r_0} * v_0$	$\frac{r_5}{r_0} * v_0$	$\frac{r_4}{r_0} * v_0$	$\frac{r_3}{r_0} * v_0$	$\frac{r_2}{r_0} * v_0$	$\frac{r_1}{r_0} * v_0$	v_0	v_0
第八节龙身	$\frac{r_8}{r_0} * v_0$	$\frac{r_7}{r_0} * v_0$	$\frac{r_6}{r_0} * v_0$	$\frac{r_5}{r_0} * v_0$	$\frac{r_4}{r_0} * v_0$	$\frac{r_3}{r_0} * v_0$	$\frac{r_2}{r_0} * v_0$	$\frac{r_1}{r_0} * v_0$	v_0
第九节龙身	$\frac{r_9}{r_0} * v_0$	$\frac{r_8}{r_0} * v_0$	$\frac{r_7}{r_0} * v_0$	$\frac{r_6}{r_0} * v_0$	$\frac{r_5}{r_0} * v_0$	$\frac{r_4}{r_0} * v_0$	$\frac{r_3}{r_0} * v_0$	$\frac{r_2}{r_0} * v_0$	$\frac{r_1}{r_0} * v_0$

表 **1**　情况 **2** 各节点速度递推

正在离开掉头区域，此时可由盘出螺线方程以及龙头前把手坐标递推出第 $i-1$ 板前把手位置坐标 (r_i, θ_i)，根据盘入和盘出螺线的中心对称性可得 $(r_i, \theta_i) = (r_{i+10}, \theta_{i+10})$，继续进行递推可得任一把手位置。

针对速度 v, 可由盘出螺线方程同样通过问题 1 中递推公式从 v_0 逆推至 v_i，同时根据圆弧上各点运动速度大小相等，可以得到

$$v_i = v_{i+1} = v_{i+2} = ... = v_{i+9}$$

从 v_{i+9} 可由第一问得出的盘入螺线位置与速度递推公式，递推出任一节点的位置和速度。

5.6.4 基于弧线半径调整的模型的建立

由于在不同半径圆弧上减速运动的复杂性，以及问题四给出的“能否调整圆弧，仍保持各部分相切，使得调头曲线变短”的条件，可由基本不等式得出最短掉头轨迹为两个相切的半径 $r = \frac{9}{4}m$ 的半圆。由轨迹对称性可得给定时间节点位置和速度呈现在表格 9和 10中。

5.7 问题四模型的求解

	-100s	-50s	0s	50s	100s
龙头 $x(m)$	8.114087	6.770785	-3.368062	2.58018	1.107567
龙头 $y(m)$	2.821146	0.956084	-2.896377	5.703593	7.960922
第 1 节龙身 $x(m)$	7.01429	6.389503	-2.73958	1.182728	-1.128568
第 1 节龙身 $y(m)$	5.095932	2.618487	-3.567203	6.212519	8.033879
第 51 节龙身 $x(m)$	-6.989585	-5.015423	-3.930349	-5.347988	-1.94745
第 51 节龙身 $y(m)$	-8.871894	7.573062	-4.780576	6.302041	-10.394022
第 101 节龙身 $x(m)$	-11.984064	11.581299	0.029986	5.433503	12.753724
第 101 节龙身 $y(m)$	10.532876	-2.475983	-8.075949	-9.296725	-6.516722
第 151 节龙身 $x(m)$	-23.87089	4.846491	3.816306	9.226953	-23.760469
第 151 节龙身 $y(m)$	-6.427809	-19.446249	9.808317	11.432272	-1.668312
第 201 节龙身 $x(m)$	6.351661	18.048186	-9.830684	-13.320931	-19.82267
第 201 节龙身 $y(m)$	26.820232	-17.715751	10.290198	-20.043436	18.329168
龙尾后 $x(m)$	13.103671	-13.656628	16.710078	-24.43103	28.019657
龙尾后 $y(m)$	-25.425744	-0.711752	-11.416788	-7.585207	-1.061458

图 **9**　$result4$ 各节把手位置

	-100s	-50s	0s	50s	100s
龙头 (m/s)	1.000000	1.000000	1.000000	1.000000	1.000000
第 1 节龙身 (m/s)	0.9919	0.994181	0.996014	0.994725	0.992603
第 51 节龙身 (m/s)	0.724356	0.805803	0.874862	0.834815	0.750042
第 101 节龙身 (m/s)	0.249258	0.567113	0.739656	0.62866	0.374337
第 151 节龙身 (m/s)	-0.644395	-0.141937	0.564228	0.305498	-0.544494
第 201 节龙身 (m/s)	-0.934004	-0.595917	0.298662	-0.466748	-0.847696
龙尾后 (m/s)	-1.040194	1.402383	-0.13167	-0.591555	-0.949202

图 **10**　$result4$ 各节把手速度

5.8 问题五模型的建立与求解

基于问题四四个模型与掉头过程的分析，推断出当龙头前把手进入掉头过程的瞬间，龙尾后把手达到整个掉头过程中所有节点速度的最大值，因此再次运用在螺线 $r = \frac{1.7}{2\pi} * \gamma$ 上速度的递推关系。当龙头进入掉头区域瞬间，龙尾后把手速度为 $2m/s$ 时，龙头行进速度最大。

$$\frac{r_{223}}{r_{0max}} * v_0 = 2m/s$$

$$v_0 \approx 0.4447 m/s$$

因此，使得舞龙队各把手的速度均不超过 $2m/s$ 的最大龙头速度约为 $0.4447m/s$。

六、 模型的评价，改进与推广

6.1 模型的评价

6.1.1 优点

- 数据较为精确，运用牛顿迭代，微元法等多个方法进行比较分析，求解每秒整个舞龙队的位置和速度。
- 该模型设计考虑了广泛的适用性，从二维和三维两个不同维度考虑板凳的碰撞情况情况。
- 该模型具有较高的灵活性，能够根据不同的步长、螺距以及时间变量进行较为便捷的调整。

6.1.2 缺点

- 将龙头划出掉头空间后后续节点在圆弧上的变速看为匀减速，数值会存在一定偏差。
- 模型四为简化模型，未考虑掉头路线的两个相切圆弧半径不同对于整个模型的影响。
- 本模型假设了相对严格的板间位置，未考虑板前后之间的拉扯产生的形变以及铆钉之间缝隙对板凳龙各节点速度的影响。

6.2 模型的改进与推广

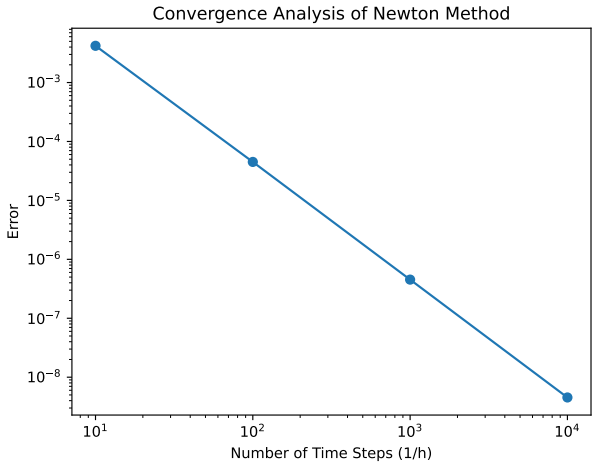
- 本题将模型进入盘入螺线前的轨迹均设定为相同螺距等螺距曲线的外圈，在生活中，可能存在各种进入队形的不规则路径。可从不同进入轨迹出发，讨论板凳龙演出的最优路径。
- 在本研究中，所采用的模型主要依赖于离散数值方法对变量进行逼近和求解。可考虑采用拟合方法，将离散数据点与适当的数学函数表达式相匹配，构建一个连续的函数模型，从而为变量间的关系提供更精确的描述。在有助于读者深入理解变量之间的动态关联的同时，使模型更具有普适性。

七、 模型的检验

基于本题建立的模型，我们可以从数值方法精度，边界条件和变化趋势上来证明其可行性和准确性。

7.1 方法精度检验

本文在推论过程中运用牛顿迭代的数值求解方法，由于文章内式子的解析解难以获得，以 $\frac{\partial y}{\partial x} = \frac{2y}{t} + 4t^2 cos(4t)$ 这个易得出解析解的偏微分方程为例，求误差以及收敛阶如下图所示：



(a) 牛顿迭代收敛

Step Size	Error	Convergence Rate	Runtime
1.00e-01	4.20e-03	0.0000	0.00
1.00e-02	4.50e-05	1.9705	0.00
1.00e-03	4.53e-07	1.9971	0.01
1.00e-04	4.53e-09	1.9997	0.04

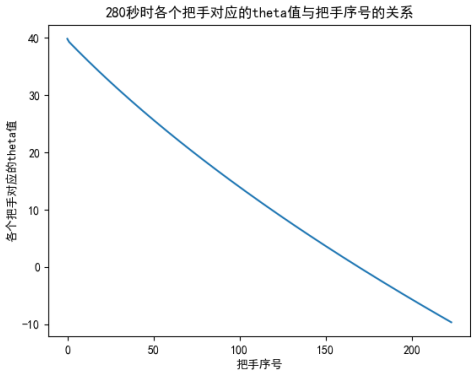
由此证明牛顿迭代具有二阶精度，计算坐标数值较为准确。

7.2 边界条件检验

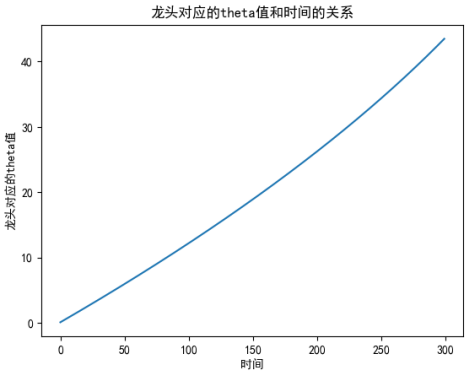
验证模型在初始和最终时刻的状态是否符合预期，我们再次确认了代码算出的龙头的初始速度 $v_0 = 0$ 和位置 $(8.8, 0)$ 等端点数值，这些计算结果与模型的预期行为相吻合，从而验证了模型在边界条件下的有效性和准确性。

7.3 变化趋势检验

在第一二题目条件下，由于盘入运动的单项性，在分析把手的角位置（行进弧度）θ，我们观察到每个把手的 θ 始终小于其前一个把手值。因此对于任意给定的时刻，把手的 θ 值呈现出递减趋势。这种排列方式与所提供的数据一致，验证了把手角位置的递减性假设。



(a) 280s 时各个把手对应的 θ 值与把手序号的关系



(b) 龙头对应的 θ 值和时间的关系

参考文献

[1] Banner, A. D. The calculus lifesaver : all the tools you need to excel at calculus. Princeton University Press, 2007.

[2] 王栋, 周可璞. 基于阿基米德螺线走法的全区域覆盖路径规划 / Path Planning of the District Board Based on Archimedes’Spiral. 工业控制计算机 / Industrial Control Computer, 2018, 31(5), 83-87.

[3] 郑立飞, 魏宁, 万阿英. 如何用元素法解决阿基米德螺线的弧长. 呼伦贝尔学院学报 / JOURNAL OF HULUNBEIER UNIVERSITIES, 2021, 19(2):95-97.

程序一：Python 模拟问题一盘入曲线位置与速度

```python
import numpy as np
from sympy import symbols, integrate, sqrt
from scipy.optimize import root
import pandas as pd


# 螺距比例
b = 170/(2*np.pi)

theta = symbols('theta')

# 定义被积函数
f = sqrt(b**2 * theta**2 + b**2)

# 计算积分
Total_Length = integrate(f, (theta, 0, 32*np.pi))

def r_length(theta):
    return 880-b * theta

# 坐标转换方程
def positionX(theta):
    return (b*(32*np.pi-theta)*np.cos(32*np.pi-theta))/100

def positionY(theta):
    return (b*(32*np.pi-theta)*np.sin(32*np.pi-theta))/100

# 两个孔距
L1 = 341-27.5*2
L2 = 220-27.5*2
# 龙头的速度
V0 = 100

def equation(x, V0, t):
```

```
35      return 0.5 * x * np.sqrt(1 + x**2) + 0.5 * np.log(np.abs(x
        + np.sqrt(1 + x**2))) - (float(Total_Length) - V0 * t) / b
36
37
38  # 龙头对应时间的角度
39  solutions = []
40
41  # 找每个时刻头对应的角度，从里往外
42  for t_value in range(442):
43  # for t_value in range(1510):
44      result_head_theta = root(equation, 1, args=(V0, t_value))
45      if result_head_theta.success:
46          solutions.append(result_head_theta.x[0])
47      else:
48          solutions.append(None)  # 如果没有找到解，则存储None
49
50
51
52  theta_rad = 32 * np.pi - np.array(solutions)
53  mei_miao_long_tou_de_theta=theta_rad.tolist()  #龙头对应时间的
        角度
54  # print(mei_miao_long_tou_de_theta)
55  # x=positionX(theta_rad)
56  # y=positionY(theta_rad)
57  # plt.plot(x,y)
58
59  # 两点间距离判断公式
60  def eq_others(x,theta):
61      return sqrt(r_length(theta)**2 + r_length(x)**2 - 2 *
        r_length(theta) * r_length(x) * np.cos(theta - x)) - L2
62
63  # 龙头专用距离判断公式，root（）备用
64  def eq_head(x,theta_temp,L1):
65      return np.sqrt(r_length(theta_temp)**2 + r_length(x)**2 -
        2 * r_length(theta_temp) * r_length(x) * np.cos(theta_temp -
```

```
    x)) - L1
66
67  # 返回某个时刻，某个把手的theta 值
68  # body_theta是返回的把手的前一个的theta值
69  def longshenliebiaodetheta(body_theta):
70      step=0.05
71      panduan=True
72      theta_Loong=[]
73      theta_Loong.append(body_theta)
74
75
76      while panduan:
77
78          theta_Loong_new_element=theta_Loong[-1]-step
79
80          if np.abs(eq_others(body_theta,theta_Loong_new_element
    ))<0.01:
81              panduan = False
82              return theta_Loong_new_element
83          elif round(eq_others(body_theta,theta_Loong[-1]),3) *
    round(eq_others(body_theta,theta_Loong_new_element),3) < 0:
84              panduan = False
85              return (theta_Loong_new_element+theta_Loong[-1])/2
86          else:
87              theta_Loong.append(theta_Loong_new_element)
88
89  # 存储每个时刻，每个把手的角度
90  table1=[]
91  # 存储每个时刻，每个把手在笛卡尔坐标系下的位置
92  table_posxy=[]
93  # 存储每个时刻，每个把手在笛卡尔坐标系下的速度
94  v_pos=[]
95  for time in range(301):
96  # for time in range(442):
97      print(time)
```

```
    solutions_row=[]
    solutionsXY_row=[]

    v_row=[]

    theta_temp = mei_miao_long_tou_de_theta[time]
    # 龙头信息写入
    solutions_row.append(theta_temp)
    solutionsXY_row.append(positionX(theta_temp))
    solutionsXY_row.append(positionY(theta_temp))
    v_row.append(1)

    result_head = root(eq_head, theta_temp-1, args=(theta_temp
, L1))
    solutions_row.append(result_head.x[0])

    # 第一个龙神把手信息写入
    solutionsXY_row.append(positionX(result_head.x[0]))
    solutionsXY_row.append(positionY(result_head.x[0]))
    v_row.append(r_length(solutions_row[1])/r_length(
solutions_row[0]))

    for i in range(222):

        # 现在的把手的位置是solutions_row[i+1]
        # 这个循环开始求解下一个角度
        bbb = longshenliebiaodetheta(solutions_row[i+1])
        # print(bbb)

        # 从第二个龙身开始，信息逐个写入
        solutions_row.append(bbb)
        solutionsXY_row.append(positionX(bbb))
        solutionsXY_row.append(positionY(bbb))

        v_row.append(r_length(solutions_row[i+2])/r_length(
```

```
    solutions_row[0]))
    # print(v_row)
    # print(solutions_row)
    # 逐行写入总表
    table1.append(solutions_row)
    table_posxy.append(solutionsXY_row)
    v_pos.append(v_row)


# 统一保存
# 列表转化到result中要求的格式
table1_final = np.round(np.transpose(table_posxy),6)
v_pos_final = np.round(np.transpose(v_pos),6)
df1 = pd.DataFrame(table1_final)
df2 = pd.DataFrame(v_pos_final)

file1_path = r'D:\Mathematical_Modeling\MCM2024\
    CUMCM2024Problems\A题\Solution\result1逼近位置.xlsx'
df1.to_excel(file1_path, index=False, header=False)
file2_path = r'D:\Mathematical_Modeling\MCM2024\
    CUMCM2024Problems\A题\Solution\result1逼近速度.xlsx'
df2.to_excel(file2_path, index=False, header=False)
df3 = pd.DataFrame(table1)
file3_path = r'D:\Mathematical_Modeling\MCM2024\
    CUMCM2024Problems\A题\Solution\question1角度.xlsx'
df3.to_excel(file3_path, index=False, header=False)
```

程序二：Python 模拟问题二盘入终止时刻

```
import numpy as np
from sympy import symbols, integrate, sqrt
from scipy.optimize import root
import pandas as pd


# 螺距比例
```

```
b = 55/(2*np.pi)

theta = symbols('theta')

# 定义被积函数
f = sqrt(b**2 * theta**2 + b**2)

# 计算积分
Total_Length = integrate(f, (theta, 0, 32*np.pi))

def r_length(theta):
    return 880-b * theta

# 坐标转换方程
def positionX(theta):
    return (b*(32*np.pi-theta)*np.cos(32*np.pi-theta))/100

def positionY(theta):
    return (b*(32*np.pi-theta)*np.sin(32*np.pi-theta))/100

# 两个孔距
L1 = 341-27.5*2
L2 = 220-27.5*2
# 龙头的速度
V0 = 100

def equation(x, V0, t):
    return 0.5 * x * np.sqrt(1 + x**2) + 0.5 * np.log(np.abs(x
    + np.sqrt(1 + x**2))) - (float(Total_Length) - V0 * t) / b


# 龙头对应时间的角度
solutions = []

# 找每个时刻头对应的角度，从里往外
```

```
for t_value in range(442):
# for t_value in range(1510):
    result_head_theta = root(equation, 1, args=(V0, t_value))
    if result_head_theta.success:
        solutions.append(result_head_theta.x[0])
    else:
        solutions.append(None)  # 如果没有找到解，则存储None



theta_rad = 32 * np.pi - np.array(solutions)
mei_miao_long_tou_de_theta=theta_rad.tolist()  #龙头对应时间的角度
# print(mei_miao_long_tou_de_theta)
# x=positionX(theta_rad)
# y=positionY(theta_rad)
# plt.plot(x,y)

# 两点间距离判断公式
def eq_others(x,theta):
    return sqrt(r_length(theta)**2 + r_length(x)**2 - 2 * r_length(theta) * r_length(x) * np.cos(theta - x)) - L2

# 龙头专用距离判断公式，root（）备用
def eq_head(x,theta_temp,L1):
    return np.sqrt(r_length(theta_temp)**2 + r_length(x)**2 - 2 * r_length(theta_temp) * r_length(x) * np.cos(theta_temp - x)) - L1

# 返回某个时刻，某个把手的theta 值
# body_theta是返回的把手的前一个的theta值
def longshenliebiaodetheta(body_theta):
    step=0.001
    panduan=True
    theta_Loong=[]
```

```
    theta_Loong.append(body_theta)


    while panduan:

        theta_Loong_new_element=theta_Loong[-1]-step

        if np.abs(eq_others(body_theta,theta_Loong_new_element
    ))<0.01:
            panduan = False
            return theta_Loong_new_element
        elif round(eq_others(body_theta,theta_Loong[-1]),3) *
    round(eq_others(body_theta,theta_Loong_new_element),3) < 0:
            panduan = False
            return (theta_Loong_new_element+theta_Loong[-1])/2
        else:
            theta_Loong.append(theta_Loong_new_element)

# 存储每个时刻，每个把手的角度
table1=[]
# 存储每个时刻，每个把手在笛卡尔坐标系下的位置
table_posxy=[]
# 存储每个时刻，每个把手在笛卡尔坐标系下的速度
v_pos=[]
for time in range(301):
# for time in range(442):
    print(time)
    solutions_row=[]
    solutionsXY_row=[]

    v_row=[]

    theta_temp = mei_miao_long_tou_de_theta[time]
    # 龙头信息写入
    solutions_row.append(theta_temp)
```

```
    solutionsXY_row.append(positionX(theta_temp))
    solutionsXY_row.append(positionY(theta_temp))
    v_row.append(1)

    result_head = root(eq_head, theta_temp-1, args=(theta_temp, L1))
    solutions_row.append(result_head.x[0])

    # 第一个龙神把手信息写入
    solutionsXY_row.append(positionX(result_head.x[0]))
    solutionsXY_row.append(positionY(result_head.x[0]))
    v_row.append(r_length(solutions_row[1])/r_length(solutions_row[0]))

    for i in range(222):

        # 现在的把手的位置是solutions_row[i+1]
        # 这个循环开始求解下一个角度
        bbb = longshenliebiaodetheta(solutions_row[i+1])
        # print(bbb)

        # 从第二个龙身开始，信息逐个写入
        solutions_row.append(bbb)
        solutionsXY_row.append(positionX(bbb))
        solutionsXY_row.append(positionY(bbb))

        v_row.append(r_length(solutions_row[i+2])/r_length(solutions_row[0]))
    # print(v_row)
    # print(solutions_row)
    # 逐行写入总表
    table1.append(solutions_row)
    table_posxy.append(solutionsXY_row)
    v_pos.append(v_row)

```

```
138
139 # 统一保存
140 # 列表转化到result中要求的格式
141 table1_final = np.round(np.transpose(table_posxy),6)
142 v_pos_final = np.round(np.transpose(v_pos),6)
143 df1 = pd.DataFrame(table1_final)
144 df2 = pd.DataFrame(v_pos_final)
145
146 file1_path = r'D:\Mathematical_Modeling\MCM2024\
       CUMCM2024Problems\A题\Solution\result2逼近位置.xlsx'
147 df1.to_excel(file1_path, index=False, header=False)
148 file2_path = r'D:\Mathematical_Modeling\MCM2024\
       CUMCM2024Problems\A题\Solution\result2逼近速度.xlsx'
149 df2.to_excel(file2_path, index=False, header=False)
150 df3 = pd.DataFrame(table1)
151 file3_path = r'D:\Mathematical_Modeling\MCM2024\
       CUMCM2024Problems\A题\Solution\question2角度.xlsx'
152 df3.to_excel(file3_path, index=False, header=False)
153
154
155 # 统一用 米制
156 # 任意两点的直线方程
157
158 # 奇怪坐标转换方程
159 c = 55/(2*np.pi)
160 def positionX1(theta):
161     return (c*(32*np.pi-theta)*np.cos(32*np.pi-theta))/100
162
163 def positionY1(theta):
164     return (c*(32*np.pi-theta)*np.sin(32*np.pi-theta))/100
165
166 def line_equation_center(theta1,theta2):
167     x1=positionX1(theta1)
168     y1=positionY1(theta1)
169     x2=positionX1(theta2)
```

```
    y2=positionY1(theta2)
   # 孔的直线方程
    k_center=(y2-y1)/(x2-x1)
    b_center=y1-k_center*x1
    print(theta1,theta2,x1,y1,x2,y2,k_center,b_center)
    return k_center,b_center

# 在已知中心直线的情况下，求龙头的靠外的直线方程
def line_equation_out(k,bp):
    b1=bp+0.15*np.sqrt(1+k**2)
    b2=bp-0.15*np.sqrt(1+k**2)

    if b1<b2:
        b_out=b1
    else:
        b_out=b2
    k_out=k
    return k_out,b_out

# 已知龙头的位置，求对应的点到龙身体的距离
def distance(theta_head,k_out,b_out,k,b):
    x0=positionX1(theta_head)
    y0=positionY1(theta_head)
    delta= (2*k*(b_out-y0)-2*x0)**2-4*(k_out**2+1)*(x0**2+y0
   **2-2*y0*b_out+b_out**2-157/1600)
    # print(delta)
    m1=(2*x0-2*k*(b_out-y0)+np.sqrt(delta))/(2*(k_out**2+1))
    m2=(2*x0-2*k*(b_out-y0)-np.sqrt(delta))/(2*(k_out**2+1))
    n1=k_out*m1+b_out
    n2=k_out*m2+b_out


    A = -k
    B = 1
    C = -b
```

```
204
205     dis1=abs(A *m1+ B *n1+ C) / np.sqrt(A**2 + B**2)
206     dis2=abs(A *m2 + B * n2 + C) / np.sqrt(A**2 + B**2)
207     return dis1,dis2
208
209 break_bool=False
210 final_time=0
211 for time in range(390,442):
212     # 这里找到在这个时刻，龙头的相关信息
213
214     k_head_center,b_head_center=line_equation_center(table1[
    time][0],table1[time][1])
215     k_head_out,b_head_out=line_equation_out(k_head_center,
    b_head_center)
216     # print(time)
217     for loong in range(1,16):
218
219     # 龙头的把手是第0个把手，身子的把手是第1个把手
220     # 龙头是loong-1
221         k_body_center,b_body_center=line_equation_center(
    table1[time][loong],table1[time][loong+1])
222         dis1,dis2=distance(table1[time][0],k_head_out,
    b_head_out,k_body_center,b_body_center)
223
224         # 碰撞检测
225         if dis1<=0.15:
226             break_bool=True
227             final_time=time
228             final_boom=loong
229             break
230         if dis2<=0.15:
231             break_bool=True
232             final_time=time
233             final_boom=loong
234             break
```

```
    if break_bool:
        break

# 龙头把手到原点的距离
r=sqrt(positionX1(table1[final_time][0])**2+positionY1(table1[
   final_time][0])**2)

print(final_time,final_boom,r)
```

程序三：Python 模拟问题三最小螺距

```
import numpy as np
from sympy import symbols, integrate, sqrt
from scipy.optimize import root
import pandas as pd


for chouxiang in range(40,80):

    b = chouxiang/(2*np.pi)

    theta = symbols('theta')

    # 定义被积函数
    f = sqrt(b**2 * theta**2 + b**2)

    # 计算积分
    Total_Length = integrate(f, (theta, 0, 32*np.pi))

    def r_length(theta):
        return 880-b * theta

    def positionX(theta):
        return (b*(32*np.pi-theta)*np.cos(32*np.pi-theta))/100

    def positionY(theta):
```

```
        return (b*(32*np.pi-theta)*np.sin(32*np.pi-theta))/100


    L1 = 341-27.5*2
    L2 = 220-27.5*2
    V0 = 100

    def equation(x, V0, t):
        return 0.5 * x * np.sqrt(1 + x**2) + 0.5 * np.log(np.abs(x + np.sqrt(1 + x**2))) - (float(Total_Length) - V0 * t) / b


    # 龙头对应时间的角度
    solutions = []

    # 对每个t值求解方程
    # for t_value in range(301):
    for t_value in range(444):
        result_head_theta = root(equation, 1, args=(V0, t_value))
        if result_head_theta.success:
            solutions.append(result_head_theta.x[0])
        else:
            solutions.append(None)  # 如果没有找到解，则存储None


    theta_rad = 32 * np.pi - np.array(solutions)
    mei_miao_long_tou_de_theta=theta_rad.tolist()  #龙头对应时间的角度
    # print(mei_miao_long_tou_de_theta)
    # x=positionX(theta_rad)
    # y=positionY(theta_rad)
```

```
56      # plt.plot(x,y)
57
58      # 两点间距离判断公式
59      def eq_others(x,theta):
60          return sqrt(r_length(theta)**2 + r_length(x)**2 - 2 *
    r_length(theta) * r_length(x) * np.cos(theta - x)) - L2
61
62
63      def eq_head(x,theta_temp,L1):
64          return np.sqrt(r_length(theta_temp)**2 + r_length(x)
    **2 - 2 * r_length(theta_temp) * r_length(x) * np.cos(
    theta_temp - x)) - L1
65
66      # 返回某个时刻，某个把手的theta 值
67      # body_theta是返回的把手的前一个的theta值
68      def longshenliebiaodetheta(body_theta):
69          step=0.01
70          panduan=True
71          theta_Loong=[]
72          theta_Loong.append(body_theta)
73
74
75          while panduan:
76
77              theta_Loong_new_element=theta_Loong[-1]-step
78
79              if np.abs(eq_others(body_theta,
    theta_Loong_new_element))<0.01:
80                  panduan = False
81                  return theta_Loong_new_element
82              elif round(eq_others(body_theta,theta_Loong[-1])
    ,3) * round(eq_others(body_theta,theta_Loong_new_element),3)
     < 0:
83                  panduan = False
84                  return (theta_Loong_new_element+theta_Loong
```

```
[-1])/2
85              else:
86                  theta_Loong.append(theta_Loong_new_element)
87
88
89
90      table1=[]
91      table_posxy=[]
92      v_pos=[]
93      # for time in range(301):
94      for time in range(442):
95          # print(time)
96          solutions_row=[]
97          solutionsXY_row=[]
98
99          v_row=[]
100
101         theta_temp = mei_miao_long_tou_de_theta[time]
102
103         solutions_row.append(theta_temp)
104         solutionsXY_row.append(positionX(theta_temp))
105         solutionsXY_row.append(positionY(theta_temp))
106         v_row.append(1)
107
108         result_head = root(eq_head, theta_temp-1, args=(
    theta_temp, L1))
109         solutions_row.append(result_head.x[0])
110
111
112         solutionsXY_row.append(positionX(result_head.x[0]))
113         solutionsXY_row.append(positionY(result_head.x[0]))
114         v_row.append(r_length(solutions_row[1])/r_length(
    solutions_row[0]))
115
116         for i in range(18):
```

```
117
118             # 现在的把手的位置是solutions_row[i+1]
119             # 这个循环开始求解下一个角度
120             bbb = longshenliebiaodetheta(solutions_row[i+1])
121             # print(bbb)
122             solutions_row.append(bbb)
123             solutionsXY_row.append(positionX(bbb))
124             solutionsXY_row.append(positionY(bbb))
125
126             v_row.append(r_length(solutions_row[i+2])/r_length
    (solutions_row[0]))
127         # print(v_row)
128         # print(solutions_row)
129         table1.append(solutions_row)
130         table_posxy.append(solutionsXY_row)
131         v_pos.append(v_row)
132
133
134     # 统一用 米制
135     # 任意两点的直线方程
136     c = chouxiang/(2*np.pi)
137     def positionX1(theta):
138         return (c*(32*np.pi-theta)*np.cos(32*np.pi-theta))/100
139
140     def positionY1(theta):
141         return (c*(32*np.pi-theta)*np.sin(32*np.pi-theta))/100
142
143     def line_equation_center(theta1,theta2):
144         x1=positionX1(theta1)
145         y1=positionY1(theta1)
146         x2=positionX1(theta2)
147         y2=positionY1(theta2)
148     # 孔的直线方程
149         k_center=(y2-y1)/(x2-x1)
150         b_center=y1-k_center*x1
```

```
        # print(theta1,theta2,x1,y1,x2,y2,k_center,b_center)
        return k_center,b_center

    # 在已知中心直线的情况下，求龙头的靠外的直线方程
    def line_equation_out(k,bp):
        b1=bp+0.15*np.sqrt(1+k**2)
        b2=bp-0.15*np.sqrt(1+k**2)

        if b1<b2:
            b_out=b1
        else:
            b_out=b2
        k_out=k
        return k_out,b_out

    def distance(theta_head,k_out,b_out,k,b):
        x0=positionX1(theta_head)
        y0=positionY1(theta_head)
        delta= (2*k*(b_out-y0)-2*x0)**2-4*(k_out**2+1)*(x0**2+
    y0**2-2*y0*b_out+b_out**2-157/1600)
        # print(delta)
        m1=(2*x0-2*k*(b_out-y0)+np.sqrt(delta))/(2*(k_out
    **2+1))
        m2=(2*x0-2*k*(b_out-y0)-np.sqrt(delta))/(2*(k_out
    **2+1))
        n1=k_out*m1+b_out
        n2=k_out*m2+b_out


        A = -k
        B = 1
        C = -b

        dis1=abs(A *m1+ B *n1+ C) / np.sqrt(A**2 + B**2)
        dis2=abs(A *m2 + B * n2 + C) / np.sqrt(A**2 + B**2)
```

```
        return dis1,dis2

 break_bool=False
 final_time=0
 for time in range(380,442):
     # 这里找到在这个时刻，龙头的相关信息

     k_head_center,b_head_center=line_equation_center(table1[time][0],table1[time][1])
     k_head_out,b_head_out=line_equation_out(k_head_center,b_head_center)
     # print(time)
     for loong in range(1,16):

     # 龙头的把手是第0个把手，身子的把手是第1个把手
     # 龙头是loong-1
         k_body_center,b_body_center=line_equation_center(table1[time][loong],table1[time][loong+1])
         dis1,dis2=distance(table1[time][0],k_head_out,b_head_out,k_body_center,b_body_center)

         if dis1<=0.15:
             break_bool=True
             final_time=time
             final_boom=loong
             break
         if dis2<=0.15:
             break_bool=True
             final_time=time
             final_boom=loong
             break
     if break_bool:
         break
 r=sqrt(positionX1(table1[final_time][0])**2+positionY1(table1[final_time][0])**2)
```

```
213
214     print(final_time,final_boom,r,chouxiang)
```

程序四：Python 模拟问题四掉头路径

```
1  import numpy as np
2  from sympy import symbols, integrate, sqrt
3  from scipy.optimize import root
4  import pandas as pd
5
6
7  b = 170/(2*np.pi)
8
9  theta = symbols('theta')
10
11 # 定义被积函数
12 f = sqrt(b**2 * theta**2 + b**2)
13
14 # 计算积分
15 Total_Length = integrate(f, (theta, 0, 32*np.pi))
16
17 def r_length(theta):
18     return 880-b * theta
19
20 def positionX(theta):
21     return (b*(32*np.pi-theta)*np.cos(32*np.pi-theta))/100
22
23 def positionY(theta):
24     return (b*(32*np.pi-theta)*np.sin(32*np.pi-theta))/100
25
26
27 L1 = 341-27.5*2
28 L2 = 220-27.5*2
29 V0 = 100
30
31 def equation(x, V0, t):
```

```
    return 0.5 * x * np.sqrt(1 + x**2) + 0.5 * np.log(np.abs(x + np.sqrt(1 + x**2))) - (float(Total_Length) - V0 * t) / b


# 龙头对应时间的角度
solutions = []

# 对每个t值求解方程
# for t_value in range(301):
for t_value in range(444):
    result_head_theta = root(equation, 1, args=(V0, t_value))
    if result_head_theta.success:
        solutions.append(result_head_theta.x[0])
    else:
        solutions.append(None)  # 如果没有找到解，则存储None



theta_rad = 32 * np.pi - np.array(solutions)
mei_miao_long_tou_de_theta=theta_rad.tolist()  #龙头对应时间的角度
# print(mei_miao_long_tou_de_theta)
# x=positionX(theta_rad)
# y=positionY(theta_rad)
# plt.plot(x,y)

# 两点间距离判断公式
def eq_others(x,theta):
    return sqrt(r_length(theta)**2 + r_length(x)**2 - 2 * r_length(theta) * r_length(x) * np.cos(theta - x)) - L2


def eq_head(x,theta_temp,L1):
    return np.sqrt(r_length(theta_temp)**2 + r_length(x)**2 - 2 * r_length(theta_temp) * r_length(x) * np.cos(theta_temp -
```

```
    x)) - L1
63
64  # 返回某个时刻，某个把手的theta 值
65  # body_theta是返回的把手的前一个的theta值
66  def longshenliebiaodetheta(body_theta):
67      step=0.01
68      panduan=True
69      theta_Loong=[]
70      theta_Loong.append(body_theta)
71
72
73      while panduan:
74
75          theta_Loong_new_element=theta_Loong[-1]-step
76
77          if np.abs(eq_others(body_theta,theta_Loong_new_element
    ))<0.01:
78              panduan = False
79              return theta_Loong_new_element
80          elif round(eq_others(body_theta,theta_Loong[-1]),3) *
    round(eq_others(body_theta,theta_Loong_new_element),3) < 0:
81              panduan = False
82              return (theta_Loong_new_element+theta_Loong[-1])/2
83          else:
84              theta_Loong.append(theta_Loong_new_element)
85
86
87
88  table1=[]
89  table_posxy=[]
90  v_pos=[]
91  # for time in range(301):
92  for time in range(442):
93      # print(time)
94      solutions_row=[]
```

```
    solutionsXY_row=[]

    v_row=[]

    theta_temp = mei_miao_long_tou_de_theta[time]

    solutions_row.append(theta_temp)
    solutionsXY_row.append(positionX(theta_temp))
    solutionsXY_row.append(positionY(theta_temp))
    v_row.append(1)

    result_head = root(eq_head, theta_temp-1, args=(theta_temp
, L1))
    solutions_row.append(result_head.x[0])


    solutionsXY_row.append(positionX(result_head.x[0]))
    solutionsXY_row.append(positionY(result_head.x[0]))
    v_row.append(r_length(solutions_row[1])/r_length(
solutions_row[0]))

    for i in range(18):

        # 现在的把手的位置是solutions_row[i+1]
        # 这个循环开始求解下一个角度
        bbb = longshenliebiaodetheta(solutions_row[i+1])
        # print(bbb)
        solutions_row.append(bbb)
        solutionsXY_row.append(positionX(bbb))
        solutionsXY_row.append(positionY(bbb))

        v_row.append(r_length(solutions_row[i+2])/r_length(
solutions_row[0]))
    # print(v_row)
    # print(solutions_row)
```

```
    table1.append(solutions_row)
    table_posxy.append(solutionsXY_row)
    v_pos.append(v_row)


# 统一用 米制
# 任意两点的直线方程
c = 170/(2*np.pi)
def positionX1(theta):
    return (c*(32*np.pi-theta)*np.cos(32*np.pi-theta))/100

def positionY1(theta):
    return (c*(32*np.pi-theta)*np.sin(32*np.pi-theta))/100

def line_equation_center(theta1,theta2):
    x1=positionX1(theta1)
    y1=positionY1(theta1)
    x2=positionX1(theta2)
    y2=positionY1(theta2)
# 孔的直线方程
    k_center=(y2-y1)/(x2-x1)
    b_center=y1-k_center*x1
    # print(theta1,theta2,x1,y1,x2,y2,k_center,b_center)
    return k_center,b_center

# 在已知中心直线的情况下，求龙头的靠外的直线方程
def line_equation_out(k,bp):
    b1=bp+0.15*np.sqrt(1+k**2)
    b2=bp-0.15*np.sqrt(1+k**2)

    if b1<b2:
        b_out=b1
    else:
        b_out=b2
    k_out=k
```

```
    return k_out,b_out

def distance(theta_head,k_out,b_out,k,b):
    x0=positionX1(theta_head)
    y0=positionY1(theta_head)
    delta= (2*k*(b_out-y0)-2*x0)**2-4*(k_out**2+1)*(x0**2+y0**2-2*y0*b_out+b_out**2-157/1600)
    # print(delta)
    m1=(2*x0-2*k*(b_out-y0)+np.sqrt(delta))/(2*(k_out**2+1))
    m2=(2*x0-2*k*(b_out-y0)-np.sqrt(delta))/(2*(k_out**2+1))
    n1=k_out*m1+b_out
    n2=k_out*m2+b_out


    A = -k
    B = 1
    C = -b

    dis1=abs(A *m1+ B *n1+ C) / np.sqrt(A**2 + B**2)
    dis2=abs(A *m2 + B * n2 + C) / np.sqrt(A**2 + B**2)
    return dis1,dis2

break_bool=False
final_time=0
for time in range(380,442):
    # 这里找到在这个时刻，龙头的相关信息

    k_head_center,b_head_center=line_equation_center(table1[time][0],table1[time][1])
    k_head_out,b_head_out=line_equation_out(k_head_center,b_head_center)
    # print(time)
    for loong in range(1,16):

    # 龙头的把手是第0个把手，身子的把手是第1个把手
```

```
194     # 龙头是loong-1
195         k_body_center,b_body_center=line_equation_center(
    table1[time][loong],table1[time][loong+1])
196         dis1,dis2=distance(table1[time][0],k_head_out,
    b_head_out,k_body_center,b_body_center)
197 
198         if dis1<=0.15:
199             break_bool=True
200             final_time=time
201             final_boom=loong
202             break
203         if dis2<=0.15:
204             break_bool=True
205             final_time=time
206             final_boom=loong
207             break
208     if break_bool:
209         break
210 r=sqrt(positionX1(table1[final_time][0])**2+positionY1(table1[
    final_time][0])**2)
211 
212 print(final_time,final_boom,r)
```