“板凳龙”行进过程的建模分析

摘要

“板凳龙”由多条板凳通过把手首尾相连形成，盘龙时舞龙队沿等距螺线盘入盘出、沿圆弧进行调头。本文研究了舞龙队在不同路径下行进时，相邻把手中心点之间的位置和速度关系以及板凳之间发生碰撞的临界状态，同时在不影响舞龙队自如盘入盘出的前提下，对板凳龙的行进的路径和速度进行规划，减小盘龙所需面积、提高行进速度，增加了板凳龙的观赏性。

针对问题一，我们利用**第一类曲线积分**、余弦定理等数学方法，建立了龙头前把手中心点的定位模型、相邻两把手中心点的定位模型以及相邻两把手中心点的关联速度模型，求解得到 0 到 $300s$ 内整个舞龙队的位置和速度。通过对问题一求解结果的分析，我们发现在舞龙队沿螺线盘入 $300s$ 后，**最终仍有 43 个把手中心点位于第 16 圈螺线外**，第 180 节龙身位于边界处，它的前把手中心点位于螺线内，后把手中心点位于螺线外。

针对问题二，根据**贪心思想**，我们将判断板凳之间是否相撞的问题，转化为判断龙头的板凳是否与它外侧相邻的板凳发生碰撞的问题。通过平面解析几何的分析，确定了板凳之间发生碰撞的临界时刻 **412.180374s**，此时龙头板凳外侧的边角点恰好未与其外侧第 8 节、第 10 节龙身板凳发生碰撞，利用问题一建立的模型，给出此刻整个舞龙队的位置和速度。

针对问题三，我们首先通过联立螺线和圆的极坐标方程，确定不同螺距的盘入螺线与调头区域边界的交点；然后运用问题二中判断板凳之间是否相撞的方法，得到当舞龙队沿不同螺距的螺线盘入时，板凳之间发生碰撞的临界位置；最后通过比较交点与临界点的位置关系，判断该螺距是否符合题意。我们使用**二分法**对最小螺距进行查找，确定了满足约束条件的**最小螺距为 0.452209m**。

针对问题四，由中心对称的性质，盘出螺线绕中心旋转 π 可与盘入螺线重合，首先证明在保持各部分相切关系时，**调整圆弧不会改变调头路径的长度**；然后我们考虑了把手中心点位于不同位置时的情况，并给出了判断条件，建立龙头前把手中心点的定位模型、相邻两把手中心点的定位模型以及相邻两把手中心点的关联速度模型，求解得到调头前和调头后每秒舞龙队的位置和速度。

针对问题五，当龙头前把手中心点的行进速度不变时，**各把手中心点的速度只与此时整个舞龙队的位置有关**，所以我们只需要确定龙头前把手中心点位于哪个位置时会出现相对龙头速度的最大比值，可以使用**粒子群算法**搜索全局最优解。

关键字： 等距螺线　第一类曲线积分　贪心思想　二分查找　粒子群算法

一、 问题重述

1.1 背景资料

板凳龙是一项迎神祈福的民俗活动，蕴含着深厚的地域文化。村民们将每户一节的板凳钻孔连接，组成长龙，有时长度可达两百多米。在表演过程中，龙头在前领头，龙身和龙尾相随盘旋，整体成圆盘状。舞龙队如果能自如地盘入和盘出，并在较小面积内迅速行进，则观赏效果最佳。

1.2 需要解决的问题

问题一：各节板凳上把手中心点沿着螺距为 $55cm$ 的等距螺线顺时针盘入，龙头前把手中心点的行进速度保持 $1m/s$ 不变。初始时刻，龙头前把手的中心点位于螺线第 16 圈 A 点处。要求给出从初始时刻到 $300s$ 为止，每个时刻龙头、龙身和龙尾各前把手及龙尾后把手中心的位置和速度。

问题二：舞龙队沿问题 1 设定的螺线盘入，确定板凳之间发生碰撞的临界时刻，给出此时龙头、龙身和龙尾各前把手及龙尾后把手中心的位置和速度。

问题三：经过调头的过程，舞龙队由顺时针盘入调头切换为逆时针盘出，调头空间是以螺线中心为圆心、直径为 $9m$ 的圆形区域。要求确定使得龙头前把手能够沿着相应螺线盘入到调头空间的边界，而不发生碰撞的最小螺距。

问题四：盘出螺线与盘入螺线关于螺线中心呈中心对称，它们的螺距均为 $1.7m$，舞龙队在问题 3 设定的调头空间内完成调头，调头路径是由两段圆弧相切连接形成的 S 形曲线，判断能不能在保持各部分相切时，调整圆弧的形状，使调头路径变短；然后在龙头前把手的行进速度为 $1m/s$ 时，给出从 $-100s$ 到 $100s$ 的时间范围内，每个时刻舞龙队的位置和速度。

问题五：当舞龙队沿问题 4 设定的路径行进时，龙头前把手的行进速度保持不变，要求确定龙头的最大行进速度，使得在盘入、调头和盘出过程中，舞龙队各把手的速度均不超过 $2m/s$。

二、 问题分析

2.1 问题一分析

问题一要求我们求解沿等距螺线盘入时，各时刻舞龙队的位置和速度。我们将建模过程分为三步：求解各个时刻龙头前把手中心点的位置；求解各个时刻整个舞龙队的位置；求解各个时刻整个舞龙队的速度。

2.2 问题二分析

问题二要求我们确定舞龙队沿问题一中路径盘入过程中，板凳发生碰撞的临界状态。我们将其转化为判断龙头的板凳是否与它外侧相邻的板凳发生碰撞的问题，进一步转化为判断龙头板凳的外侧两个边角点和与其相邻的两条板凳的内侧边界线的问题。通过求解边角点的坐标，边界线的直线方程，判断点线关系来求解碰撞的临界时刻。

2.3 问题三分析

问题三要求我们设计盘入螺线的最小螺距，使得龙头前把手中心点能够到达圆形调头空间的边界，而不与其他板凳发生碰撞。运用问题二中判断板凳是否相撞的方法，确定龙头与其外侧板凳相撞的临界位置，然后使用二分法查找满足约束条件的最小螺距。

2.4 问题四分析

问题四要求我们在盘出螺线与盘入螺线关于螺线中心呈中心对称的条件下，研究舞龙队按照规定路径完成调头时，每个时刻各把手中心点的位置和速度。我们首先判断能否调整圆弧，仍保持调头路径的两端圆弧相切、盘入螺线与盘出螺线分别与两段圆弧相切，使得调头路径变短。

2.5 问题五分析

问题五要求我们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$2m/s$。我们发现，当龙头前把手中心点的行进速度不变时，各把手中心点的速度至于此时整个舞龙队的位置有关，所以我们只需要确定龙头前把手中心点位于哪个位置时会出现相对龙头速度的最大速度，可以使用粒子群算法求解该全局最大值。

三、 模型的假设

- 假设两节板凳通过把手的连接部分无摩擦阻力。
- 假设各板凳的长度信息准确无误。
- 假设盘入、盘出轨迹为标准等距螺线。
- 假设在各把手中心点在第 16 圈螺线的 A 点外也位于螺线上。
- 假设板凳为形状、体积不发生变化的刚体。
- 假设板凳的形状为标准矩形。
- 假设螺线与圆弧相切部分过渡平滑。

四、 符号说明

符号	含义	单位
p	等距螺线的螺距	米
v_i	把手中心点 i 的速度	米 / 秒
L_i	相邻两把手中心点之间的距离	米
k_i	点 P_i 处螺线的切线的斜率	/
m_i	点 P_i、P_{i+1} 连线的斜率	/
α_i	点 P_i 处速度方向与 P_iP_{i+1} 的夹角	角度
θ_c	龙头与其外侧板凳相撞的临界位置的极角	角度
s	调头路径的长度	米

五、 问题一模型的建立与求解

5.1 问题一的模型准备

本问题中舞龙队的行进轨迹为螺距为 $55cm$ 的等距螺线，首先写出等距螺线的极坐标方程。

5.1.1 等距螺线的极坐标方程

以螺线中心 O 为坐标原点，原点与起始点 A 之间的连线方向为极轴，建立极坐标系，等距螺线上的径向距离 r 随角度 θ 线性增加，其极坐标方程为

$$r = a + b\theta$$

其中，r 是点到原点的径向距离；θ 是极角；a 是初始半径，表示当 $\theta = 0$ 时，螺线的起点到原点的距离；b 是螺距系数，决定每转一圈（即 θ 增加 2π）螺线径向距离的增量。

在本问题中，螺线从原点开始，即 $a = 0$；设螺距为 p，螺距系数 $b = \frac{p}{2\pi}$，**螺线的极坐标方程**为

$$r = \frac{p}{2\pi}\theta \tag{1}$$

其中，螺距 $p = 0.55m$

舞龙队沿螺线顺时针盘入时，各把手中心均位于螺线上，**初始时，我们认为龙头前把手的中心点位于螺线第 16 圈 A 点处**。已知龙头前把手的行进速度始终保持 v_0 不变，要想求解它行进过程中的位置，我们需要建立起螺线上两点之间的螺线长度与两点的极坐标之间的关系。

5.1.2 等距螺线的长度计算

等距螺线的长度计算可以通过**第一类曲线积分**求得。对于给定的等距螺线方程 $r = a + b\theta$，螺线的长度 l 从 $\theta = \theta_1$ 到 $\theta = \theta_2$ 的计算公式为

$$l = \int_{\theta_1}^{\theta_2} \sqrt{\left(\frac{dr}{d\theta}\right)^2 + r^2} d\theta \tag{2}$$

在本问题中，$r = \frac{p}{2\pi}\theta$，求导得 $\frac{dr}{d\theta} = \frac{p}{2\pi}$，代入式（2）得到螺线上两点之间的螺线长度与两点的极角之间的关系式

$$l = \frac{p}{2\pi} \int_{\theta_1}^{\theta_2} \sqrt{\theta^2 + 1} d\theta \tag{3}$$

由于本题中为舞龙队为顺时针盘入，把手中心点的极角不断减小，因为螺线长度 l 一定是正值，我们规定 θ_2 为某一次位移起点的极角、θ_1 为某一次位移终点的极角。

5.2 问题一的模型建立

在本问题中，舞龙队沿螺线 $r = \frac{p}{2\pi}\theta$ 盘入时，龙头前把手的中心始终位于螺线上且行进速度保持不变，龙身和龙尾各前把手及龙尾后把手中心的位置和速度均由龙头前把手中心决定。我们首先通过研究不同时刻龙头前把手中心的走过的螺线长度，确定不同时刻龙头前把手中心的位置。

5.2.1 龙头前把手中心点的定位模型

我们设某时刻 t 龙头前把手中心的坐标为 $P_0\,(r_0, \theta_0)$，已知龙头前把手中心的行进速度 v_0，则经过 T 它走过的螺线长度 $l = v_0 T$，设 $t + T$ 时刻龙头前把手中心的坐标为 $P_0'\,(r_0', \theta_0')$。

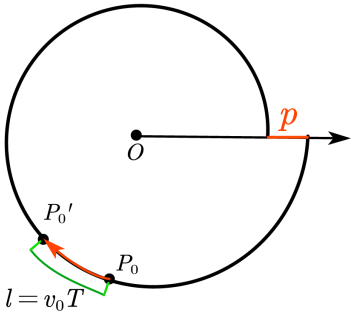

图 **1**　每秒龙头前把手中心点的位移示意图

由于龙头前把手中心的移动方向为顺时针方向，因此将 $l = v_0 T$、$\theta_2 = \theta_0$、$\theta_1 = \theta_0'$ 代入式（3），得到**龙头前把手中心的定位模型**

$$\frac{p}{2\pi}\int_{\theta_0'}^{\theta_0}\sqrt{\theta^2+1}d\theta = v_0 T \tag{4}$$

其中，p 为螺距；v_0 为龙头前把手中心点的行进速度；T 为我们设定的时间间隔。

初始时刻 $t=0$ 时，龙头前把手中心点所在位置的极角为 32π，由式（4）可递推得到每秒龙头前把手中心点所在位置的坐标。

5.2.2 相邻两把手中心点的定位模型

由每个时刻龙头前把手中心点所在的位置，可以确定此时整个舞龙队的位置。设相邻两把手中心点 P_i、P_{i+1} 之间的距离为 L_i，由题意知，龙头前把手中心点 P_0 与第一节龙头 P_1 之间的距离 $L_0=2.86m$，其余龙身和龙尾相邻两把手中心点之间的距离 $L_i(i\in\{1,2,3...223\})$ 均为 $1.65m$。

接下来我们研究相邻两个把手中心点之间的位置关系，设某时刻前一个把手中心点的坐标为 $P_i\,(r_i,\theta_i)$，后一个把手中心点的坐标为 $P_{i+1}\,(r_{i+1},\theta_{i+1})$。

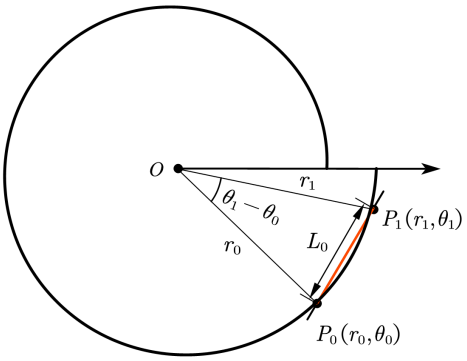

图 **2**　P_0 与 P_1 的位置示意图

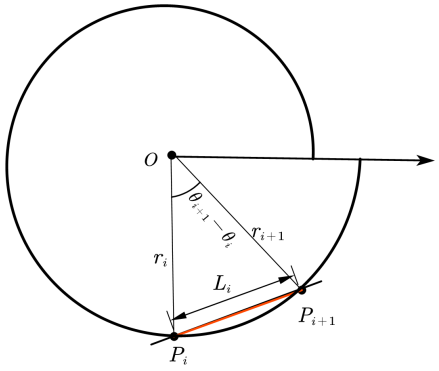

图 **3**　P_i 与 P_{i+1} 的位置示意图

在三角形 OP_iP_{i+1} 中使用**余弦定理**，建立起**相邻两个把手中心点的定位模型**

$$L_i{}^2 = r_i{}^2 + r_{i+1}{}^2 - 2r_i r_{i+1}\cos(\theta_{i+1}-\theta_i) \tag{5}$$

其中，

$$L_i = \begin{cases} 2.86m & i=0 \\ 1.65m & i\in\{1,2,3...223\} \end{cases}$$

由 5.2.1 已经得到了各个时刻龙头前把手中心点所在位置 $P_0\,(r_0,\theta_0)$，将 $i=0$ 代入式（5）可以解得各个时刻第一节龙身前把手中心点的位置，由此可递推得到各个时刻整个舞龙队的位置。

5.2.3 相邻两把手中心点的关联速度模型

对于某一节板凳来说，在盘入过程中可将其视为形状、体积不发生变化的刚体。利用**刚体的不可压缩性**，我们可以得到此板凳上两个把手中心点的速度关系。

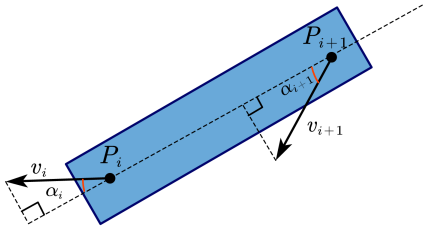

图 **4**　板凳两端的速度关系示意图

刚体的不可压缩性从物理上可以解释为一个刚体在运动过程中，其所有长度（尤其是端点之间的距离）不发生变化。因此，刚体的两端沿其长度方向的速度分量必须一致。对于两个相邻把手中心点 P_i、P_{i+1} 来说，它们的速度 v_i、v_{i+1} 在 P_iP_{i+1} 方向上的投影速度相等。

为了确定速度方向的投影夹角 α_i、α_{i+1}，首先我们需要求解点 P_i、P_{i+1} 处的速度方向。由**速度切线定理**可知，沿着曲线运动的点在某一处的速度方向为该曲线的切线方向。对于螺线 $r=\frac{p}{2\pi}\theta$，其上某一点 (r,θ) 的直角坐标可表示为

$$\begin{cases} x=\dfrac{p}{2\pi}\theta\cos\theta \\ y=\dfrac{p}{2\pi}\theta\sin\theta \end{cases}$$

利用微分学的方法，求解该点处 **x、y 坐标关于 θ 的导数**

$$\begin{cases} \dfrac{dx}{d\theta}=\dfrac{p}{2\pi}(\cos\theta-\theta\sin\theta) \\ \dfrac{dy}{d\theta}=\dfrac{p}{2\pi}(\sin\theta+\theta\cos\theta) \end{cases} \tag{6}$$

我们知道曲线上某点处的切线斜率为 $k=\frac{dy}{dx}$，得到螺线上某点 (r,θ) 处的切线方向

$$k=\frac{\sin\theta+\theta\cos\theta}{\cos\theta-\theta\sin\theta} \tag{7}$$

对于我们研究的两个相邻把手中心点 P_i、P_{i+1} 来说，该点处的速度方向已经得到，接下来我们研究这两点的连线 P_iP_{i+1} 的方向。根据两点之间连线的斜率定义式 $k=\frac{y_2-y_1}{x_2-x_1}$，我们得到了 P_iP_{i+1} 的斜率 m_i 的表达式为

$$m_i=\frac{r_{i+1}\sin\theta_{i+1}-r_i\sin\theta_i}{r_{i+1}\cos\theta_{i+1}-r_i\cos\theta_i} \tag{8}$$

接下来我们根据正切公式，确定速度方向的投影夹角 α_i、α_{i+1} 如式（10）、（11）所示。利用刚体的不可压缩性，我们可以得到两个把手中心点 P_i、P_{i+1} 的**关联速度模型**

$$v_{i+1} = \frac{v_i \cos \alpha_i}{\cos \alpha_{i+1}} \tag{9}$$

其中，

$$\alpha_i = \arctan \left| \frac{k_i - m_i}{1 + k_i m_i} \right| \tag{10}$$

$$\alpha_{i+1} = \arctan \left| \frac{k_{i+1} - m_i}{1 + k_{i+1} m_i} \right| \tag{11}$$

$$k_i = \frac{\sin \theta_i + \theta_i \cos \theta_i}{\cos \theta_i - \theta_i \sin \theta_i} \tag{12}$$

$$k_{i+1} = \frac{\sin \theta_{i+1} + \theta_{i+1} \cos \theta_{i+1}}{\cos \theta_{i+1} - \theta_{i+1} \sin \theta_{i+1}} \tag{13}$$

5.3 问题一的模型求解

在本问题中，螺距为 $0.55m$，则螺距系数表示为 $b = \frac{0.55}{2\pi}$，螺线的极坐标方程为

$$r = \frac{0.55}{2\pi}\theta \tag{14}$$

在本问题中，初始时刻我们认为龙头前把手中心点位于螺线第 16 圈 A 点处，其余各把手中心点位于外圈的螺线上（如图 5 中虚线所示）。在各点沿螺线 $r = \frac{0.55}{2\pi}\theta$ 盘入时，各个时刻整个舞龙队的位置和速度由初始时刻龙头前把手中心点的位置和速度决定。

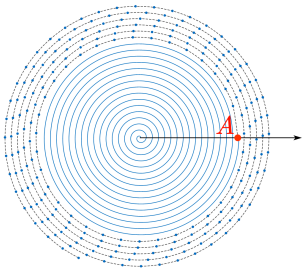

图 **5**　初始时刻各把手中心点位置示意图

问题一中龙头的行进速度 $v_0 = 1m/s$，经过 $1s$ 它走过的螺线长度 $l = 1m$，$t = 0$ 时刻龙头前把手中心点的极角为 32π，式（4）转化为

$$\frac{0.55}{2\pi} \int_{\theta_0'}^{\theta_0} \sqrt{\theta^2 + 1} d\theta = 1 \tag{15}$$

其中，$\theta_0 = 32\pi$。

由式（15）得到 $t=1s$ 时龙头前把手中心点的极角 $\theta_0'=31.963809\pi$，递推得到每秒龙头前把手中心点的坐标。然后我们将每个时刻龙头前把手中心点的坐标，代入相邻两把手中心点的定位模型，计算此时刻整个舞龙队的位置，部分位置结果如表 1 所示。

表 **1**　问题一位置结果示例

	0 s	60 s	120 s	180 s	240 s	300 s
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	6.094780	-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)	-9.518732	-8.686317	-5.543150	2.890455	5.980011	-6.301346
第 51 节龙身 y(m)	1.341137	2.540108	6.377946	7.249289	-3.827758	0.465829
第 101 节龙身 x(m)	2.913983	5.687116	5.361939	1.898794	-4.917371	-6.237722
第 101 节龙身 y(m)	-9.918311	-8.001384	-7.557638	-8.471614	-6.379874	3.936008
第 151 节龙身 x(m)	10.861726	6.682311	2.388757	1.005154	2.965378	7.040740
第 151 节龙身 y(m)	1.828753	8.134544	9.727411	9.424751	8.399721	4.393013
第 201 节龙身 x(m)	4.555102	-6.619664	-10.627211	-9.287720	-7.457151	-7.458662
第 201 节龙身 y(m)	10.725118	9.025570	1.359847	-4.246673	-6.180726	-5.263384
龙尾（后） x(m)	-5.305444	7.364557	10.974348	7.383896	3.241051	1.785033
龙尾（后） y(m)	-10.676584	-8.797992	0.843473	7.492370	9.469336	9.301164

在本问题中，初始时刻，所有把手中心点均位于第 16 圈螺线外，舞龙队沿螺线

$r = \frac{0.55}{2\pi}\theta$ 顺时针盘入 $300s$ 后，**最终仍有 43 个把手中心点位于第 16 圈螺线外**，第 180 节龙身位于边界处，它的前把手中心点位于螺线内，后把手中心点位于螺线外。

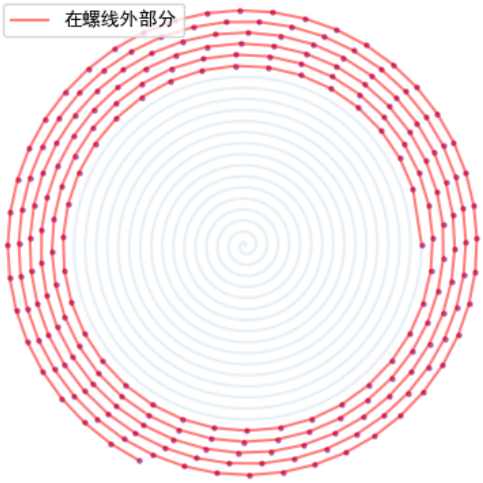

图 **6**　$0s$ 时各把手中心点的位置

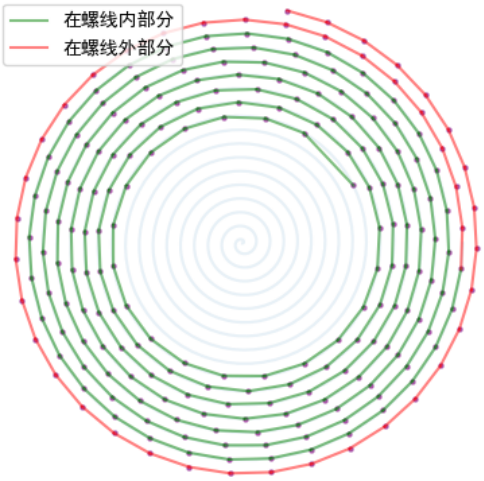

图 **7**　$300s$ 时各把手中心点的位置

最后我们通过各个时刻把手中心点的位置与龙头前把手中心点的速度 $v_0 = 1m/s$，求解各个把手中心点的速度，从而得到每个时刻整个舞龙队的速度，部分结果如表 2 所示。

表 **2**　问题一速度结果示例

	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)	0.999971	0.999961	0.999945	0.999917	0.999859	0.999709
第 51 节龙身 (m/s)	0.999742	0.999662	0.999538	0.999331	0.998941	0.998065
第 101 节龙身 (m/s)	0.999575	0.999453	0.999269	0.998971	0.998435	0.997302
第 201 节龙身 (m/s)	0.999348	0.999180	0.998935	0.998551	0.997894	0.996574
龙尾（后）(m/s)	0.999311	0.999136	0.998883	0.998489	0.997816	0.996478

通过分析求解的速度结果，我们发现在盘入时，离龙头越远的把手中心点的速度越小。同一位置的把手中心点的位置在不同时刻的速度也不同，对于某一个把手中心点来说，离螺线中心越近速度越小。整体上来看，各个把手中心点在不同时刻的速度均小于龙头前把手中心点的速度，但这个差值的范围不超过 0.4%。

六、 问题二模型的建立与求解

6.1 问题二的模型建立

在本问题中，舞龙队沿着问题一中的螺线 $r = \frac{0.55}{2\pi}\theta$ 顺时针盘入。要求通过判断板凳之间发生碰撞的临界状态，确定舞龙队不能再继续盘入的时间。由于龙身和龙尾的运动轨迹均是对龙头的重复，且龙头的板长要大于龙身和龙尾的板长，根据**贪心思想**，我们将判断板凳之间是否相撞的问题，转化为**判断龙头的板凳是否与它外侧相邻的板凳发生碰撞**的问题。通过几何分析，我们发现龙头的板凳与它外侧的板凳存在两种位置关系（如图 8 所示）。设龙头的板凳外侧的两个边角点为 A、B，与其外侧相邻的两节板凳设为 a、b，板凳 a、b 内侧的边界线设为 $y_a = m_a x + n_a$、$y_b = m_b x + n_b$。

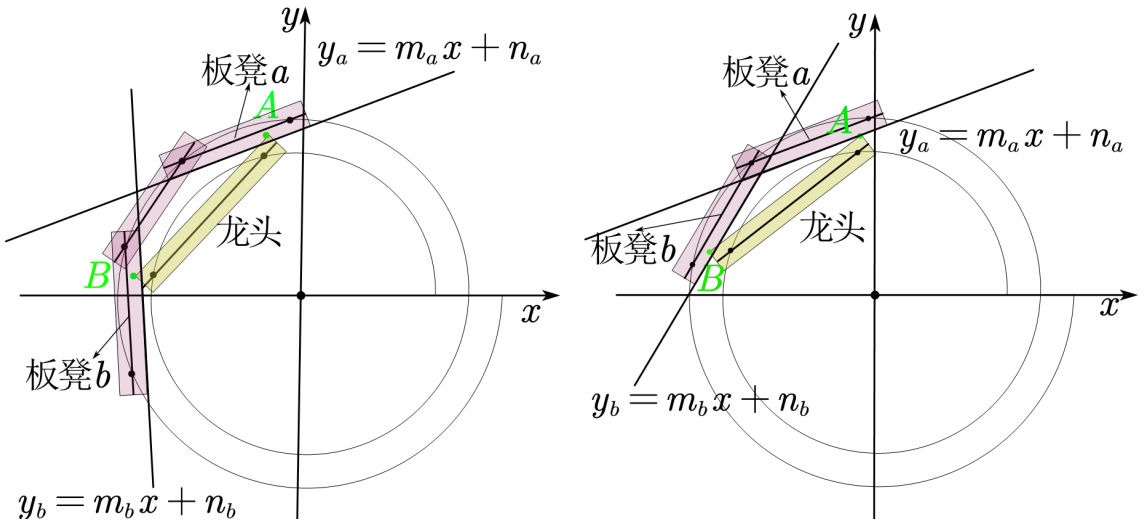

图 **8**　龙头所在板凳与其外侧相邻板凳的位置关系示意图

我们将判断龙头的板凳是否与它外侧相邻的板凳相撞的问题，转化为判断边角点 A 与边界线 y_a、边角点 B 与边界线 y_b 之间的位置关系的问题，再将这个问题划分为三个子问题：计算边角点 A、B 的坐标，求解边界线 y_a、y_b 的直线方程，判断边角点与对应边界线的点线关系。

6.1.1 边角点的定位模型

首先我们需要确定龙头板凳的外侧两个边角点 A、B 的坐标，通过几何分析我们发现，龙头的位置有四种情况，如图 9 所示，其中 P_0 表示龙头前把手的中心点，P_0 表示龙头后把手（即第一节龙身前把手）的中心点，龙头的四种位置情况分别如图 15 序号①、②、③、④ 所示，龙头的不同位置会影响对边角点 A、B 坐标的计算。

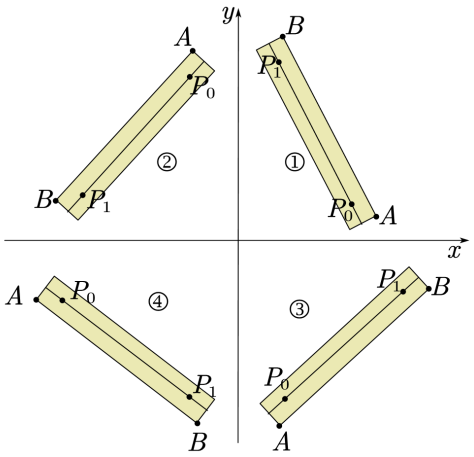

图 **9**　龙头板凳的四种位置示意图

易知边角点 A 与中心点 P_0 之间的距离、边角点 B 与中心点 P_1 之间的距离相等，我们设为 d；P_0A 与 P_0P_1 之间的夹角、P_1B 与 P_0P_1 之间的夹角相等，设为 β。

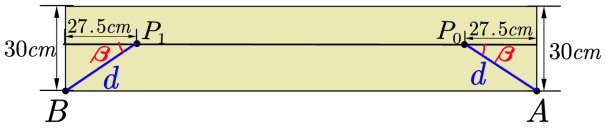

图 **10**　龙头的俯视图

下面以龙头的板凳位于第二象限时为例，求解出此时边角点 A、B 的坐标。设两个把手中心点的坐标分别为 $P_0(x_0, y_0), P_1(x_1, y_1)$，边角点 A 的坐标为 (x_A, y_A)，边角点 B 的坐标为 (x_B, y_B)。设 P_0P_1 所在的直线方程为 $y_0 = m_0x + n_0$。

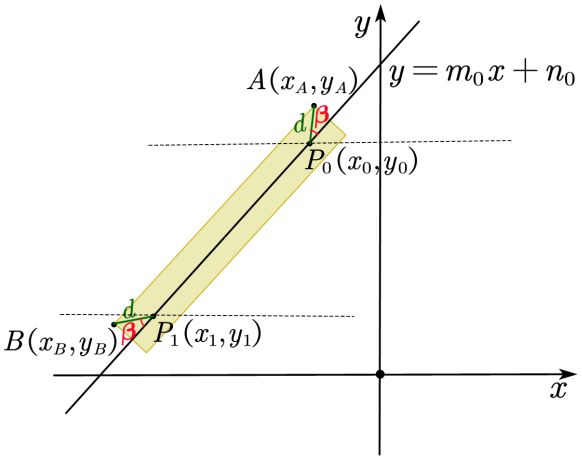

图 **11**　龙头的板凳位于第二象限的示意图

通过对龙头的把手中心点与边角点之间几何关系的分析，我们得到了此时边角点

A、B 的直角坐标

$$x_A = x_0 + d\cos(\arctan m_0 + \beta) \qquad y_A = y_0 + d\sin(\arctan m_0 + \beta)$$

$$x_B = x_1 - d\cos(\arctan m_0 - \beta) \qquad y_B = y_1 - d\sin(\arctan m_0 - \beta)$$

经过讨论，我们发现，当 $n_0 > 0$ 时（图 9 中 ①、② 两种情况），得到的结果就和上述表达式相同；当 $n_0 < 0$ 时（图 9 中 ③、④ 两种情况），A、B 的直角坐标可由以下公式得到

$$x_A = x_0 - d\cos(\arctan m_0 + \beta) \qquad y_A = y_0 - d\sin(\arctan m_0 + \beta)$$

$$x_B = x_1 + d\cos(\arctan m_0 - \beta) \qquad y_B = y_1 + d\sin(\arctan m_0 - \beta)$$

综合考虑以上两种情况，我们得到了**龙头板凳的外侧两个边角点 A、B 的定位模型**

$$\begin{cases} x_A = x_0 + \dfrac{|n_0|}{n_0} d\cos(\arctan m_0 + \beta) \\ y_A = y_0 + \dfrac{|n_0|}{n_0} d\sin(\arctan m_0 + \beta) \\ x_B = x_1 - \dfrac{|n_0|}{n_0} d\cos(\arctan m_0 - \beta) \\ y_B = y_1 - \dfrac{|n_0|}{n_0} d\sin(\arctan m_0 - \beta) \end{cases} \tag{16}$$

其中，d 表示对应边角点与把手中心点之间的距离；

β 表示 P_0A 与 P_0P_1 之间、P_1B 与 P_0P_1 之间的夹角；

n_0 表示 P_0P_1 所在直线在 y 轴上的截距；

m_0 表示 P_0P_1 所在直线的斜率。

6.1.2 边界线的直线方程模型

我们首先研究如何确定距离边角点 A、B 最近的两条板凳 a、b，求解板凳 a、b 位于靠近龙头的板凳的一侧的边界所在直线的直线方程。通过平面几何的分析我们发现，板凳 a、b 之间存在两种位置关系：板凳 a、b 相邻；板凳 a、b 不相邻。当 OP_0 延长线与 OP_1 延长线所夹把手中心点个数为 1 时，设被夹的这个把手中心点为 P_i，它既是板凳 a 的后把手中心点，又是板凳 b 的前把手中心点，此时板凳 a、b 相邻（如图 12 所示）；当 OP_0 延长线与 OP_1 延长线所夹把手中心点个数不为 1 时，设被夹的把手中心点为 P_i、

P_{i+1}，点 P_i 是板凳 a 的后把手中心点，点 P_{i+1} 是板凳 b 的前把手中心点，此时板凳 a、b 不相邻（如图 13 所示）。

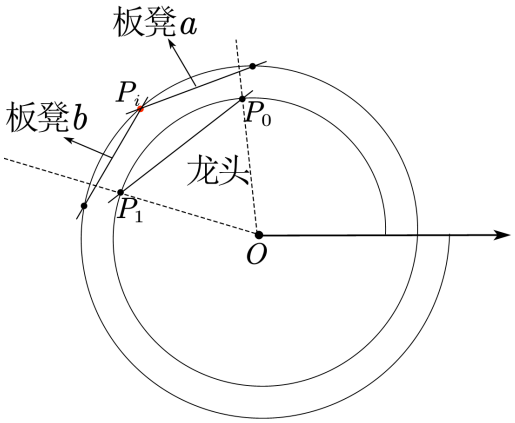

图 **12**　板凳 a、b 相邻

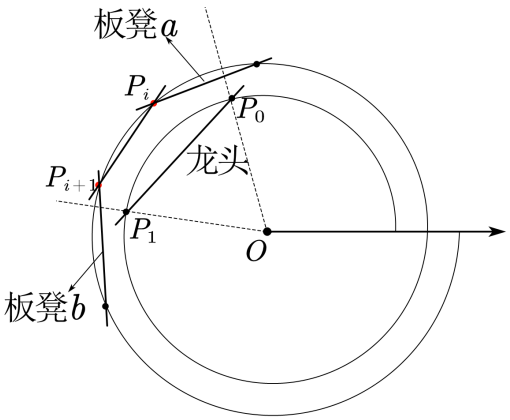

图 **13**　板凳 a、b 不相邻

接下来我们求解板凳 a、b 内侧边界线的直线方程。通过几何分析，我们发现龙身的板凳存在四种位置情况分别如图 14 序号 ①、②、③、④ 所示。首先我们设第 i 节龙身的龙身前把手和龙身后把手的所成直线的直线方程为 $y = m_i x + n_i$。根据平行线的性质，我们设此时第 i 节龙身靠内侧的板凳的边界线方程为

$$y = m_i x + n_i' \tag{17}$$

其中，

$$n_i' = n_i - \frac{0.15\,|n_i|}{n_i}\sqrt{1 + m_i^2}$$

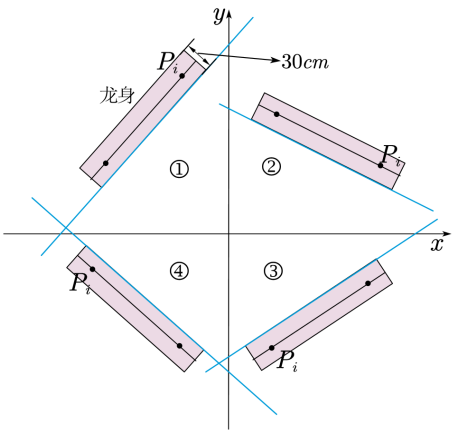

图 **14**　龙身板凳的四种位置示意图

综合考虑以上四种情况，我们得到了**板凳 a、b 内侧边界线的定位模型**

$$y = m_i x + n_i - \frac{0.15\,|n_i|}{n_i}\sqrt{1 + m_i^2} \tag{18}$$

其中，m_i 表示第 i 节龙身的前把手中心点和后把手中心点所成直线的斜率。

6.1.3 边角点与边界线的点线关系模型

接下来我们通过边角点与边界线的点线关系，判断龙头的板凳是否与其外侧相邻板凳相撞。对于边角点 A 与板凳 a 内侧边界线，边角点 B 与板凳 b 内侧边界线的关系，我们经过探讨分析，得出以下结论：不管是对于 A 点还是 B 点，统一设其直角坐标为 (x, y)，当 A、B 同时满足关系式（19）时，龙头和龙身之间不会发生碰撞；反之，若 A、B 和板凳内侧边界线的关系之间有一个不满足该关系式时，龙头和龙身之间会发生碰撞。

$$(m_i x + n_i' - y) n_i' > 0 \tag{19}$$

其中，m_i 为此时板凳 a 或板凳 b 的斜率；n_i' 为此时板凳 a 或板凳 b 对应内侧边界线在 y 轴上的截距。

6.2 问题二的模型求解

本问题中，舞龙队沿问题一设定的螺线 $r = \frac{0.55}{2\pi}\theta$ 盘入，我们首先求解出龙头的板凳恰好未与其外侧相邻板凳发生碰撞的临界位置，得到**龙头的板凳恰好未与其外侧相邻板凳发生碰撞的临界时刻为 412.180374s**。然后利用龙头前把手中心点的位置和速度，求解出这个时刻整个舞龙队的位置和速度。

表 **3**　问题二求解结果示例

	横坐标 x (m)	纵坐标 y (m)	速度 (m/s)
龙头	0.957076	2.091336	1.000000
第 1 节龙身	-1.851116	1.549433	0.991744
第 51 节龙身	1.005337	4.404580	0.977218
第 101 节龙身	-0.251175	-5.903437	0.974919
第 151 节龙身	1.251544	-6.915839	0.973979
第 201 节龙身	-7.847207	-1.512651	0.973467
龙尾（后）	0.672311	8.353459	0.973309

图 **15**　临界时刻各把手中心点的位置示意图

在龙头的板凳恰好未与其外侧相邻板凳发生碰撞的临界时刻下，各把手中心点的位置如图 15 所示，此时龙头的板凳外侧的边角点恰好未与第 8 节、第 10 节龙身发生碰撞。

七、 问题三模型的建立与求解

在本问题中需要我们设计盘入螺线的最小螺距 p，使得龙头前把手中心点能够到达盘入螺线圆形调头空间的边界，而不与其他板凳发生碰撞。

首先我们联立盘入螺线的极坐标方程 $r = \frac{p}{2\pi}\theta$ 与圆形调头区域的边界方程 $r = 4.5$，得到龙头前把手中心点进入调头区域时的极角 $\theta = \frac{9\pi}{p}$

运用问题二中判断板凳之间是否相撞的方法，得到螺距为 p 时，龙头与其外侧板凳相撞的临界位置的极角 θ_c，为了使龙头前把手中心点能够沿着螺线盘入到调头空间的边界，要求 $\theta \geqslant \theta_c$。

下面为使用**二分法**查找满足条件的最小螺距的算法步骤

step1 初始化上下界：

为螺距 p 设置合适的上下边界 low、$high$，要求包括我们所求的最小螺距。设 tol 为容许误差，可将其设置为一个极小的值，如 10^{-6}。

step2 二分查找：

将螺距 p 取中间值 $\frac{low+high}{2}$，得到这种情况下龙头前把手中心点进入调头区域时的极角 $\theta_{mid} = \frac{9\pi}{p}$。

step3 更新边界值：

运用问题二中判断板凳之间是否相撞的方式，得到螺距为 $p = \frac{low+high}{2}$ 时，龙头与其外侧板凳相撞的临界位置的极角 θ_c，比较此时 θ_{mid} 与 θ_c 的大小关系。若 $\theta_{mid} > \theta_c$，说明螺距为中间值时龙头前把手不能进入调头区域，更新下边界为 p；若 $\theta_{mid} \leqslant \theta_c$，说明螺距为中间值时龙头前把手可以进入调头区域，更新上边界为 p。

step4 判断是否继续迭代：

若 $high - low \leqslant tol$，回到 step2 进行迭代；

若 $high - low > tol$，停止迭代，输出满足条件的最小螺距 p。

图 **16**　最小螺距条件下的临界时刻各把手中心点的位置示意图

使用 python 编写程序求解，得到**满足约束条件的最小螺距 p=0.452209m**，达到调头空间的边界时，龙头前把手的中心点的极角为 $\theta = 19.902281\pi$，此时各把手中心点的位置如图 16 所示。

八、 问题四模型的建立与求解

8.1 问题四的模型准备

在本问题中，设盘入螺线的极坐标方程为 $r = \frac{p}{2\pi}\theta$，其中 $p = 1.7m$，盘出螺线与盘入螺线关于螺线中心呈中心对称，即**盘出螺线绕中心旋转 π 可与盘入螺线重合**，得到盘出螺线的极坐标方程为 $r = \frac{p}{2\pi}(\theta + \pi)$。联立螺线与调头区域边界的极坐标方程，得到盘入螺线与调头区域的交点 $Q_1\left(4.5, \frac{9\pi}{1.7}\right)$、盘出螺线与调头区域的交点 $Q_2\left(4.5, \frac{7.3\pi}{1.7}\right)$。$Q_1$、$Q_2$ 之间的调头路径是由两段圆弧相切连接而成的 S 形曲线，前一段圆弧的半径是后一段的 2 倍，在点 Q_1、Q_2 处调头路径的两条弧线分别与盘入、盘出螺线相切。

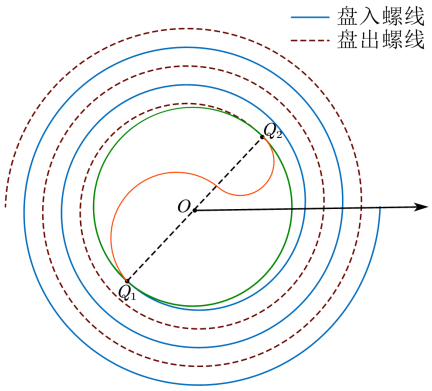

图 **17**　舞龙队掉头过程路径示意图

我们首先来证明，在保持各部分相切关系时，**调整圆弧不会改变调头路径的长度**。

设前一段圆弧的半径 R_1 是后一段圆弧半径 R_2 的 n 倍，前一段圆弧的弧长为 s_1、半径为 R_1，后一段圆弧的弧长为 s_2、半径为 R_2。设调头路径的起点为 $Q_1\,(r_1,\theta_1)$、终点为 $Q_2\,(r_2,\theta_2)$，两端圆弧的交点设为 Q_3。由中心对称图形的性质，我们得到 Q_1、Q_2 关于原点对称，它们之间的距离 $d=r_1+r_2=9m$。因为圆 O_1、O_2 相切于点 Q_3，由相似三角形的判定方法，我们得到 $\Delta O_1Q_1Q_3\sim\Delta O_2Q_2Q_3$，$Q_1$、$Q_3$ 之间的距离 $d_1=\frac{n}{n+1}d$，Q_2、Q_3 之间的距离 $d_2=\frac{1}{n+1}d$。

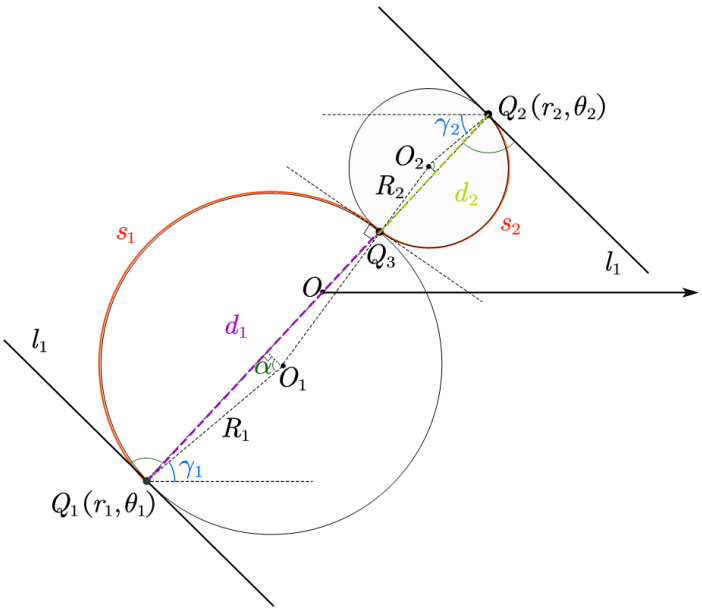

图 **18**　调头路径示意图

为了得到两段圆弧的弧长 s_1、s_2，我们首先需要求解 α。盘入螺线在 Q_1 和圆 O_1 相切，切线 l_1 的斜率设为 k_1；Q_1、Q_2 连线的斜率设为 k_2，利用问题一中的求解方法得到

$$\begin{cases}\alpha=\arctan\left|\dfrac{k_1-k_2}{1+k_1k_2}\right|\\ k_1=\dfrac{\sin\theta_1+\theta_1\cos\theta_1}{\cos\theta_1-\theta_1\sin\theta_1}\\ k_2=\dfrac{r_2\sin\theta_2-r_1\sin\theta_1}{r_2\cos\theta_2-r_1\cos\theta_1}\end{cases}\tag{20}$$

因为 $\theta_1=\theta_2+\pi$，可将式（20）化简得到

$$\alpha=\arctan\theta_1$$

其中，θ_1 是调头路径的起点 Q_1 处的极角。

然后我们利用 α 求解两段圆弧的半径 R_1、R_2，从而求解两段圆弧的弧长 s_1、s_2。由直角三角形中的几何关系我们得到

$$R_1 = \frac{n}{2(n+1)} \frac{d}{\sin\alpha} \tag{21}$$

$$R_2 = \frac{1}{2(n+1)} \frac{d}{\sin\alpha} \tag{22}$$

由圆的弧长计算公式 $s = 2\alpha R$

$$s_1 = \frac{n}{(n+1)} \frac{\alpha}{\sin\alpha} d \tag{23}$$

$$s_2 = \frac{1}{(n+1)} \frac{\alpha}{\sin\alpha} d \tag{24}$$

则**整个调头路径的长度**为

$$s = s_1 + s_2 = \frac{\alpha}{\sin\alpha} d \tag{25}$$

其中，

$$\begin{cases} d = r_1 + r_2 \\ \alpha = \arctan\theta_1 \\ k_1 = \dfrac{\sin\theta_1 + \theta_1\cos\theta_1}{\cos\theta_1 - \theta_1\sin\theta_1} \\ k_2 = \dfrac{r_2\sin\theta_2 - r_1\sin\theta_1}{r_2\cos\theta_2 - r_1\cos\theta_1} \end{cases}$$

调头路径的长度 s 是一个与 n 无关的定值，因此我们得出结论：**在保持各部分相切关系时，调整圆弧不会改变调头路径的长度**。下面我们建立模型，研究舞龙队沿着规定的调头路径行进时，每个时刻各把手中心点的位置和速度。

为了建模部分计算方便，我们在这里求解圆心坐标 $O_1\,(a_1, b_1)$、$O_2\,(a_2, b_2)$，由图 24 中的几何关系我们得到

$$\gamma_1 = (\theta_1 - 5\pi) - \left(\frac{\pi}{2} - \alpha\right)$$

$$\gamma_2 = (\theta_2 - 4\pi) - \left(\frac{\pi}{2} - \alpha\right)$$

圆 O_1 的直角坐标为

$$\begin{cases} a_1 = r_1\cos\theta_1 + R_1\cos\gamma_1 \\ b_1 = r_1\sin\theta_1 + R_1\sin\gamma_1 \end{cases} \tag{26}$$

圆 O_2 的直角坐标为

$$\begin{cases} a_2 = r_2\cos\theta_2 - R_2\cos\gamma_2 \\ b_2 = r_2\sin\theta_2 - R_2\sin\gamma_2 \end{cases} \tag{27}$$

8.2 问题四的模型建立

在本问题中，各把手中心点所在曲线分为四种情况：盘入螺线、前一段调头圆弧、后一段调头圆弧和盘出螺线。由于龙头前把手的行进速度保持不变，我们可以通过研究不同时刻龙头前把手中心点走过的曲线长度，确定不同时刻龙头前把手中心点的位置。

8.2.1 龙头前把手中心点的定位模型

我们设某时刻 t 龙头前把手中心的极坐标为 $P_0(r,\theta)$，直角坐标设为 $P_0(x,y)$。已知龙头前把手中心的行进速度 v_0，则 t 时刻它相对 $Q_1(r_1,\theta_1)$ 点走过的螺线长度 $l=v_0t$，我们根据 l 的大小来判断龙头前把手中心点所在的曲线。由式（23）和式（24）我们得到了前后两端圆弧的长度 s_1 和 s_2。

（1）当 $l\leqslant 0$ 时，龙头前把手中心点位于盘入螺线上，通过第一类曲线积分计算 l 的长度

$$l=\frac{p}{2\pi}\int_{\theta}^{\theta_1}\sqrt{\theta^2+1}d\theta \tag{28}$$

其中，θ_1 为边界点 Q_1 的极角。

通过求解式（28）的数值解得到（2）当 $l>s_1+s_2$ 时，龙头前把手中心点位于盘出螺线上，通过第一类曲线积分计算 $l-(s_1+s_2)$ 的长度

$$l-(s_1+s_2)=\frac{p}{2\pi}\int_{\theta_2}^{\theta}\sqrt{\theta^2+1}d\theta \tag{29}$$

其中，θ_2 为边界点 Q_2 的极角。

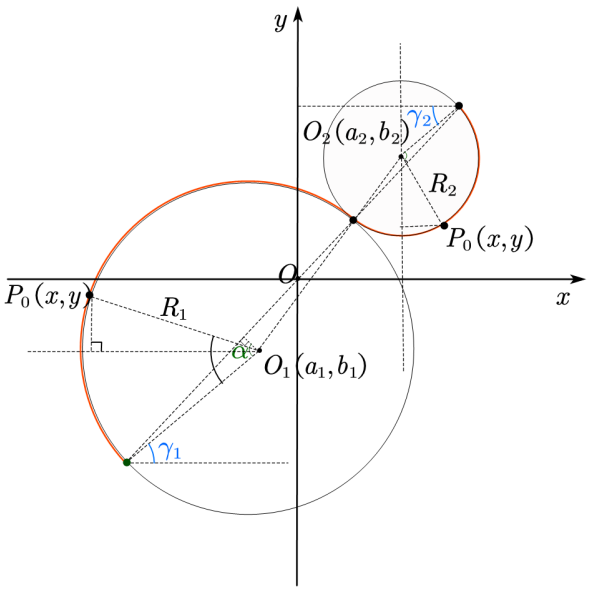

图 **19**　龙头前把手中心点 P_0 定位示意图

（3）当 $0 < l \leqslant s_1$ 时，龙头前把手中心点位于前一段调头圆弧上，由式（21）可以得到该圆弧的半径 R_1，由式（26）可以得到圆心 O_1 的直角坐标 $O_1\,(a_1, b_1)$。由几何关系，我们得到点 P_0 的直角坐标

$$\begin{cases} x = a_1 - R_1 \cos\left(\dfrac{l}{R_1} - \gamma_1\right) \\ y = b_1 + R_1 \sin\left(\dfrac{l}{R_1} - \gamma_1\right) \end{cases} \tag{30}$$

其中，

$$\gamma_1 = (\theta_1 - 5\pi) - \left(\frac{\pi}{2} - \alpha\right)$$

（4）当 $s_1 < l \leqslant s_1 + s_2$ 时，龙头前把手中心点位于前后一段调头圆弧上，由式（22）可以得到该圆弧的半径 R_2，由式（27）可以得到圆心 O_2 的直角坐标 $O_2\,(a_2, b_2)$。由几何关系，我们得到点 P_0 的直角坐标

$$\begin{cases} x = a_2 + R_2 \cos\left(\dfrac{l - s_1}{R_2} - 2\alpha - \gamma_2\right) \\ y = b_2 + R_2 \sin\left(\dfrac{l - s_1}{R_2} - 2\alpha - \gamma_2\right) \end{cases} \tag{31}$$

其中，

$$\gamma_2 = (\theta_2 - 4\pi) - \left(\frac{\pi}{2} - \alpha\right)$$

8.2.2 相邻两把手中心点的定位模型

当龙头开始进行掉头时，相邻两个把手中心点存在不同的位置关系，下面我们给出了不同位置关系下通过前一个点的坐标计算后一个点的坐标的方法，并且给出了到底是属于哪种位置关系的判断条件。

相邻两点均在前一段圆弧上

判断条件：

$$2R_1 \arcsin\frac{L}{2R_1} < l_i < s_1 \tag{32}$$

其中，l_i 表示该点与 Q_1 所连接弧线的长度，即在题目中给定的 0 时刻时，该点走过的路程。

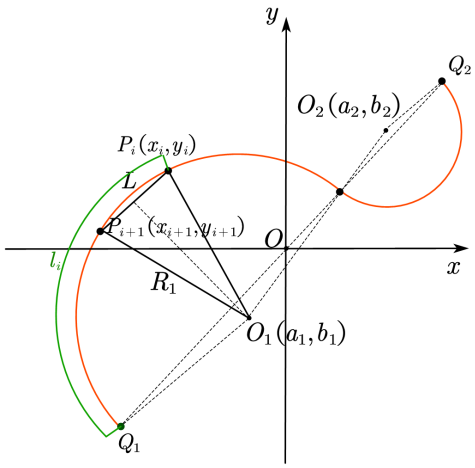

图 **20**　相邻两点均在前一段圆弧示意图

此时由几何关系可知，如果 P_i 和 P_{i+1} 均在前一段圆弧上，则有

$$l_{i+1} = l_i - 2R_1 \arcsin \frac{L}{2R_1} \tag{33}$$

此时应用公式（29）即可得出此时 P_{i+1} 的坐标

相邻两点位于两段圆弧交点两侧

判断条件：

$$s_1 < l_i < s_1 + 2R_2 \arcsin \frac{L}{2R_2} \tag{34}$$

我们设 $|P_iP_{i+1}| = L$，如果 $i = 0$，则 $L = 2.86m$；如果 $i \geqslant 1$，则 $L = 1.55m$ 在三角形 $O_1P_iP_{i+1}$ 中，根据余弦定理可得

$$\varphi = \arccos \frac{L^2 + |O_1P_i|^2 - {R_1}^2}{2L\,|O_1P_i|}$$

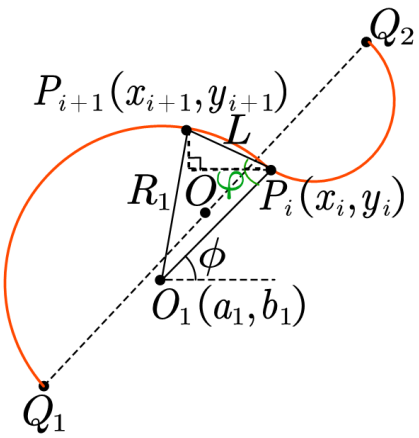

图 **21**　相邻两点位于两段圆弧交点两侧示意图

设直线 O_1P_i 与 x 轴的夹角为 ϕ，根据几何关系，我们可得

$$\begin{aligned} x_{i+1} &= x_i - L\cos(\varphi - \phi) \\ y_{i+1} &= y_i + L\sin(\varphi - \phi) \end{aligned} \tag{35}$$

其中，

$$\phi = \arctan\frac{y_i - b_1}{x_i - a_1}$$

相邻两个点均在后一段圆弧上

判断条件：

$$s_1 + 2R_2\arcsin\frac{L}{2R_2} < l_i < s_1 + s_2 \tag{36}$$

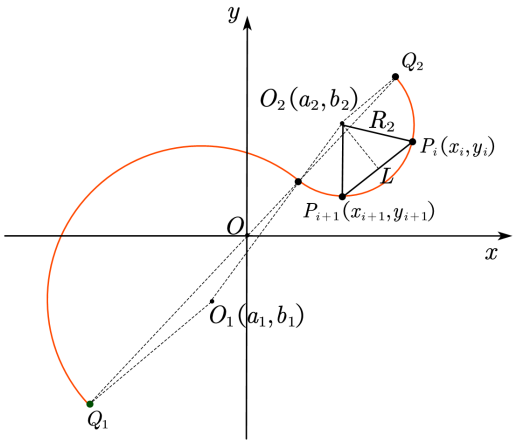

图 **22**　相邻两个点均在后一段圆弧示意图

和情况 1 类似，如果两个点均在圆弧 2 上，我们可以得到

$$l_{i+1} = l_i - 2R_2\arcsin\frac{L}{2R_2} \tag{37}$$

此时由公式（30）即可得出 P_{i+1} 的坐标。

8.2.3 相邻两把手中心点的关联速度模型

已知每个点的直角坐标，下面我们来进行速度计算，我们可以沿用第一问的思路

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - 2\alpha - \gamma_2 \tag{38}$$

$$v_{i+1} = \frac{v_i\cos\alpha_i}{\cos\alpha_{i+1}} \tag{39}$$

$$\alpha_i = \arctan\left|\frac{k_i - m_i}{1 + k_i m_i}\right| \tag{40}$$

$$\alpha_{i+1} = \arctan\left|\frac{k_{i+1} - m_i}{1 + k_{i+1}m_i}\right| \tag{41}$$

其中，k_i 的值需要进行分类讨论，且 ki 的表达式仅仅与该点的位置有关

当 P_i 在圆弧 1 时

$$k_i = -\frac{x_i - a_1}{y_i - b_1}$$

当 P_i 在圆弧 2 上时

$$k_i = -\frac{x_i - a_2}{y_i - b_2}$$

当 P_i 在螺线上（盘出螺线和盘入螺线）时

$$k_i = \frac{\sin\theta_i + \theta_i\cos\theta_i}{\cos\theta_i - \theta_i\sin\theta_i}$$

九、 问题五模型的建立与求解

本问题中整个舞龙队沿着问题四中规定的路径行进，当龙头前把手中心点的行进速度 v_0 不变时，**各把手中心点的速度 v_i 只与此时整个舞龙队的位置有关**，所以我们需要确定龙头前把手中心点位于哪个位置时会出现 v_i/v_0 的最大值，从而确定龙头的最大行进速度，使得在任意时刻各把手速度 v_i 均不超过 $2m/s$。

下面为我们使用**粒子群算法**求解 v_i/v_0 的最大值的算法步骤：

（1）初始化阶段

参数初始化：设置粒子速度（个体最大速度）$v = 0.5$；惯性权重 $\omega = 1$；个体学习系数 $c_1 = 2$；全局学习系数 $c_2 = 2$；最大迭代次数 $n = 1000$。

粒子随机初始化：将粒子群中每个粒子的位置定义为龙头前把手中心点 P_0 与调头路径起点 Q_1 之间的距离，设置粒子群的初始位置，范围为 $[0, 300]$，步长为 1。

适应度计算：设每个粒子对应的整个舞龙队所在位置下的各把手中心点的最大速度为 v_{max}，计算 v_{max} 与龙头速度 v_0 的比值，即每个粒子的适应度 fit。

（2）迭代过程

step1 修订惯性权重：

将惯性权重 ω 进行线性或非线性修改。

step2 更新粒子速度与位置

$$v_i^k = \omega v_i^{k-1} + c_1\left(\text{pbest}_i^{k-1} - x_i^{k-1}\right) + c_2\left(\text{gbest}^{k-1} - x_i^{k-1}\right) \tag{42}$$

其中，v_i^k 表示第 k 次迭代中第 i 个粒子的速度；x_i^{k-1} 表示 $k-1$ 次迭代中第 i 个粒子点的当前位置；pbest_i^{k-1} 表示第 k 次迭代前，第 i 个粒子记录的个体最优解；gbest_i^{k-1} 表示第 k 次迭代前，粒子群的全局最优解；ω、c_1、c_2 分别表示惯性、个体学习经验和全体学习经验的权重。

位移 = 速度 × 时间，算法中默认时间为 1，因此位移 = 速度。

step3 更新个体和全局最适位置

根据每个粒子的个体适应度，更新每个粒子的最优解 pbest$_i^k$，从而更新整个粒子群的全局最优解 gbest^k。

step4 继续迭代或输出全局最优解

若迭代次数 $k < n$，回到 $step1$ 继续进行迭代；

若达到最大迭代次数 n，停止迭代，输出全局最优解 gbest^n。

参考文献

[1] 刘崇军. 等距螺旋的原理与计算 [J]. 数学的实践与认识,2018,48(11):165-174.

[2] 杨智明, 夏春梅. 几种常用查找算法的对比分析 [J]. 保山学院学报,2017,36(02):57-59.

[3] 杨月敏. 中国民间体育——板凳龙 [J]. 现代语文,2006,(09):124-125.

[4] 刘建华. 粒子群算法的基本理论及其改进研究 [D]. 中南大学,2009.

[5] 张辉, 李应岐, 敬斌, 等. 第一类曲线积分的计算方法 [J]. 高等数学研究,2015,18(02):27-29.

附录 A　支撑材料文件列表

文件名	类型	简介
result1.xlsx	XLSX 工作表	问题一的计算结果
result2.xlsx	XLSX 工作表	问题二的计算结果
result4.xlsx	XLSX 工作表	问题四的计算结果
problem1.py	python 源文件	python 源程序代码

附录 B　python 源程序代码

```
import pandas as pd
from sympy import symbols, solve, sqrt,nsolve,integrate,Eq
import numpy as np
import sympy as sp
import openpyxl
from matplotlib import pyplot as plt
import scipy as sci

plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False # 用来正常显示负号

theta1=0
l=2.86
PI=np.acos(-1)
d=0.55# the distance between adjacent curves
b =d/(2*PI) #rho=b*theta
dot_num=224 #the number of the dots
pi=np.acos(-1)
BiasLen = np.sqrt(0.275 ** 2 + 0.15 ** 2) # the distance between the peak angle and the
    Drangon dot
BiasAngle=np.atan(15/17.5)#the angle between the peak angle and the Drangon dot centet line

def calu_theta2(theta1,b,l):
"""
Calculate the next theta
:param theta1: the former theta
:param b: the coefficent
:param l:the distance between the theta1 and theta_needed
:return:the next theta
"""
theta2=symbols('theta2')
res = nsolve((theta1 * b) ** 2 + (theta2 * b) ** 2 - 2 * (theta1 * b) * (theta2 * b) *
    sp.cos(theta2 - theta1) - l ** 2,theta2,theta1+0.1)
```

```
res=float(res)
return res

#calu the next pos of the head to move
def intf(thetaa):
"""
the function of integrand
:param theta:
:return: the function
"""
return np.sqrt(thetaa**2+1)
def distance(theta1,theta2):
"""
calculate the curve_distance between the two points
:param theta1:
:param theta2:
:return: distance
"""
res, _=sci.integrate.quad(intf,theta1,theta2)
res=res*b
return res

def move(dist,theta_now):
"""
calu the theta of the dragon_head dot(moved dist)
:param dist: the distance moved
:param theta_now: the theta of the dragon_head dot
:return: the tehta of the next dot
"""
# thetaa = symbols('thetaa')
# theta_next=symbols('thetaa')
# fx= b * integrate(mt.sqrt(thetaa ** 2 + 1), theta_next, theta_now)-dist
theta,theta_next=symbols('theta,theta_next')
intf=sp.sqrt(theta**2+1)
integral = b*integrate(intf,(theta,theta_next,theta_now))
f2=integral-dist
res=nsolve(f2,theta_next,theta_now)
return float(res)

def calu_k_tangent(theta):
"""
calculate the slope of tangent
:param theta:the pos now
:return:the slope
"""
k=(np.sin(theta)+theta*np.cos(theta))/(np.cos(theta)-theta*np.sin(theta))
return k
```

```
def get_xy(theta):
"""
calculate the x and y coordinates of the dot
:param theta:
:return:
"""
x=b*theta*np.cos(theta)
y=b*theta*np.sin(theta)
return x,y

def calu_k_dots(theta_former,theta_next):
"""
calculate the slope between adjacent dots
:param theta_former:
:param theta_next:
:return:
"""
x1,y1=get_xy(theta_former)
x2,y2=get_xy(theta_next)
return (y1-y2)/(x1-x2),x2,y2

def calu_degree(k1,k2):
"""
calculate the degree between two line(slope: k1,k2)
:param k1: slope
:param k2: slope
:return: degree between two lines
"""
res=np.atan(abs((k1-k2)/(1+k1*k2)))
return res

def calu_speed(theta1,theta2,v1):
"""
calculate the speed of the next dot
:param v1:the speed of the theta1_dot
:param theta1: the theta of the dot whose speed is known
:param theta2: the theta of the dot whose speed is need_calu
:return: the speed of the theta2_dot
"""
k1=calu_k_tangent(theta1)#the targent_slope of theta1
k2=calu_k_tangent(theta2)
com_k,x2,y2=calu_k_dots(theta1,theta2)#the slope between theta1 and theta2
betal1=calu_degree(k1,com_k)#the angle between the targent_line and the speed_direc
betal2=calu_degree(k2,com_k)
v2=v1*np.cos(betal1)/np.cos(betal2)
return v2,x2,y2
```

```
def calu_all_pos_sped(theta_head,b):
"""
calculate the position of the all dots
:param theta_head:
:param b: the param of rho=a+b*theta
:return: the pos and speed of the dot behind theta_head
"""
theta1=theta_head
print(f'当前龙头处的theta角为{theta_head}')
b=b
dot_num=224#the number of the dots
thetas = [theta1] # store the thetas of every dot
v1=1#the speed of the former dot
speeds=[v1]
l = 2.86 # the distance between dragon_head and the second dot
x1,y1=get_xy(theta1)
xx=[x1]
yy=[y1]
for i in range(2,dot_num+1):
if i != 2:
l=1.65#the distance between adjacent dots behind
theta2=calu_theta2(theta1,b,l)
thetas.append(theta2)
v2,x2,y2=calu_speed(theta1,theta2,v1)
xx.append(x2)
yy.append(y2)
speeds.append(v2)
v1=v2
theta1=theta2
return thetas,speeds,xx,yy


def plotFigure(dragonTheta,b,name):
"""
plot the picture of dragon status whose headTheta is dragonTheta
:param dragonTheta:
:param b:
:return:
"""
thetaaa,speeddd,_,_=calu_all_pos_sped(dragonTheta,b)
ax=plt.subplot(111,projection='polar')
# plot the curve
deg=np.linspace(0,32*PI,1000)
ax.plot(deg,b*deg,alpha=0.1)#plot the curve
a=np.linspace(0,2*PI,1000)
ax.plot(a,4.5*np.ones(np.shape(a)),'--',color='blue',alpha=0.5,label='掉头空间边界')
ax.scatter(thetaaa,np.multiply(b,thetaaa),s=5,color='purple',alpha=0.5)
```

```
thetaPre=thetaaaa[0]
cnt=0
lines=[]
for i in range(1,dot_num):
thetaNext=thetaaaa[i]
if thetaNext>32*PI:
if cnt==0:
cnt=i
line=ax.plot([thetaPre,thetaNext],[thetaPre*b,thetaNext*b],color='green',alpha=0.5,label='在螺线外部分')
else:
line=ax.plot([thetaPre,thetaNext],[thetaPre*b,thetaNext*b],color='green',alpha=0.5,label='在螺线内部分')
thetaPre=thetaNext
lines.append(line)
# plt.setp(lines[1:cnt-1],label='_')
# plt.setp(lines[cnt+1:dot_num+1],label='_')
plt.setp(lines,label='_')
plt.title(name)
plt.legend(loc='upper left')
plt.axis('off')
plt.savefig(name)
plt.show()


def caluThetaToCheak(thetaHead,b):
"""
calu the four theta need to cheak
:param thetaHead: the theta of the Dragon head
:param b: the coefficient
:return:the four theta need to cheak
"""
thetaHead1=thetaHead
lb=thetaHead1+2*PI#the right line to judge
l=2.86#(1.65)
thetaHead2=calu_theta2(thetaHead,b,l)
rb=thetaHead2+2*PI#the left line to judge
thetaPre=thetaHead1
thetaTemp=thetaHead2

l=1.65#the distance between the adjacent dots
for i in range(2,dot_num):
if (thetaPre<lb)&(thetaTemp>=lb):
thetaPre1=thetaPre
thetaPre2=thetaTemp#the former board's dots
elif (thetaPre<rb)&(thetaTemp>=rb):
thetaNext1=thetaPre
thetaNext2=thetaTemp#the latter bound's dots
break
```

```
thetaPre = thetaTemp
thetaTemp = calu_theta2(thetaTemp, b, l)
# print('b',b)
return thetaHead1,thetaHead2,thetaPre1,thetaPre2,thetaNext1,thetaNext2


def get_line(theta1,theta2):
"""
calu the line of the two theta
:param theta1:
:param theta2:
:return: the coeficient(k,b) of the formual(y=kx+b)
"""
x1,y1=get_xy(theta1)
x2,y2=get_xy(theta2)
k=(y1-y2)/(x1-x2)
b=y1-k*x1
return k,b,x1,y1,x2,y2

def cheakPeakLine(x,y,k,b):
"""
cheak the collapse between peak and line
:param x: the x coordinate of the peak
:param y: the y coordinate of the peak
:param k: the k coefficient of the center line
:param b: the b coefficient of the center line
:return: the reuslt of the judge suceess:1 false 0
"""
D = 0.15 # the distance between the edge and the center line of the board
kPreEg = k
bPreEg = b - D * np.sqrt(1 + kPreEg ** 2) * (b / abs(b))
# cheak the dot inner the line
res = ((kPreEg * x + bPreEg - y) * bPreEg > 0) # satissfy: 1
return res

def cheak(thetaHead1,thetaHead2,thetaPre1,thetaPre2,thetaNext1,thetaNext2):
"""
find the thetas of the bord_to_cheak
:param thetaHead:the theta of Dragon
:param thetaPre1:the former theta of the board aganst the former angle
:param thetaPre2:the latter theta of the board aganst the former angle
:param thetaNext1:the former theta of the board aganst the latter angle
:param thetaNext2:the latter theta of the board aganst the latter angle
:return:whether satisfy the conditions no collapse:1
"""
kHead,bHead,xHead1,yHead1,xHead2,yHead2=get_line(thetaHead1,thetaHead2)
kPre,bPre,xPre1,yPre1,xPre2,yPre2=get_line(thetaPre1,thetaPre2)
```

```
kNext,bNext,xNext1,yNext1,xNext2,yNext2=get_line(thetaNext1,thetaNext2)
#calu the x,y of the Dragon head angle to cheak
xPeak1=xHead1+BiasLen*np.cos(BiasAngle+np.atan(kHead))*(b/abs(b))
yPeak1=yHead1+BiasLen*np.sin(BiasAngle+np.atan(kHead))*(b/abs(b))
xPeak2=xHead2-BiasLen*np.cos(np.atan(kHead)-BiasAngle)*(b/abs(b))
yPeak2=yHead2-BiasLen*np.sin(np.atan(kHead)-BiasAngle)*(b/abs(b))
#calu the b of the edge_line of the two board
sign1=cheakPeakLine(xPeak1,yPeak1,kPre,bPre)
sign2=cheakPeakLine(xPeak2,yPeak2,kNext,bNext)
return sign1*sign2

def bs_ans(thetaLb,thetaUb,threshold,b):
"""
binary search the ans under the threshold
:param thetaLb: left boundary
:param thetaUb: right boundary
:param threshold: the difference between the precise_ans and the search_ans
:return: ans_finded
"""
l=thetaLb
r=thetaUb
while(r-l>threshold):
mid=(l+r)/2
print(mid)
thetaHead1,thetaHead2,thetaPre1,thetaPre2,thetaNext1,thetaNext2=caluThetaToCheak(mid,b)
if cheak(thetaHead1,thetaHead2,thetaPre1,thetaPre2,thetaNext1,thetaNext2):
r=mid
else:
l=mid
return r


def calu_k_tangent_withPI(theta):
"""
calculate the slope of tangent
:param theta:the pos now
:return:the slope
"""
k = (np.sin(theta+PI) + theta * np.cos(theta+PI)) / (np.cos(theta+PI) - theta *
    np.sin(theta+PI))
return k

def get_xy_withPI(thetaLen):
"""
calculate the x,y coordinates of the dot added with PI
:param thetaLen: the theta to calu the length(not added with PI)
:return: x,y
```

```
"""
x = b * thetaLen * np.cos(thetaLen + PI)
y = b * thetaLen * np.sin(thetaLen + PI)
return x, y

def calu_k_dots_withPI(theta_former,theta_next):
"""
calculate the slope between adjacent dots added with PI
:param theta_former:the theta to calu the length(not added with PI)
:param theta_next:
:return:
"""
x1,y1=get_xy_withPI(theta_former)
x2,y2=get_xy_withPI(theta_next)
return (y1-y2)/(x1-x2),x2,y2

def calu_speed_withPI(theta1,theta2,v1):
"""
calculate the speed of the next dot
:param v1:the speed of the theta1_dot
:param theta1: the theta of the dot whose speed is known
:param theta2: the theta of the dot whose speed is need_calu
:return: the speed of the theta2_dot
"""
k1=calu_k_tangent_withPI(theta1)#the targent_slope of theta1
k2=calu_k_tangent_withPI(theta2)
com_k,x2,y2=calu_k_dots_withPI(theta1,theta2)#the slope between theta1 and theta2
betal1=calu_degree(k1,com_k)#the angle between the targent_line and the speed_direc
betal2=calu_degree(k2,com_k)
v2=v1*np.cos(betal1)/np.cos(betal2)
return v2,x2,y2

def calu_theta2_withPI_cheak(theta1,theta2,l):
"""
calu the value for the bscalu_theta2_withPI_cheak
:param tehta1:
:param theta2:
:param l:
:return:
"""
value=(theta1 * b) ** 2 + (theta2 * b) ** 2 - 2 * (theta1 * b) * (theta2 * b) *
    np.cos(theta2 - theta1) - l ** 2
if value==0:
return 0
elif value<0:
return -1
else:
```

```
return 1

def calu_theta2_withPI(theta1,b,l):
"""
binary search Calculate the next theta
:param theta1: the former theta
:param b: the coefficent
:param l:the distance between the theta1 and theta_needed
:return:the next theta
"""
# theta2=symbols('theta2')
# res = nsolve((theta1 * b) ** 2 + (theta2 * b) ** 2 - 2 * (theta1 * b) * (theta2 * b) *
    sp.cos(theta2 - theta1) - l ** 2,theta2,theta1-1)
# res=float(res)
lb=thetain-PI
ub=theta1
threshold=0.00001
mid=0
while(ub-lb>threshold):
mid=(lb+ub)/2
value=calu_theta2_withPI_cheak(theta1,mid,l)
if value==0:
return mid
elif value==-1:
ub=mid
else:
lb=mid
mid=(ub+lb)/2
return mid

def caluDotsAfterExit(dragonTheta):
"""
calut the dots after the out_port
:param dragonTheta: the theta of the head
:return: the x,y of the dots and the theta of the least dot after the out_port(not added
    with PI)
"""
cnt=1
l=2.86
xx=[]
yy=[]
x,y=get_xy_withPI(dragonTheta)
xx.append(x)
yy.append(y)
pre_theta=dragonTheta
nextTheta=calu_theta2_withPI(dragonTheta,b,l)
while nextTheta>thetain:
```

```
l = 1.65
x,y=get_xy_withPI(nextTheta)
xx.append(x)
yy.append(y)
pre_theta = nextTheta
nextTheta=calu_theta2_withPI(nextTheta,b,l)
return xx,yy,pre_theta,l


def getCicle2_xy(road):
x = x2_circle + r2 * np.sin((road - first_curve_road) / r2 - gamma2)
y = y2_circle - r2 * np.cos((road - first_curve_road) / r2 - gamma2)
return x,y

def getCicle1_xy(road):
x = x1_circle - r1 * np.cos(road / r1 - gamma1)
y = y1_circle + r1 * np.sin(road / r1 - gamma1)
return x, y

def caluDOtsInCicle2(road,l):
"""
calut the x,y of the dots in cicle2
:param road: the road of the dot
:param l: the length of the fisrt board in cicle2
:return: the x,y of the dots in cicle2 and the road of the last dot in cicle2 and the length
    of the next road to calu
"""
xx=[]
yy=[]
x,y=getCicle2_xy(road)
xx.append(x)
yy.append(y)
pre_road = road
temp_road=road-2*r2*np.asin(l/(2*r2))

while temp_road>first_curve_road:# in the circle2
x,y=getCicle2_xy(temp_road)
xx.append(x)
yy.append(y)
l=1.65
pre_road=temp_road
temp_road=temp_road-2*r2*np.asin(l/(2*r2))
return xx,yy,pre_road,l

def caluRoadInCircle2By_xy(x,y):
"""
calu the road to the intial dot
```

```
:param x: the x,y of the dots in cicle2
:return: the road to the intial dot
"""
# road=symbols('road')
# x=np.cos(thetaTo0)*rhoTo0
# y=np.sin(thetaTo0)*rhoTo0
# eq1=Eq(x,x2_circle+r2*sp.sin((road-first_curve_road)/ r2 - gamma2))
# eq2=Eq(y,y2_circle-r2*sp.cos((road-first_curve_road/r2 - gamma2)))
# eq=[eq1,eq2]
#res=sp.solve(eq,road)
distTemp=np.sqrt((x+inx/3)**2+(y+iny/3)**2) #
road=2*r2*np.asin(distTemp/(2*r2))+first_curve_road
return road

def caluRoadInCircle1By_xy(x,y):
"""
calu the road to the intial dot
:param x: the x of the dot in circle1
:param y: the y of the dot in circle1
:return: the road to the intial dot
"""
# road = symbols('road')
# eq1 = Eq(x, x1_circle + r1 * sp.sin(road / r1 - gamma1))
# # eq2 = Eq(y, y1_circle - r1 * sp.cos(road/ r1 - gamma1))
# # eq = [eq1, eq2]
# res = sp.nsolve(eq1,road,first_curve_road-0.1)
# return float(res)
distTemp=np.sqrt((x-inx)**2+(y-iny)**2)#the distance between the dot(x,y)in the cicle1 and
    the dot in(inx,iny)
road=2*r1*np.asin(distTemp/(2*r1))
return road

def caluFirstDotInCurve1(road,l):
"""
calu the dot in the curve1
:param road: the road of the last dot in cicle2
:param l: the length of the next board
:return: the theta of the next dot in the curve1(spiral1)
"""
x,y=getCicle1_xy(road)
theta_pre=np.atan(y/x)+PI #the degree of the dot in cicle1
theta,rho=symbols('theta,rho')
eq1=Eq(rho,b*theta)
eq2=Eq(l**2,x**2+y**2+rho**2-2*np.sqrt(x**2+y**2)*rho*sp.cos(theta-theta_pre))
eq=[eq1,eq2]
res=sp.nsolve(eq,(theta,rho),[thetain+0.1,(thetain+0.1)*b])
return float(res[0])
```

```
def calu_DotsInCircle1(road,l):
"""
Calculate the dot in the cicle1
:param road: the road of the first dot in the cicle1
:param l: the length of the first length of the board in cicle1
:return: the x and y coordinates of the dots and the road of the last dot in the cicle1 and
    the length of next board
"""
xx=[]
yy=[]
x,y=getCicle1_xy(road)
xx.append(x)
yy.append(y)
preroad = road
temproad=road-2*r1*np.asin(l/(2*r1))
while(temproad>0):
x,y=getCicle1_xy(temproad)
xx.append(x)
yy.append(y)
l=1.65
preroad=temproad
temproad=temproad-2*r1*np.asin(l/(2*r1))
return xx,yy,preroad,l

def caluFirstDotInCircle2(prex,prey,l):
"""
:param preTheta: the theta with the former dot
:param preRho:
:param cicleTheta: the thetai of the nextdot_in_cicle
:param dist:
:return:the the x,y
"""
# #theta np.float64(0.9239978392911148)
# rho_need,theta_need=sp.symbols('rho_need,theta_need')
# #
    eq1=sp.Eq(preRho,rho_need**2+cicleRho**2-2*rho_need*cicleRho*sp.cos(theta_need-cicleTheta))
#
    eq1=sp.Eq(r2**2,rho_need**2+rho_cicle2**2-2*rho_cicle2*rho_need*sp.cos(theta_need-theta_cicle2))
# # eq2=sp.Eq(dist**2,rho_need**2+preRho**2-2*rho_need*preRho*sp.cos(theta_need-preTheta))
# eq2=sp.Eq(dist**2,preRho**2+rho_need**2-2*rho_need*preRho*sp.cos(theta_need-preTheta))
# eq=[eq1,eq2]
# res=sp.solve(eq,[rho_need,theta_need],[3.5,0.92])
# print('calu_cicle2StartDot:')
# print('rho_need=',rho_need,'theta_need=',theta_need)
# return float(res['rho_need']),float(res['theta_need'])
dist_to_o=np.sqrt((prex-x2_circle)**2 + (prey-y2_circle)**2)
```

```
deg=np.acos((dist_to_o**2+l**2-r2**2)/(2*dist_to_o*l))
slope_angle=np.atan((prey-y1_circle)/(prex-x1_circle))
x=prex-l*np.cos(deg+slope_angle)
y=prey-l*np.sin(deg+slope_angle)
return x,y

def caluFirstDotInCicle1(road,l):
"""
calu the first dot in the cicle1
:param road: the road
:return: the x,y of the first dot in the cicle1 and the road of the first dot in the cicle1
"""
lastx, lasty = getCicle2_xy(road) # the x,y of the last dot in the cicle2
last_dist = np.sqrt((lastx - x1_circle) ** 2 + (lasty - y1_circle) ** 2) # the distance
    between the last dot in the cicle2 and the center in the cicle1
x=lastx-l*np.cos(np.arccos((l**2+last_dist**2-r1**2)/(2*l*last_dist))-np.atan((lasty-y1_circle)/(lastx-x1_cir
y=lasty+l*np.sin(np.arccos((l**2+last_dist**2-r1**2)/(2*l*last_dist))-np.atan((lasty-y1_circle)/(lastx-x1_cir
res_road=caluRoadInCircle1By_xy(x,y)
return x,y,res_road

def calu_DotsByHeadRoad(road):
"""
calculate the pos of the head of the dragon
:param road: the length of the road dragon visited
:return:dots_x the x of dots before the import dot
"""
l=2.86
x=[]
y=[]
if road<first_curve_road:
# x=x1_circle-r1*np.cos(road/r1-gamma1)# the x of the Dragon head
# y=y1_circle+r1*np.sin(road/r1-gamma1)#the y of the Dragon head
xx,yy,lastRoad,l=calu_DotsInCircle1(road,l)
x.extend(xx)
y.extend(yy)
thetaTemp = caluFirstDotInCurve1(lastRoad, l)
elif road<second_curve_road:
xx,yy,lastRoad,l=caluDOtsInCicle2(road,l)#the dots in the cicle2
x.extend(xx)
y.extend(yy)
xx,yy,roadTemp = caluFirstDotInCicle1(lastRoad, l)
l=1.65
xx, yy, lastRoad, l = calu_DotsInCircle1(roadTemp, l)
x.extend(xx)
y.extend(yy)
thetaTemp = caluFirstDotInCurve1(lastRoad, l)
else:
```

```
basex=-inx
basey=-iny
thetaTemp=move(-road+second_curve_road,thetain)
# xx,yy=get_xy_withPI(thetaTemp)
# x.append(xx)
# y.append(yy)
#add first part
xx,yy,last_theta,l=caluDotsAfterExit(thetaTemp)
x.extend(xx)
y.extend(yy)
#
    cicle2StartDotRho,cicle2StartDotTheta=caluFirstDotInCircle2(last_theta+PI,last_theta*b,theta_cicle2,rho_
    #from curve to the cicle2,calu the first dot in the cicle2
# roadTemp=caluRoadInCircle2By_thetaRho(cicle2StartDotTheta,cicle2StartDotRho)
x_next,y_next=caluFirstDotInCircle2(xx[len(xx)-1],yy[len(yy)-1],l)
roadTemp=caluRoadInCircle2By_xy(x_next,y_next)
#add second part
l = 1.65
resx,resy,lastRoad,l=caluDOtsInCicle2(roadTemp,l)
x.extend(resx)
y.extend(resy)
_,_,roadTemp=caluFirstDotInCicle1(lastRoad,l)#the last road in the cicle1
#add third part
xx,yy,lastRoad,l=calu_DotsInCircle1(roadTemp,l)
x.extend(xx)
y.extend(yy)
#forth part
thetaTemp=caluFirstDotInCurve1(lastRoad,l)
l=1.65
while(224-len(x)):#the number of dots need in curve1
xx,yy=get_xy(thetaTemp)
x.append(xx)
y.append(yy)
thetaTemp=calu_theta2(thetaTemp,b,l)
return x,y




# def caluFirstDotInCurve1_f(parm):
#     rho,theta = parm
#     x, y = getCicle1_xy(road)
#     theta_pre = np.atan(y / x) # the degree of the dot in cicle1
#     eq1 = Eq(rho, b * theta)
#     eq2 = Eq(l ** 2, x ** 2 + y ** 2 + rho ** 2 - 2 * np.sqrt(x ** 2 + y ** 2) * rho *
    sp.cos(theta - theta_pre))
#     return [eq1,eq2]
```

```
#problem1
dist=1#the distance of moved
theta_now=32*pi#the theta of the former dot 100.53096491487338
d=0.55# the distance between adjacent curves
b =d/(2*pi) #rho=b*theta
# l = 2.86 # the distance between adjacent dots
# wb=openpyxl.load_workbook('result1.xlsx')
# #calu the data of everytime
# for i in range(301):
#    print(f'第{i}秒时')
#    print(theta_now)
#    thetas,speeds,xx,yy=calu_all_pos_sped(theta_now,b)
#    df={'角度':thetas,'速度':speeds,'x坐标':xx,'y坐标':yy}
#    next_theta=move(dist,theta_now)
#    theta_now=next_theta
#    print(f'{'=' * 150}')
#    #store data
#    ws_pos=wb['位置']
#    ws_speed=wb['速度']
#    for j in range(1,dot_num+1):
#        ws_pos.cell(row=j*2,column=i+2,value=xx[j-1])
#        ws_pos.cell(row=j*2+1,column=i+2,value=yy[j-1])
#        ws_speed.cell(row=j+1,column=i+2,value=speeds[j-1])
# wb.save('result1.xlsx')
#plot the figure
# theta9=57.03207659834498
# theta0=32*PI
# thetaaa,speeddd,_,_=calu_all_pos_sped(theta0,b)
# ax=plt.subplot(111,projection='polar')
# deg=np.linspace(0,32*PI,1000)
# ax.plot(deg,b*deg,alpha=0.1)#plot the curve
# ax.scatter(thetaaa,np.multiply(b,thetaaa),s=5,color='purple',alpha=0.5)
# thetaPre=thetaaa[0]
# cnt=0
# lines=[]
# for i in range(1,dot_num):
#    thetaNext=thetaaa[i]
#    if thetaNext>32*PI:
#        if cnt==0:
#            cnt=i
#
    line=ax.plot([thetaPre,thetaNext],[thetaPre*b,thetaNext*b],color='red',alpha=0.5,label='在螺线外部分')
#    else:
#
    line=ax.plot([thetaPre,thetaNext],[thetaPre*b,thetaNext*b],color='green',alpha=0.5,label='在螺线内部分')
```

```
#     thetaPre=thetaNext
#     lines.append(line)
# plt.setp(lines[1:cnt-1],label='_')
# plt.setp(lines[cnt:dot_num],label='_')
# plt.title('0秒时状态示意图')
# plt.legend(loc='upper left')
# plt.axis('off')
# plt.savefig('0秒时示意图')
# plt.show()
# #plot
# df=pd.read_excel('result1.xlsx',sheet_name='速度')
# x=np.linspace(1,223,224)
#
# for i in range(0,301,60):
#     fig = plt.figure()
#     print(f'第{i}秒时')
#     data=df.iloc[:,i+1].values.tolist()
#     plt.plot(x,data)
#     plt.grid(linestyle='--',linewidth=0.5,color='gray',alpha=0.5)
#     plt.xlabel('点编号',fontsize=15,loc='right')
#     # plt.ylabel('速度',fontsize=10,loc='top')
#     plt.tick_params(axis='both', which='major', labelsize=14.5)
#     plt.title(f'第{i}秒的速度变化示意图')
#     plt.savefig(f'第{i}秒的速度变化示意图.png')
#     plt.close(fig)



# #problem2
# # res=bs_ans(0,57,0.00000001,b)
# # print(res)
# pro2_res_theta=26.27434970601462
# thetas, speeds, xx, yy = calu_all_pos_sped(pro2_res_theta, b)
# wr=wb=openpyxl.load_workbook('result2.xlsx')
# ws=wb.active
# for i in range(dot_num):
#     ws.cell(row=i+2,column=2,value=xx[i])
#     ws.cell(row=i+2,column=3,value=yy[i])
#     ws.cell(row=i+2,column=4,value=speeds[i])
# wr.save('result2.xlsx')
# costTime=distance(pro2_res_theta,32*PI)/1
# print("costTime=",costTime)
# plotFigure(pro2_res_theta,b,'碰撞时各点状态示意图')
```

```
# # #problem3
# # #binary find the distance of spiral under the d_threshold
# d_lb=0
# d_rb=1
# r=4.5
# d_threshold=0.0001
# # thetaIntersection=0
# b=0
# while(d_rb-d_lb>d_threshold):
#     d_mid=(d_lb+d_rb)/2
#     b=d_mid/(2*PI)
#     thetaIntersection=r/b #the theta of the point whose rho is r
#     print(d_mid)
#     thetaHead1, thetaHead2, thetaPre1, thetaPre2, thetaNext1, thetaNext2 =
    caluThetaToCheak(thetaIntersection, b)
#     if cheak(thetaHead1, thetaHead2, thetaPre1, thetaPre2, thetaNext1, thetaNext2):# no
    collapse
#         d_rb=d_mid
#     else:
#         d_lb =d_mid
# thetaIntersection=62.52485980938541
# plotFigure(thetaIntersection,b,'龙头前把手刚好沿着相应的螺线盘入到调头空间的边界状态示意图')
# print('success')




#problem4
b=1.7/(2*PI)
r=4.5
thetain = r / b #the theta of the point whose rho is r 16.63196110724008
k_tangent=calu_k_tangent(thetain) #-0.8540686593465319
inx,iny=get_xy(thetain) #-2.7118558637066594 -3.591077522761074
first_curve_road=6/thetain*abs(np.atan(thetain))*np.sqrt(1+thetain**2)
second_curve_road=9/thetain*abs(np.atan(thetain))*np.sqrt(1+thetain**2) #the road to the
    second terminal
##前100秒数据
# theta_now=thetain
# dist=1
# wb=openpyxl.load_workbook('result4.xlsx')
# ws_pos=wb['位置']
# ws_speed=wb['速度']
# for i in range(-100,1):
#     print(f'第{i}秒时')
#     print(theta_now)
#     thetas,speeds,xx,yy=calu_all_pos_sped(theta_now,b)
```

```
#     # df={'角度':thetas,'速度':speeds,'x坐标':xx,'y坐标':yy}
#     next_theta=move(-dist,theta_now)
#     theta_now=next_theta
#     print(next_theta)
#     print(f'{'=' * 150}')
#     # store data
#     for j in range(1,dot_num+1):
#         ws_pos.cell(row=j*2,column=i+102,value=xx[j-1])
#         ws_pos.cell(row=j*2+1,column=i+102,value=yy[j-1])
#         ws_speed.cell(row=j+1,column=i+102,value=speeds[j-1])
# wb.save('result4.xlsx')

#后100秒计算
vv=1
r1=3*np.sqrt(1+thetain**2)/thetain
r2=r1/2
gamma1=thetain-11/2*PI+np.atan(thetain) #the angle between y_axes and the curve1_edge_line
gamma2=np.atan(thetain)-thetain+5*PI#the angle between y_axes and the curve2_edge_line
x1_circle=inx+r1*np.cos(gamma1)
y1_circle=iny+r1*np.sin(gamma1)
x2_circle=-inx-r2*np.cos(gamma1)
y2_circle=-iny-r2*np.cos(gamma1)
theta_cicle2=np.arctan(y2_circle/x1_circle)
theta_cicle1=np.arctan(y1_circle/x1_circle)+PI
rho_cicle1=np.sqrt(x1_circle**2+y1_circle**2)
rho_cicle2=np.sqrt(x2_circle**2+y2_circle**2)
wb=openpyxl.load_workbook('result4.xlsx')
ws_pos=wb['位置']
# ws_speed=wb['速度']
for i in range(1,101):
print(f'第{i}秒时')
road=vv*i #the road of the dragon_head visitr
resx,resy=calu_DotsByHeadRoad(road)
print(len(resx),len(resy))
print(f'{'=' * 150}')
for j in range(1, dot_num + 1):
ws_pos.cell(row=j*2,column=i+102,value=resx[j-1])
ws_pos.cell(row=j*2+1,column=i+102,value=resy[j-1])
# ws_speed.cell(row=j+1,column=i+102,value=speeds[j-1])
wb.save('result4.xlsx')



#problem5
```