

“板凳龙”的优化设计

摘要

通过数学建模对盘龙所需要的面积，行进速度进行优化设计，我们可以达成将盘龙所需要的面积缩小，行进速度加快，最终提高盘龙的观赏性提升的目的。

针对问题一：我们构建了**板凳龙盘入模型**。首先，我们以螺线圆心为坐标原点建立平面直角坐标系，其次，我们利用平面几何及微分几何知识确定了龙头前把手关于时间 t 的函数，再以螺线方程和余弦定理递推出了前把手和后把手之间的位置、速度随时间的变化模型，最后算出给定区间的位置、速度，存放在 `result1.xlsx` 中。部分结果放于附录中表 1、表 2。

针对问题二：我们构建了碰撞检测模型。首先，我们将“板凳龙”的碰撞条件数学抽象为了点与线的共线问题，然后，借助问题一的模型加上碰撞检测机制求解出了盘龙的终止时刻为 $398s$ ，最后，同样借助问题一的模型，代入碰撞的时间点，即可求出相应时刻各把手的位置和速度，存放在 `result2.xlsx` 中。部分结构放于附录中表 3、表 4。

针对问题三：我们构建了最小螺距模型。参考问题二，我们将螺距合格的条件设立为碰撞发生的时间，龙头的前把手已经进入了调头区域或没发生过碰撞，然后，借助问题二的模型，我们利用二分法求解最小螺线，先大步长缩小最小螺距的范围，找到从符合条件到不合条件的范围之后，再用二分法不断逼近真实的最小螺距，求解得到最小螺距为 $0.3312m$ 。

针对问题四：首先利用圆与圆，圆与螺线间的几何关系推断出，调头曲线在保持各部分相切的前提下，无论如何调整圆弧，调头曲线都不会发生改变，然后在根据题目给出的条件和圆的几何关系，求解得到龙头前把手在调头曲线上的轨迹方程和相邻把手间位置的递推式，再设置步长为 $\Delta t = 0.001s$ ，计算一段极短时刻下，把手的路径长度，从而利用平均速度逼近瞬时速度，得到结果存放于 `result4.xlsx`，部分结果放于附录表 5，表 6。

针对问题五：首先借由前文的分析，我们可以得到盘入状态时龙头的速度最快，调头和盘出状态时龙头并非速度最快，其次，我们提出假设，龙尾在后两个状态时行进速度最快的，然后，我们通过问题四的模型求得了以一定步长取得的龙尾不同时刻的坐标，将相邻坐标求路程得到路程最大的时刻，也即为龙尾速度最快的时刻，将其设为 $2m/s$ 计算得新步长，最后，我们通过新步长得到 n 个可能的龙头的速度，对其求平均即可得到龙头的最大行进速度为 $1.957m/s$ 。

关键词： 盘入模型、碰撞检测机制、二分法、圆的几何关系

一、问题重述

1.1 问题一

问题一给出舞龙队盘入螺线的性质和螺距，假设各把手位于螺线上，龙头前把手以一定值进行移动，要求建立各把手的位置和速度时间变化的模型，得到准确，可为后续问题沿用的数据。

1.2 问题二

问题二在问题一的基础上加深，要求建立舞龙队的碰撞检测机制，并在问题一给定的螺线及龙头前把手的初始速度下，得到终止时刻，并求出终止时刻的各把手相应位置和速度结果。

1.3 问题三

问题三提出舞龙队在盘入之后，会预留调头空间为后续盘出做准备，问题三即是在给定调头空间为一半径为定值的圆形区域后，要求求出使得龙头前把手能够盘入调头空间而不终止的最小螺距。

1.4 问题四

问题四给定盘出螺线与盘入螺线的中心对称关系，要求在给定调头空间中调头曲线的形式为两相切圆弧连接成的曲线后，要求寻找该曲线是否还有调整变短的空间，并求出调头前后一定时间段内各把手的位置和速度结果。

1.5 问题五

问题五沿问题四的路径行进，要求在龙头保持速度不变的前提下，建立模型计算使得整支舞龙队从头到结束最大速度不超过一定定值的计算模型。

二、问题分析

2.1 问题一的分析

在问题一中，要求在舞龙队沿螺距为 55cm 的等距螺线顺时针盘入，龙头前把手的行进速度始终保持 1m/s 的前提下，求解 $0\text{s}\sim 300\text{s}$ 中以一秒为步长得到的每一时刻各把手的位置和速度。首先，我们以螺线中心为坐标原点建立平面直角坐标系，然后用平面几何微分几何知识、余弦定理建立各把手位置与时间的关系式，最后代入初始条件，得出各时刻下，各把手的位置和速度。

2.2 问题二的分析

在问题二中，要求在已知舞龙队沿问题一设定的螺线盘入的前提下，计算得到板凳之间恰不发生碰撞，也就是舞龙队盘入的终止时刻。首先，根据题目条件，我们可以将盘龙相撞的实际情况数学转化为龙头左上角的顶点与其中一节龙身的右侧边线共线，然后，由问题一的模型获取各时刻各把手的位置，最后，通过从 0s 开始模拟盘入，当盘入符合相撞的数学条件时，程序终止，得到的即是盘龙的终止时刻，并求出相应时间下各点的位置和速度即可。

2.3 问题三的分析

在问题三中，提出舞龙队为从盘入切换为盘出需要一定的调头空间，在假设调头空间为以螺线中心为圆心、直径为 9m 的圆形区域后，需要计算使得龙头能够在不相

撞的前提下能够进入调头区域的最小螺距。首先，我们将螺距合格的条件设立为当龙头与龙身相撞时，其与圆心的距离小于等于 $4.5m$ ；然后，通过问题二的模型，我们可以求出在给定螺距 a 的前提下，每次龙头碰撞时刻与圆心的相距距离；最后，我们通过二分法，以一不断减半的步长逐步逼近实际的最小螺距，达到计算出最小螺距的效果。

2.4 问题四的分析

在问题四中，我们在已知盘入螺线螺距为 $1.7m$ ，盘出螺线与盘入螺线关于螺线中心呈中心对称，调头路径是一段由两段圆弧相切形成的S形曲线，且与盘入盘出螺线相切，要求求出保持各部分相切条件下能否使得调头曲线变短，并求出从调头时刻开始前后 $100s$ 的各把手位置和速度。首先，我们根据等距螺线和调头路径的圆与圆之间的几何性质，可以由简单的几何知识推断出，调头曲线的长度在题目条件下是不变的，即无法使其变短；其次，我们可以根据圆与圆的几何性质，根据圆之间的关系和圆与盘入、盘出螺线的关系得出调头曲线的切点位置及其与坐标轴的角度关系；然后我们通过两圆之间半径的关系和问题一求出的 A_0 的坐标关系得出龙头前把手在调头及盘出螺线的坐标以及各把手间的递推关系；最后，根据龙头的位置及递推关系，我们就可以在初始参数下求得各个时刻各把手的位置和速度结果了。

2.5 问题五的分析

在问题五中，我们在已知行进路径的前提下，要求保持龙头行进速度不变，使得舞龙队各把手的速度不超过 $2m/s$ ，求出龙头最大可能的行进速度。首先，我们根据前文分析得到要求龙头的最大行进速度，只需找到调头和盘出状态的最大行进速度即可；其次，我们假设在这两个状态下，龙尾后把手的速度是最快，以此为基础，我们通过第四问的模型，可以定一步长求得龙尾不同状态的坐标；然后，我们找到两坐标之间路程最大的时刻，即为龙尾速度最快的时刻，将其设为 $2m/s$ ，与前面求出的最大路程相除可以得到一新步长；最后，通过该新步长，我们可以得到龙头在龙尾速度最快时刻可能的 n 个速度，对其求平均即可得到龙头最大的行进速度。

三、模型假设

- 1、假设盘龙移动过程中各板凳的高度一致
- 2、假设舞龙队能准确在所需要的运动轨迹上运动
- 3、假设舞龙队能以需要的速度稳定运动

四、符号说明

符号	说明
a	螺线的螺距
A_0	龙头前把手位置
A_i	第 i 节龙身前把手位置 ($i=1,2 \dots 222$)

续符号表

符号	说明
A_{233}	龙尾后把手位置
r_i	把手位置与圆心距离，即极径
α_i	把手位置积累绕过圆心的角度
s_i	把手位置所走过的路程
D_1, D_3	圆弧与切入切出螺线的切点
D_2	前圆弧和后圆弧的切点
R_1, R_2	分别指前后圆弧的半径
R_0	调整区域的半径
θ'	切入方向与调整圆切线的夹角

五、模型的建立与求解

5.1 板凳龙盘入模型

在问题一中，要求舞龙队在螺距为55cm的等距螺线^[1]顺时针盘入，且龙头前把手的行进速度保持1m/s，各把手中心始终位于螺线的前提下，建立**板凳龙各把手的位置、速度根据时间变化的数学模型**，并根据题目参数得出0~300s时间内所有把手的位置和速度。首先，我们建立二维直角坐标系，然后由平面几何及微分几何知识、余弦定理建立模型，代入具体参数值得到结果，具体操作如下。

5.1.1 模型的建立

我们以等距螺线原点为坐标原点，做平面直角坐标系。设龙头的前把手中心点为 A_0 ，后把手为 A_1 。由题可知，第2节板凳（即第1节龙身）开始，每节板凳的前把手也是前一节板凳的后把手，则第2节板凳的前后把手分别为 A_1 、 A_2 ，以此类推，第 i 节板凳的前后把手分别为 A_{i-1} 、 A_i 。又，设龙头长为 L_1 ，龙身和龙尾长为 L_2 ，螺距为 Q ，假设第 i 节板凳的前把手绕 O 点沿等距螺线逆时针旋转经过的总角度为 α_{i-1} （ $i = 223$ 时， α_{223} 指的是龙尾的后把手，下同）， r_i 为第 $i + 1$ 节板凳的前把手与 O 点的距离， S 为龙头所走的距离， t 为经过的时间，则 A_i 的坐标为 $(r_i \cos(-\alpha_{i-1}), r_i \sin(-\alpha_{i-1}))$ 。

为求 A_i 、 A_{i+1} 的坐标随时间变化的关系，由题目条件可列关系式如下：

$$r_i = 16a - \frac{a}{2\pi} \alpha_i \quad (1)$$

$$\alpha_i = 32\pi - \frac{2\pi}{a} r_i \quad (2)$$

对于龙头所走路程 s 和角度 α_0 有如下关系:

$$ds_0 = r \cdot d\alpha_0 \quad (3)$$

则

$$\begin{aligned} \int_0^{\alpha_0} ds_0 &= \int_0^{\alpha_0} r \cdot d\alpha_0 \\ &= \int_0^{\alpha_0} 16a - \frac{a}{2\pi} \alpha_0 d\alpha_0 \\ &= \int_0^{\alpha_0} 16a \cdot \alpha_0 - \frac{a}{4\pi} \alpha_0^2 \end{aligned} \quad (4)$$

由于 $s_0 > 0$, $\alpha_0 > 0$,

$$\begin{aligned} -\frac{4\pi}{a} + (32\pi)^2 &= \alpha_0^2 - 64\pi\alpha_0 + (32\pi)^2 \\ -\sqrt{(32\pi)^2 - \frac{4\pi}{a} s_0} &= \alpha_0 - 32\pi \quad (\alpha \text{ 关于 } s_0 \text{ 严格单增}) \end{aligned}$$

$$\alpha_0 = 32\pi - \sqrt{(32\pi)^2 - \frac{4\pi}{a} s_0} \quad (5)$$

由于龙头速度为 $1m/s$, 则 $t = s_0$

$$\alpha_0 = 32\pi - \sqrt{(32\pi)^2 - \frac{4\pi}{a} t} \quad (6)$$

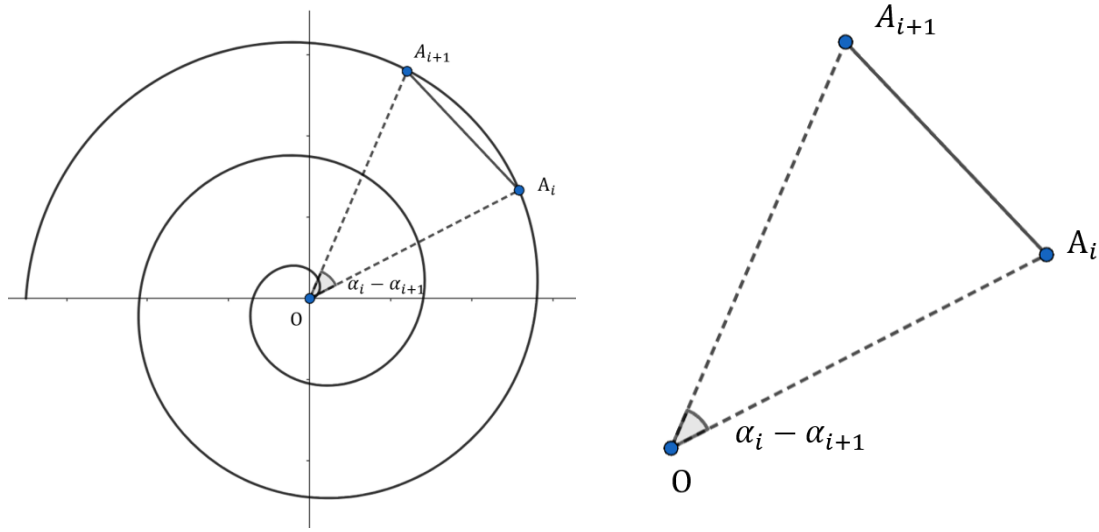


图 1 模型示意图

根据图 1 相邻两点间的三角关系，由余弦定理可得：

$$\begin{aligned} (A_{i+1}A_i)^2 &= r_{i+1}^2 + r_i^2 - 2r_{i+1}r_i \cos(\alpha_i - \alpha_{i+1}) \\ &= r_0^2 + r_1^2 - 2r_1r_0 \cos\left[\frac{2\pi}{a}(r_0 - r_1)\right] \end{aligned} \quad (7)$$

故有模型

$$\left\{ \begin{array}{l} \alpha_0 = 32\pi - \sqrt{(32\pi)^2 - \frac{4\pi}{a}t} \\ r_0 = 16a - \frac{a}{2\pi}\alpha_0 \\ L_1^2 = r_0^2 + r_1^2 - 2r_1r_0 \cos\left[\frac{2\pi}{a}(r_0 - r_1)\right] \\ L_2^2 = r_i^2 + r_{i+1}^2 - 2r_{i+1}r_i \cos\left[\frac{2\pi}{a}(r_i - r_{i+1})\right], i = 1, 2, \dots, 223 \\ \alpha_i = 32\pi - \frac{2\pi}{a}r_i, i = 1, 2, \dots, 223 \\ (x_i, y_i) = (r_i \cos(-\alpha_i), r_i \sin(-\alpha_i)), i = 0, 1, \dots, 223 \end{array} \right. \quad (8)$$

由该模型可以得出 A_i 的 r_i 值，又由公式（1），可得出其 α_i 的值，则根据此模型我们可以计算出每一时刻 A_i 的实时位置 $(r_i \cos(-\alpha_i), r_i \sin(-\alpha_i))$ 。

而由

$$ds_i = -r_i d\alpha_i = \frac{2\pi r_i}{a} dr_i \quad (9)$$

则

$$\frac{dr_i}{dt} = \frac{a}{2\pi r_i} v_i \quad (10)$$

$$\frac{dr_{i+1}}{dt} = \frac{a}{2\pi r_{i+1}} v_{i+1} \quad (11)$$

又将

$$L_1^2 = r_0^2 + r_1^2 - 2r_1r_0 \cos\left[\frac{2\pi}{a}(r_0 - r_1)\right] \quad (12)$$

$$L_2^2 = r_i^2 + r_{i+1}^2 - 2r_{i+1}r_i \cos\left[\frac{2\pi}{a}(r_i - r_{i+1})\right], i = 1, 2 \dots 223 \quad (13)$$

两边对 t 求导可得

$$\begin{aligned} & 2r_i \cdot \frac{dr_i}{dt} + 2r_{i+1} \cdot \frac{dr_{i+1}}{dt} - 2 \frac{dr_i}{dt} \cdot r_{i+1} \cos\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right] - \\ & 2 \frac{dr_{i+1}}{dt} \cdot r_i \cos\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right] + \\ & \frac{4\pi}{a} r_{i+1} r_i \sin\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right] \cdot \left(\frac{dr_{i+1}}{dt} - \frac{dr_i}{dt}\right) = 0 \\ & \frac{a}{\pi} v_i + \frac{a}{\pi} v_{i+1} - \frac{ar_{i+1}}{\pi r_i} \cos\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right] \cdot v_i - \\ & \frac{ar_i}{\pi r_{i+1}} \cos\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right] \cdot v_{i+1} + \\ & 2r_{i+1}r_i \sin\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right] \cdot \left(\frac{v_{i+1}}{r_{i+1}} - \frac{v_i}{r_i}\right) = 0 \\ & \left(1 - \frac{r_{i+1}}{r_i} \cos\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right] - \frac{2\pi r_{i+1}}{a} \sin\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right]\right) v_i + \\ & \left(1 - \frac{r_i}{r_{i+1}} \cos\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right] - \frac{2\pi r_i}{a} \sin\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right]\right) v_{i+1} = 0 \end{aligned} \quad (14)$$

即得 v_i 和 v_{i+1} 的关系式如下

$$v_{i+1} = - \frac{1 - \frac{r_{i+1}}{r_i} \cos\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right] - \frac{2\pi r_{i+1}}{a} \sin\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right]}{1 - \frac{r_i}{r_{i+1}} \cos\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right] - \frac{2\pi r_i}{a} \sin\left[\frac{2\pi}{a}(r_{i+1} - r_i)\right]} v_i \quad (15)$$

由此，我们同样可以通过前面已求得的 r_i ， r_{i+1} 和已知的 v_i 求得该时刻下的速度

v_{i+1} 。

5.1.2 模型的求解

基于前面模型的建立，我们得到了求解问题的代码（附录1），由已知条件：龙头前把手的速度保持 $1m/s$ ，等距螺线螺距为 $0.55m$ ，龙头长 $341cm$ ，龙身及龙尾长 $220cm$ ，由代码求解的结果于 result1.xlsx，其中部分结果如下(表1，表2)：

表1 求解得部分位置结果

	0s	60s	120s	180s	240s	300s
龙头 $x(m)$	8.800000	5.796934	-4.090654	-2.953259	2.578971	4.431365

续表 1

	0s	60s	120s	180s	240s	300s
龙头 $y(m)$	0	-5.773329	-6.300643	6.099638	-5.36395	2.298233
第 1 节龙身 $x(m)$	8.363824	7.455390	-1.452253	-5.229685	4.810833	2.480900
第 1 节龙身 $y(m)$	2.826544	-3.443281	-7.404474	4.368312	-3.575548	4.389951
第 51 节龙身 $x(m)$	-9.518732	-8.685392	-5.537876	2.901017	5.970599	-6.299130
第 51 节龙身 $y(m)$	1.341137	2.543160	6.382430	7.244931	-3.842032	0.490494
第 101 节龙身 $x(m)$	2.913983	5.684500	5.356255	1.887624	-4.930781	-6.224235
第 101 节龙身 $y(m)$	-9.918311	-8.003207	-7.561587	-8.473992	-6.369281	3.956754
第 151 节龙身 $x(m)$	10.861726	6.684757	2.395461	1.016495	2.981425	7.053580
第 151 节龙身 $y(m)$	1.828753	8.132500	9.725700	9.423428	8.393860	4.371872
第 201 节龙身 $x(m)$	4.555102	-6.617079	-10.626274	-9.292372	-7.467919	-5.243026
第 201 节龙身 $y(m)$	10.725118	9.027435	1.366702	-4.236248	-6.167469	-6.419008
龙尾（后） $x(m)$	-5.305444	7.362097	10.974821	7.391955	3.257155	1.809264
龙尾（后） $y(m)$	-10.676584	-8.800018	0.836572	7.484286	9.463651	9.296248

表 2 求解得部分速度结果

	0s	60s	120s	180s	240s	300s
龙头 (m/s)	1	1	1	1	1	1
第 1 节龙身 (m/s)	0.999971	0.999962	0.999946	0.999918	0.999860	0.999712
第 51 节龙身 (m/s)	0.999750	0.999672	0.999552	0.999351	0.998973	0.998122
第 101 节龙身 (m/s)	0.999589	0.999471	0.999292	0.999003	0.998484	0.997384
第 151 节龙身 (m/s)	0.999466	0.999321	0.999107	0.998768	0.998174	0.996958
第 201 节龙身 (m/s)	0.999369	0.999207	0.998969	0.998598	0.997960	0.996681
龙尾（后） (m/s)	0.999333	0.999164	0.998919	0.998537	0.997886	0.996587

由数学分析可知，龙头前把手速度在盘入阶段始终保持最快。

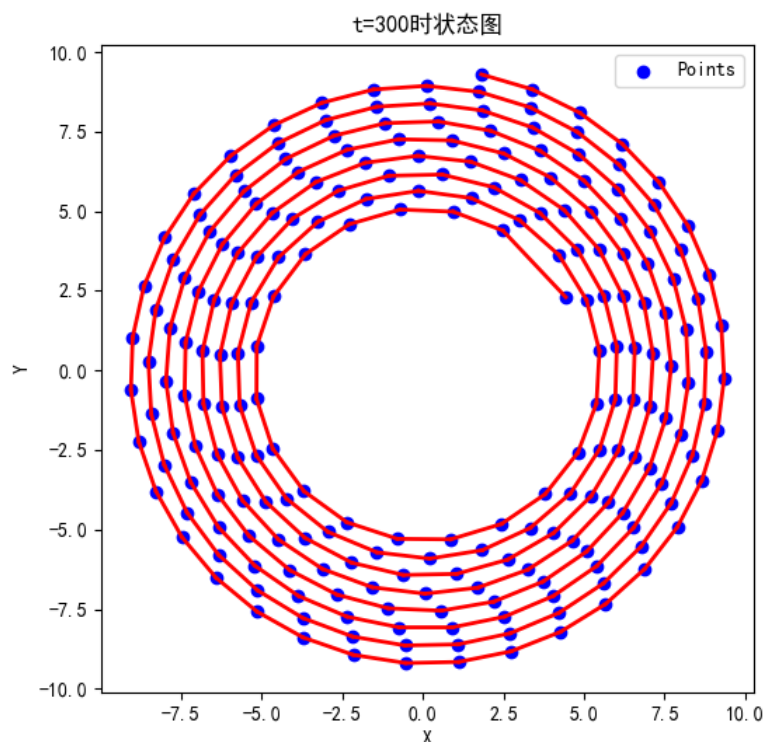


图 2 300s 各把手状态图

5.2 盘龙碰撞检测的建立与求解

在问题二中，要求在舞龙队以问题 1 设定的螺线盘入过程中，确定舞龙队盘入的终止时刻，即使得板凳之间恰好不发生碰撞，并求出此时舞龙队的位置和速度。首先，由于龙身会重复经过龙头所经过的位置，所以仅需考虑龙头与后面龙身相撞情况，根据题目条件，我们可以将盘龙相撞的实际情况**数学转化为龙头左上角的顶点与其中一节龙身的右侧边线共线**，方便问题求解；然后，我们由问题一的盘入模型可以求出每一时刻舞龙队各把手的位置；最后，通过从0s开始模拟盘龙的所有位置并检测盘龙是否碰撞，我们得到了具体的碰撞时间398s，碰撞时刻各把手的位置和速度结果收于 result2.xlsx，具体过程如下。

5.2.1 模型的建立

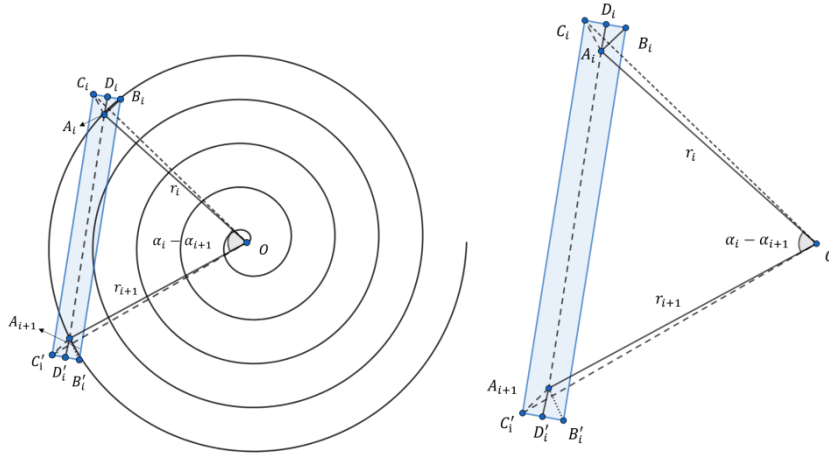


图2 把手与其四角关系示意图

如图所示，由题意知： $A_i D_i = 0.275m$ ， $C_i D_i = 0.15m$ 。设

$$B_i(r_{B_i} \cos(-\beta_{B_i}), r_{B_i} \sin(-\beta_{B_i})) = (x_{i1}, y_{i1}) \quad (16)$$

$$B'_i(x_{i2}, y_{i2}) \quad (17)$$

$$C_0(x_0, y_0) \quad (18)$$

则由板凳对称性和三角函数关系知：

$$d = A_{i+1} C'_i = A_i C_i = A_i B_i = \sqrt{A_i D_i^2 + C_i D_i^2} \quad (19)$$

$$\tan \gamma = \frac{C_i D_i}{A_i D_i} \quad (20)$$

即

$$\gamma = \angle C_i A_i D_i = \angle D_i A_i B_i = \angle C'_i A_{i+1} D'_i = \angle D'_i A_{i+1} B'_i = \arctan \frac{C_i D_i}{A_i D_i} \quad (21)$$

当 $i = 0$ ， $A_i A_{i+1} = L_1 - 2A_i D_i = d_1$ ； 当 $i \neq 0$ ， $A_i A_{i+1} = L_2 - 2A_i D_i = d_2$
则由正弦定理可知：

$$\frac{A_{i+1} A_i}{\sin(\alpha_i - \alpha_{i+1})} = \frac{r_i}{\sin \angle A_i A_{i+1} O} = \frac{r_{i+1}}{\sin \angle A_{i+1} A_i O} \quad (22)$$

由此可得

$$\angle A_i A_{i+1} O = \arcsin \left[\frac{r_i}{A_i A_{i+1}} \sin(\alpha_i - \alpha_{i+1}) \right] \quad (23)$$

$$\angle A_{i+1}A_iO = \arcsin \left[\frac{r_{i+1}}{A_iA_{i+1}} \sin(\alpha_i - \alpha_{i+1}) \right] \quad (24)$$

则

$$\begin{aligned} \angle B'_iA_{i+1}O &= \angle D'_iA_{i+1}O - \angle B'_iA_{i+1}D'_i \\ &= -\gamma + \alpha_i - \alpha_{i+1} + \arcsin \left[\frac{r_i}{d_2} \sin(\alpha_i - \alpha_{i+1}) \right] \\ &= \gamma_{i2} \end{aligned} \quad i = 1, \dots, 223 \quad (25)$$

$$\begin{aligned} \angle B_iA_iO &= \angle D_iA_iO - \angle B_iA_iD_i \\ &= -\gamma + \alpha_i - \alpha_{i+1} + \arcsin \left[\frac{r_{i+1}}{d_2} \sin(\alpha_i - \alpha_{i+1}) \right] \\ &= \gamma_{i1} \end{aligned} \quad i = 1, \dots, 223 \quad (26)$$

$$\begin{aligned} \angle C_0A_0O &= \angle D_0A_0O + \angle C_0A_0D_0 \\ &= \gamma + \alpha_i - \alpha_{i+1} + \arcsin \left[\frac{r_{i+1}}{d_2} \sin(\alpha_i - \alpha_{i+1}) \right] \\ &= \gamma_0 \end{aligned} \quad (27)$$

由余弦定理：

$$\begin{aligned} r_{B'_i} &= \sqrt{d^2 + r_{i+1}^2 - 2dr_{i+1} \cdot \cos(\alpha_i - \alpha_{i+1} - \gamma + \arcsin \left[\frac{r_i}{d_2} \sin(\alpha_i - \alpha_{i+1}) \right])} \end{aligned} \quad (28)$$

$$r_{B_i} = \sqrt{d^2 + r_i^2 - 2dr_i \cdot \cos(\alpha_i - \alpha_{i+1} - \gamma + \arcsin \left[\frac{r_{i+1}}{d_2} \sin(\alpha_i - \alpha_{i+1}) \right])} \quad (29)$$

$$r_{C_0} = \sqrt{d^2 + r_0^2 - 2dr_0 \cdot \cos(\alpha_0 - \alpha_1 - \gamma + \arcsin \left[\frac{r_i}{d_1} \sin(\alpha_0 - \alpha_1) \right])} \quad (30)$$

由正弦定理：

$$\sin \angle A_0 O C_0 = \frac{d}{r_{C_0}} \sin \angle C_0 A_0 O$$

$$\sin \angle A_i O B_i = \frac{d}{r_{B_i}} \sin \angle B_i A_i O$$

$$\sin \angle A_{i+1} O B'_i = \frac{d}{r_{B'_i}} \sin \angle B'_i A_{i+1} O$$

可得

$$\alpha_{B_i} = \alpha_i + \arcsin\left(\frac{d}{r_{B_i}} \sin \gamma_{i1}\right) \quad (31)$$

$$\alpha_{B'_i} = \alpha_{i+1} + \arcsin\left(\frac{d}{r_{B'_i}} \sin \gamma_{i2}\right) \quad (32)$$

$$\alpha_{C_0} = \alpha_0 + \arcsin\left(\frac{d}{r_0} \sin \gamma_0\right) \quad (33)$$

因为 C_0 位于线段 $B_i B'_i$ 之间

$$S_{\Delta B_i C_0 B'_i} = \frac{1}{2} |x_0(y_{i1} - y_{i2}) + x_{i1}(y_{i2} - y_0) + x_{i2}(y_0 - y_{i1})| = 0$$

即

$$x_0(y_{i1} - y_{i2}) + x_{i1}(y_{i2} - y_0) + x_{i2}(y_0 - y_{i1}) = 0$$

且

$$(x_0 - x_{i1})(x_0 - x_{i2}) \leq 0$$

得到

$$x_0(y_{i1} - y_{i2}) + x_{i1}(y_{i2} - y_0) + x_{i2}(y_0 - y_{i1}) = 0 \quad (34)$$

$$(x_0 - x_{i1})(x_0 - x_{i2}) \leq 0 \quad (35)$$

则由方程组

$$\left\{ \begin{array}{l} B_i(x_{i1}, y_{i1}) = (r_i \cos(-\alpha_{B_i}), r_i \sin(-\alpha_{B_i})) \\ B'_i(x_{i2}, y_{i2}) = (r_i \cos(-\alpha_{B'_i}), r_i \sin(-\alpha_{B'_i})) \\ C_0(x_0, y_0) = (r_i \cos(-\alpha_{C_0}), r_i \sin(-\alpha_{C_0})) \\ \gamma_{i1} = -\gamma + \alpha_i - \alpha_{i+1} + \arcsin \left[\frac{r_{i+1}}{d_2} \sin(\alpha_i - \alpha_{i+1}) \right] \\ \gamma_{i2} = -\gamma + \alpha_i - \alpha_{i+1} + \arcsin \left[\frac{r_i}{d_2} \sin(\alpha_i - \alpha_{i+1}) \right] \\ \gamma_0 = \gamma + \alpha_i - \alpha_{i+1} + \arcsin \left[\frac{r_{i+1}}{d_2} \sin(\alpha_i - \alpha_{i+1}) \right] \\ r_{B'_i} = \sqrt{d^2 + r_{i+1}^2 - 2dr_{i+1} \cdot \cos \gamma_{i2}} \\ r_{B_i} = \sqrt{d^2 + r_i^2 - 2dr_i \cdot \cos \gamma_{i1}} \\ r_{C_0} = \sqrt{d^2 + r_0^2 - 2dr_0 \cdot \cos \gamma_0} \\ \alpha_{B_i} = \alpha_i + \arcsin \left(\frac{d}{r_{B_i}} \sin \gamma_{i1} \right) \\ \alpha_{B'_i} = \alpha_{i+1} + \arcsin \left(\frac{d}{r_{B'_i}} \sin \gamma_{i2} \right) \\ \alpha_{C_0} = \alpha_0 + \arcsin \left(\frac{d}{r_0} \sin \gamma_0 \right) \\ x_0(y_{i1} - y_{i2}) + x_{i1}(y_{i2} - y_0) + x_{i2}(y_0 - y_{i1}) = 0 \\ (x_0 - x_{i1})(x_0 - x_{i2}) \leq 0 \end{array} \right.$$

可以计算得到结果判断是否龙头会和第*i*节龙身发生碰撞。

5.2.2 模型求解

通过把题目提供的数据带入到问题一建立的模型里，我们可以直接求得每一时刻各个把手的位置信息，再将位置信息代入到问题二的模型，我们判断各个时刻下，龙头有没有碰撞某一龙身，从而计算出龙头的终止时间。

通过模型我们得到了终止时刻的龙头及龙身的位置与速度如结果 result2.xlsx 中，其中部分结果如下：

表 3 龙头及龙身终止时刻位置信息

	398s
龙头 $x(m)$	2.584109
龙头 $y(m)$	1.034044
第 1 节龙身 $x(m)$	0.375742
第 1 节龙身 $y(m)$	2.851381
第 51 节龙身 $x(m)$	-1.841626
第 51 节龙身 $y(m)$	-4.408672
第 101 节龙身 $x(m)$	4.753095
第 101 节龙身 $y(m)$	3.838445

续表 3

第 151 节龙身 $x(m)$	6.160724
第 151 节龙身 $y(m)$	3.721621

第 201 节龙身 $x(m)$	2.601379
第 201 节龙身 $y(m)$	-7.714158
龙尾（后） $x(m)$	-8.520586
龙尾（后） $y(m)$	0.200369

表 4 龙头及龙身终止时刻速度信息

	398s
龙头 (m/s)	1
第 1 节龙身 (m/s)	0.996576
第 51 节龙身 (m/s)	0.987806
第 101 节龙身 (m/s)	0.985930
第 151 节龙身 (m/s)	0.985111
第 201 节龙身 (m/s)	0.984652
龙尾（后） (m/s)	0.984508

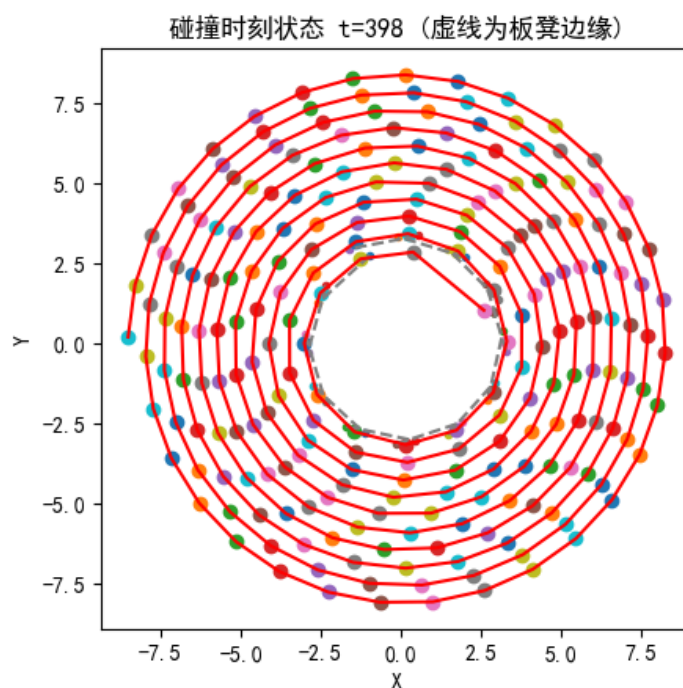


图 3 碰撞时刻各把手状态示意图

5.3 最小螺距模型的求解

在问题三中，舞龙队为了从盘入切换为盘出会提前留出一片调头区域，要求在已知调头空间为一个以螺线中心为圆心、直径为 $9m$ 的圆形区域的前提下，建立计算龙头前把手能够进入调头空间的最小螺距模型。首先，我们基于问题二的模型可以知道在给定螺距为 a 时，发生碰撞的时间节点；然后，我们求解碰撞时龙头前把手的极径 r_0 ，比较 r_0 和 $4.5m$ 之间的大小，如果小于，那么 a 减去一个步长再重复前面的步骤计

算，继续比较直到 r_0 大于 $4.5m$ ；之后，我们利用二分法，将 a 回退一个步长，将步长减半再重复前面的检验步骤，直到循环 10 次结束或者 r_0 大于 $4.5m$ ，我们得到题目所需的最小螺距为 $0.3312m$ ，具体过程如下。

5.3.1 基本假设

考虑到螺距越小时越容易发生碰撞，在本问题中我们假设螺距具有一个临界值 a ，当螺距小于 a 时在进入掉头空间之前会发生碰撞，当螺距大于 a 时不会发生碰撞。

5.3.2 问题三求解

调头空间是以螺线中心为圆心、直径为 $9m$ 的圆形区域，如下图 4 所示。

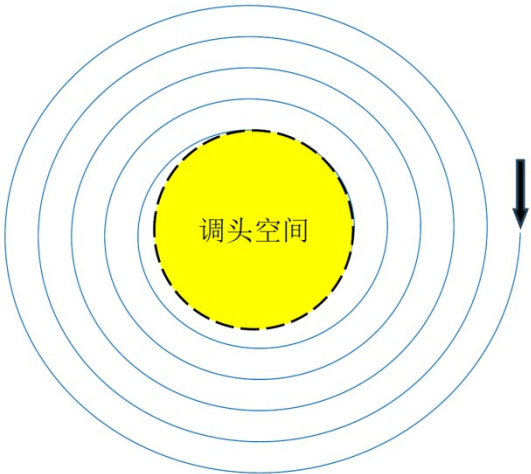


图 4 调头区域示意图

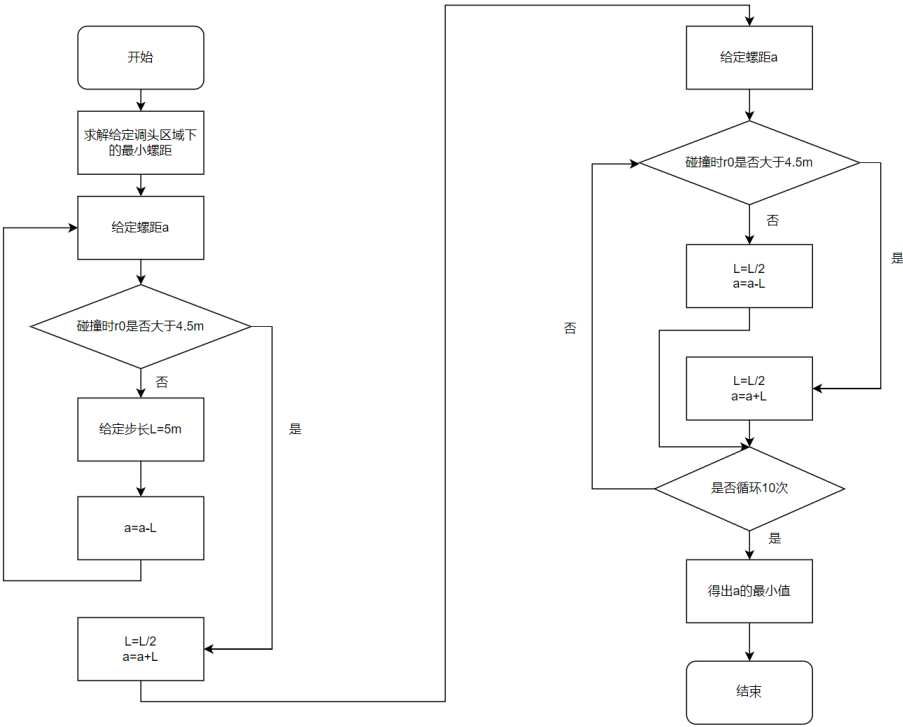


图 5 问题三流程图

对于问题三它实际上是个优化问题，我们基于二分法的思想对最小螺距进行了

逐步逼近求解，其步骤如下：

1. 以螺距 $a = 100m$ ，步长 $l = 5m$ ，对于给定螺距 a ，通过问题二的碰撞检测模型，求解碰撞时 r_0 的值。
2. 当碰撞时刻，龙头前把手的极径 $r_0 \leq 4.5m$ 或完全不碰撞，那么将螺距 a 减去步长 l ，将新的 a 重新代入碰撞检测模型。如此循环，直到碰撞时刻，龙头前把手的极径 $r_0 > 4.5m$ ，退出循环，进入下一循环。
3. 进入循环时，我们利用二分法，先规定循环最多进行 10 次，即进行 10 次二分法逼近，同时我们将步长 l 自减半，将螺距 a 加一个步长 l ，继续将新的 a 代入碰撞检测模型计算，此时如果龙头前把手的极径 $r_0 \leq 4.5m$ ，那么我们就将步长 l 减半，螺距减少 l ，循环次数+1；如果龙头前把手的极径 $r_0 > 4.5m$ ，那么我们就将步长 l 减半，螺距增加 l ，循环次数不变。
4. 直到循环 10 次之后，我们得到 a 的最小值，程序结束。
最终我们得到最小螺距 $0.3312m$ 。

5.4 调头曲线不变的盘入转盘出模型

在问题四中，要求在已知盘入螺线螺距为 $1.7m$ ，盘出螺线与盘入螺线中心对称，舞龙队的调头路径是由两段相切圆弧连接而成的S形曲线，前一段圆弧为后一段圆弧的2倍，且该曲线与盘入均相切的前提下，寻找圆弧的调整方案，使得调头曲线变短，并求出以调头开始为零时刻，龙头前把手速度保持 $1m/s$ 时，零时刻前后100s内的各把手位置和速度。首先，我们根据等距螺线和调头路径的圆与圆之间的几何性质，可以由简单的几何知识推断出，调头曲线的长度在题目条件下是不变的，即无法使其变短；然后，我们可以通过调头空间与等距螺线相交的特点，求出其相交点，并由坐标原点到该点的向量与等距螺线在该点的切向向量内积，求得该交点切线与坐标轴的夹角关系；其次，我们转动坐标轴，使得 y 轴与盘入螺线与盘出螺线的切线延长线和切线的垂线延长所组成的长方形的对角线重合；最后我们通过圆的几何知识，可以求出两圆的半径和方程，再由此算出龙头前把手位置关于时间的函数和把手位置之间的递推关系，代入具体参数求得结果于 result4.xlsx，具体操作如下。

5.4.1 盘入、盘出螺线与调头空间的交点求解

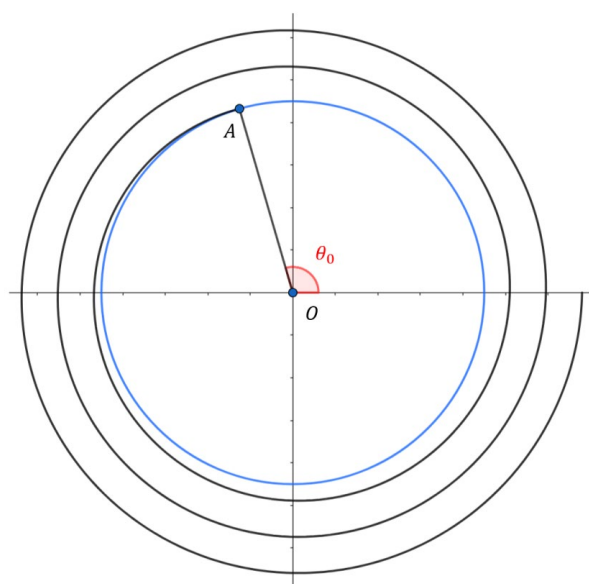


图 6 盘入螺线与调头空间的交点示意图

如图 6 所示，设盘入螺线与调头曲线的交点为 A ， $\angle AOx$ 设为角 θ_0 ，因为等距螺线与调头空间相交为 A 点，所以 A 此时的极径为 $r_0 = 4.5m$ ，角度为 $\alpha_0 = 32\pi - \frac{2\pi}{a}r_0$ ，由 r_0 即可求得 α_0 。再由 $-\alpha_0 = 2k\pi + \theta_0 (-\pi < \theta_0 \leq \pi, k \in Z)$ ，可得坐标系旋转 $\omega = \theta_0 - \frac{\pi}{2}$ 。

5.4.2 调头曲线长度不变的证明

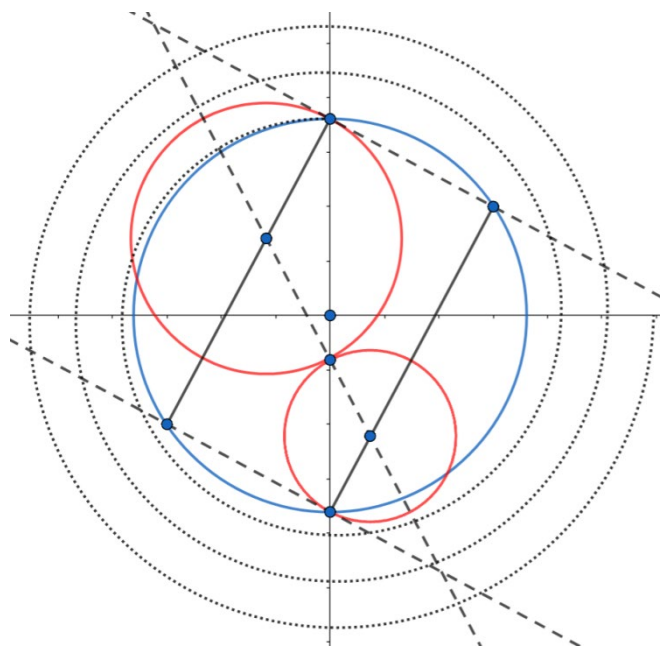


图 7 调头曲线及切线垂线组成的长方形示意图

（注意：此处切入角 θ' 刻意画大，通过后面计算得仅偏离圆弧切线 3° 左右）

如图 7 所示，根据问题四给出的题目条件：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，它与盘入、盘出螺线均相切，盘出螺线与盘入螺线呈中心对称，再由 5.4.1 所进行的坐标系旋转，我们可以得到如图 5 所示的两圆以及调头路径与盘入、盘出螺线交点处的切线和垂线组成的长方形。由该长方形，我们可以得到证明调头曲线不变的数学模型如下图 7（如下所示）。

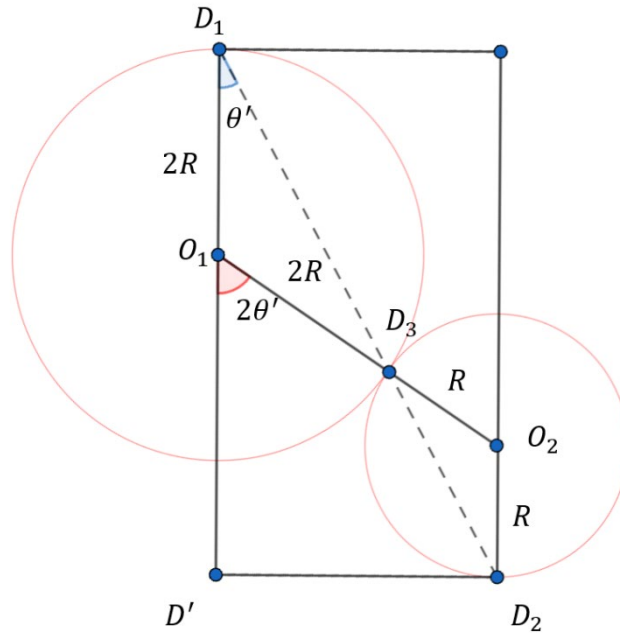


图 9 证明示意图

由 5.4.2 可知，无论如何调整两圆弧半径，调头曲线长度都不变，故此令 $k = 2$ 进行 5.4.3 的证明。

连接 O_1D_3 ，令 $\angle O_1D_1D_2 = \theta'$ 。由于 $D_1D_3 = 4R \cos \theta'$ ， $D_2D_3 = 2R \cos \theta'$ ，而

$$\begin{aligned}
 D_1D_2 &= \sqrt{D_1D'^2 + D'D_2^2} \\
 &= \sqrt{[3R(1 + \cos 2\theta')]^2 + (3R \sin 2\theta')^2} \\
 &= 6R\sqrt{\cos^4 \theta' + \sin^2 \theta' \cos^2 \theta'} \\
 &= 6R \cos \theta' \\
 &= 4R \cos \theta' + 2R \cos \theta' \\
 &= D_1D_3 + D_2D_3
 \end{aligned} \tag{38}$$

则 D_1, D_2, D_3 三点共线，加上已知的 D_1, D_2, O 三点共线，则 D_1, D_2, D_3, O 四点共线，的证。

5.4.4 求解 A_0 调头曲线的轨迹方程

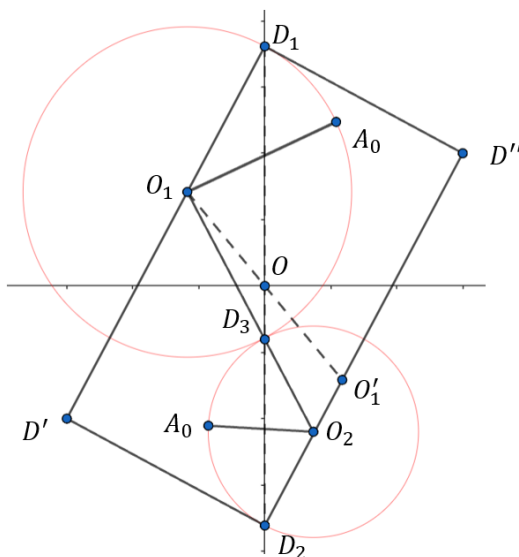


图 10 求解 A_0 调头曲线的轨迹方程示意图

为求解轨迹方程式作如下准备:

$$(X, Y) = -(\cos \frac{2\pi R_0}{a} - \frac{2\pi R_0}{a} \sin \frac{2\pi R_0}{a}, \sin \frac{2\pi R_0}{a} + \frac{2\pi R_0}{a} \cos \frac{2\pi R_0}{a}) \quad (39)$$

$$(x_0, y_0) = (R_0 \cos \frac{2\pi R_0}{a}, R_0 \sin \frac{2\pi R_0}{a}) \quad (40)$$

$$\cos \angle D_2 D_1 D'' = \frac{X \cdot x_0 + Y \cdot y_0}{\sqrt{X^2 + Y^2} \cdot \sqrt{x_0^2 + y_0^2}} \quad (41)$$

$$\cos \theta' = \sqrt{1 - \cos^2 \angle D_2 D_1 D''} \quad (42)$$

$$R_1 = \frac{\frac{2}{3}R_0}{\cos \theta'} \quad (43)$$

$$R_2 = \frac{\frac{1}{3}R_0}{\cos \theta'} \quad (44)$$

由余弦定理

$$r_{O_1} = \sqrt{R_1^2 + R_0^2 - 2R_0R_1 \cos \theta'} \quad (45)$$

$$r_{O_2} = \sqrt{R_2^2 + R_0^2 - 2R_0R_2 \cos \theta'} \quad (46)$$

由正弦定理

$$\theta_{O_1} = \frac{\pi}{2} + \arcsin\left(\frac{R_1}{r_{O_1}} \cos \angle D_2 D_1 D''\right) \quad (47)$$

$$\theta_{O_2} = -\frac{\pi}{2} + \arcsin\left(\frac{R_2}{r_{O_2}} \cos \angle D_2 D_1 D''\right) \quad (48)$$

故圆 O_1 、圆 O_2 有方程

$$\begin{cases} r^2 - 2rr_{O_1} \cos(\theta - \theta_{O_1}) + r_{O_1}^2 = R_1^2 \\ r^2 - 2rr_{O_2} \cos(\theta - \theta_{O_2}) + r_{O_2}^2 = R_2^2 \end{cases} \quad (49)$$

(说明：为方便标识点的所在曲线以及方便计算，下面用 α ， β ， θ 三个变量分别用于在进入圆之前、圆之中、出圆之后表示角度，且每个过程另外两个变量会赋值为0)

切入螺线 $r = \rho_1 + \frac{a}{2\pi}\alpha$ 经过 $(R_0, \frac{\pi}{2})$ ，则方程为 $r = R_0 - \frac{a}{4} + \frac{a}{2\pi}\alpha$ 令 $\beta = 0$ ，即

$$\alpha = \frac{2\pi}{a}(r - R_0 + \frac{a}{4}) \quad (50)$$

切出螺线 $r = \rho_2 + \frac{a}{2\pi}\beta$ 经过 $(R_0, -\frac{\pi}{2})$ ，则方程为 $r = R_0 - \frac{a}{4} + \frac{a}{2\pi}\beta$ 令 $\alpha = 0$ ，即

$$\beta = \frac{2\pi}{a}(r - R_0 + \frac{a}{4}) \quad (51)$$

如下求解 A_0 位置：

切入切出螺线：

根据(12)(13)式，求解过程和问题一中类似，便不再赘述

$$\begin{cases} r_{A_0} = \sqrt{R_0^2 - \frac{at}{\pi}}, & t \leq 0 \\ r_{A_0} = \sqrt{R_0^2 - \frac{a}{\pi}[t - (R_1 + R_2)(\pi - 2\theta')]}, & t > (R_1 + R_2)(\pi - 2\theta') \end{cases} \quad (52)$$

前圆弧：

由于弧 $DA_1 = R_1 \angle D_1 O_1 A_0$ ， A_0 速度为 $1m/s$ ，则

$$\angle D_1 O_1 A_0 = \frac{t}{R_1}, (0 \leq t \leq R_1(\pi - 2\theta')) \quad (53)$$

$$\angle D_1 O_1 O = \arccos\left(\frac{r_{O_1}^2 + R_1^2 - R_0^2}{2r_0 R_1}\right) \quad (54)$$

$$\angle O O_1 A_0 = |\angle D_1 O_1 O - \angle D_1 O_1 A| = \left| \arccos\frac{r_{O_1}^2 + R_1^2 - R_0^2}{2r_0 R_1} - \frac{t}{R} \right| \quad (55)$$

$$r_{A_0} = \sqrt{r_{O_1}^2 + R_1^2 - 2r_0R_1 \cdot \cos \angle OO_1A_0} \quad (0 \leq t \leq (\pi - 2\theta')R_1) \quad (56)$$

$$\theta_{A_0} = \begin{cases} \theta_{O_1} - \arccos \frac{r_{A_0}^2 + r_{O_1}^2 - R_1^2}{2r_{A_0}r_1}, 0 \leq t \leq R_1 - \arccos \frac{r_{A_0}^2 + r_{O_1}^2 - R_1^2}{2r_{A_0}r_1} \\ \theta_{O_1} + \arccos \frac{r_{A_0}^2 + r_{O_1}^2 - R_1^2}{2r_{A_0}r_1} - 2\pi, R_1 - \arccos \frac{r_{A_0}^2 + r_{O_1}^2 - R_1^2}{2r_{A_0}r_1} < t \leq R_1(\pi - 2\theta') \end{cases} \quad (57)$$

后圆弧:

$$\angle A_0O_2D_2 = \pi - 2\theta' - \frac{t - (\pi - 2\theta')R_1}{R_2} = \left(1 + \frac{R_1}{R_2}\right)(\pi - 2\theta') - \frac{t}{R_2} \quad (58)$$

$$(R_1(\pi - 2\theta') \leq t \leq (R_1 + R_2)(\pi - 2\theta'))$$

$$\angle D_2O_2O = \arccos \frac{r_{O_2}^2 + R_2^2 - R_0^2}{2r_{O_2}R_2} \quad (59)$$

$$\angle OO_2A_0 = \angle A_0O_2D_2 = \arccos \frac{r_{O_2}^2 + R_2^2 - R_0^2}{2r_{O_2}R_2} + \frac{t}{R_2} - \left(1 + \frac{R_1}{R_2}\right)(\pi - 2\theta') \quad (60)$$

$$r_{A_0} = \sqrt{r_{O_2}^2 + R_1^2 - 2r_{O_2}R_1 \cdot \cos \angle OO_2A_0} \quad (61)$$

$$\theta_{A_0} = \theta_{O_2} - \arccos \frac{r_{A_0}^2 + r_{O_2}^2 - R_2^2}{2r_{A_0}r_{O_2}} \quad (62)$$

则前后圆弧分别可以用上述对应公式计算。

5.4.5 求解后把手 A_{i+1} 的位置

对于后把手 A_{i+1} 的位置有四种情况: 1)切入螺线 2)前圆弧 3)后圆弧 4)切出螺线

考虑到 2)中有一个 r_{i+1} 值可能对应两个 α_{i+1} , 故 2)分为 O'_1 前后两种情况, 共五种情况, 其示意图如下图 11。

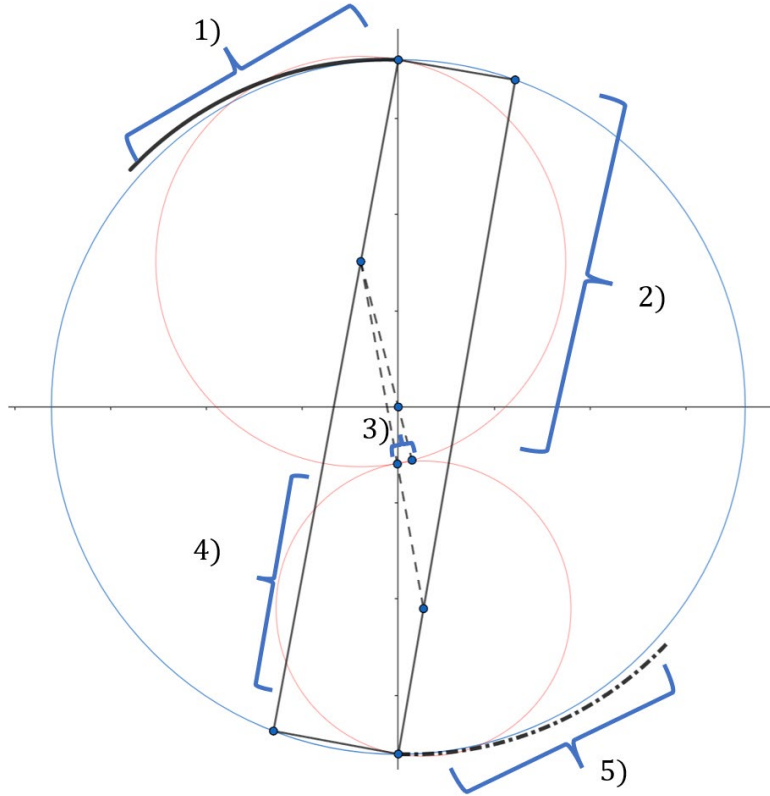


图 11 A_{i+1} 不同情况示意图

现假设1)时角度用 α 表示，2) 3) 4)时用角度 θ 表示，5)时角度用 β 表示。

$$1): \alpha = \frac{2\pi}{a} \left(r - R_0 + \frac{a}{4} \right) \quad (63)$$

$$2): \theta = \theta_{O_1} - \arccos \frac{r^2 + r_{O_1}^2 - R_1^2}{2rr_{O_1}} \quad (64)$$

$$3): \theta = \theta_{O_1} + \arccos \frac{r^2 + r_{O_1}^2 - R_1^2}{2rr_{O_1}} - 2\pi \quad (65)$$

$$4): \theta = \theta_{O_2} + \arccos \frac{r^2 + r_{O_2}^2 - R_2^2}{2rr_{O_1}} \quad (66)$$

$$5): \beta = \frac{2\pi}{a} \left(r - R_0 - \frac{a}{4} \right) \quad (67)$$

A_{i+1} 五种情况对应四个临界点: $(R_0, 0)$ 、 $(r_{O_1}, \theta_{O_1} - \pi)$ 、 $(-\frac{R_0}{3}, -\frac{\pi}{2})$ 、 $(-R_0, -\frac{\pi}{2})$ 。

此时 A_i 对应临界点:

$$R_0^2 + r_1^2 - 2R_0r_1' \cos \theta_1' = L^2, \text{ 结合2)式得}(r_1', \theta_1') \quad (68)$$

$$r_{O_1}^2 + r_2'^2 - 2r_0r_2' \cos(\theta_2' - \theta_{O_1} + \pi) = L^2, \text{ 结合4)式得}(r_2', \theta_2') \quad (69)$$

$$(\frac{R_0}{3})^2 + r_3'^2 - 2\frac{R_0}{3}r_3' \cos(\theta_3' + \frac{\pi}{2}) = L^2, \text{ 结合4)式得}(r_3', \theta_3') \quad (70)$$

$$R_0^2 + r_4'^2 - 2R_0r_4' \cos(\theta_4' + \frac{\pi}{2}) = L^2, \text{ 结合5)式得}(r_4', \theta_4') \quad (71)$$

对应不同的 L_1/L_2 ，得不同 (r_{i1}', θ_{i1}') ， (r_{i2}', θ_{i2}') ， $i = 1, 2, 3, 4$

5.4.6 递推求解 A_{i+1} 流程图

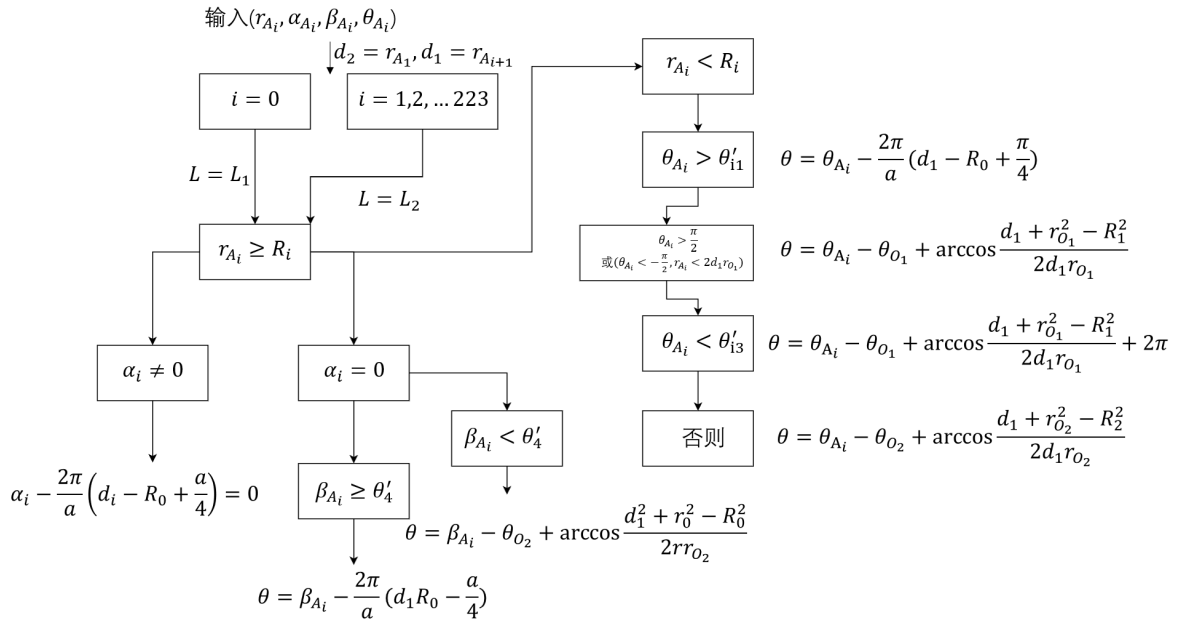


图 12 求解流程图

函数参数 d_1 、 d_2 、 θ 、 L ，求解

$$d_1^2 + d_2^2 - 2d_1d_2 \cos \theta = L^2 \quad (72)$$

其中 d_1 未知， d_2 、 L 已知， θ 可表示为 d_1 函数，可返回 d_1 。

5.4.7 速度的求解

对于确定位置的把手，其速度只需要采用逼近的方法即可求解，当我们计算某一点的速度时，只需要令求一个步长为 $\Delta t = 0.001s$ 之后的该把手的位置，通过两个位置即可求出过程中运动的距离，再除以步长，即可也平均速度逼近瞬时速度，易知，只要步长设置的足够小，即可将平均速度当作瞬时速度。

5.4.8 模型的求解结果

根据题目参数代入值得到结果于 result4.xlsx，其中部分结果如下：

表 5

	-100s	-50s	0s	50s	100s
龙头 $x(m)$					

续表 5

	-100s	-50s	0s	50s	100s
龙头 $y(m)$					
第 1 节龙身 $x(m)$					
第 1 节龙身 $y(m)$					
第 51 节龙身 $x(m)$					
第 51 节龙身 $y(m)$					
第 101 节龙身 $x(m)$					
第 101 节龙身 $y(m)$					
第 151 节龙身 $x(m)$					
第 151 节龙身 $y(m)$					
第 201 节龙身 $x(m)$					
第 201 节龙身 $y(m)$					
龙尾（后） $x(m)$					
龙尾（后） $y(m)$					

表 6

	-100s	-50s	0s	50s	100s
龙头 (m/s)	1	1	1	1	1
第 1 节龙身 (m/s)					
第 51 节龙身 (m/s)					
第 101 节龙身 (m/s)					
第 151 节龙身 (m/s)					
第 201 节龙身 (m/s)					
龙尾（后） (m/s)					

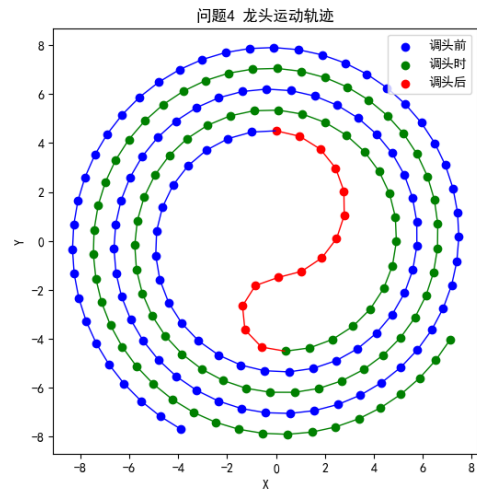


图 13 龙头运动轨迹示意图

5.5 龙头最大速度模型

在问题五中，基于问题四的路径行进，我们需要在龙头行进速度保持不变的情况下，确定龙头的最大行进速度为多少时，舞龙队各把手的速度都不超过 $2m/s$ 。首先，我们假设龙尾后把手盘入结束后为速度最快的把手，通过问题四的模型计算了调头和盘出状态时，不同时刻龙尾后把手的位置；其次，我们通过将两相邻位置求路程，从中找到最大路程的时刻，即为龙尾的最快速度的时刻；然后，我们通过假设龙尾后把手最快速度为 $2m/s$ ，通过它倒推得到龙头前把手的此时可能的 n 个速度，最后将这 n 个可能速度求平均减少误差，得到的即为龙头的最大行进速度，最大行进速度为 $1.957m/s$ ，具体过程如下。

5.5.1 模型的建立

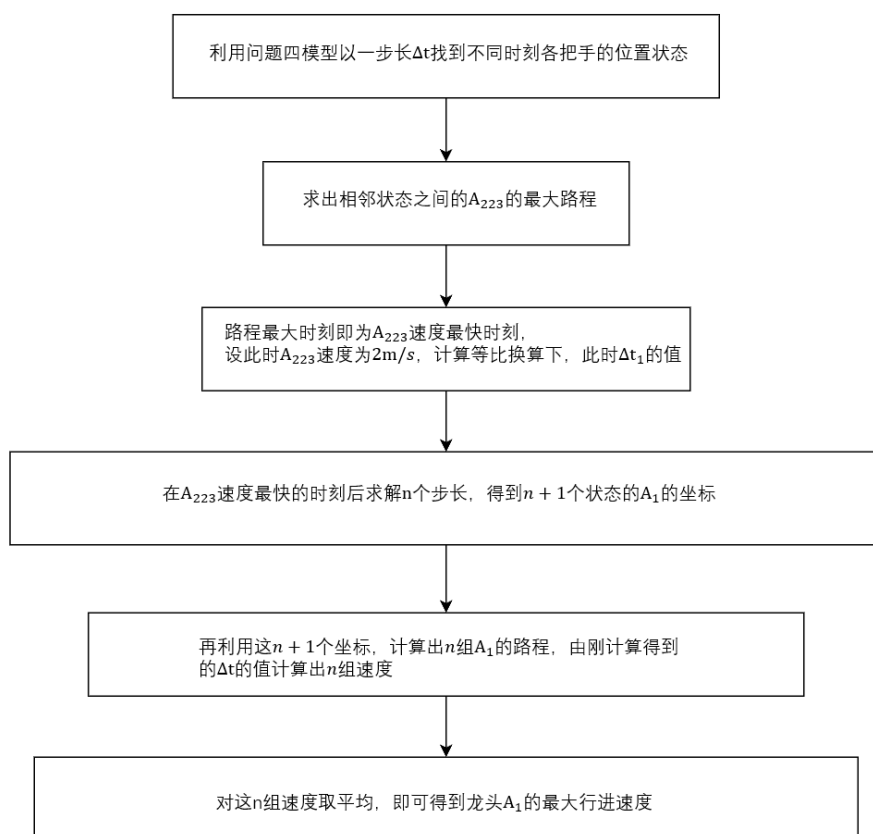


图 14 龙头最大速度求解流程图

由问题 1 我们已知，在舞龙队在盘入的状态时，无论何时，龙头前把手都保持速度最大，如果要考虑舞龙队各把手的速度都不超过 $2m/s$ ，此时我们可以让龙头的行进速度保持为 $2m/s$ 即可。而处于调头或盘出状态时则不然，龙头此时不是行进速度最快的把手，为了使舞龙队各把手的速度都不超过 $2m/s$ ，我们则要将此时舞龙队速度最快的把手速度设置为 $2m/s$ ，从该把手推出此时龙头的速度，那么此时龙头的速度即为舞龙队龙头的最大行进速度。这里为了方便求解，我们假设龙尾后把手(A_{223})为调头或盘出状态的速度最大的把手，以此假设为基础得到求解龙头最大速度的流程图如图 12 所示。问题五龙头最大速度的求解主要有以下流程：

1. 定一足够小的步长 Δt ，利用问题四的模型得到调头及盘出状态时，不同时刻 A_1 及 A_{223} 的坐标。

2. 求解 A_{223} 相邻坐标之间的路程，找到路程最大的时刻，此时刻即为 A_{223} 速度最快的时刻，根据假设，我们设此时 A_{223} 的速度为 $2m/s$ ，利用最大路程求解得到一新步长 Δt_1 。
3. 在 A_{223} 速度最快的时刻后，取得 n 个步长得到的 $n + 1$ 个不同的 A_1 的坐标。
4. 求出 A_1 相邻坐标之间的路程，得到 n 组路程，将其分别与 Δt_1 相除得到 n 组不同的速度。
5. 对这 n 组速度取平均，得到的即是龙头前把手 A_1 的最大行进速度。

5.5.2 模型的求解

根据问题四的模型，我们代入题目所给参数进行求解，得到龙头的最大行进速度为 $??m/s$ 。





六、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.

七、附录

第一部分：支撑材料

附录 1	
源代码	 A_1.py  A_2.py  A_3.py  A_4.py

数据	 result1.xlsx  result2.xlsx  result4.xlsx
图片	 图片.pptx

第二部分：问题结果表格及可视化图。

问题一：

表 1

	0s	60s	120s	180s	240s	300s
龙头 $x(m)$	8.800000	5.796934	-4.090654	-2.953259	2.578971	4.431365
龙头 $y(m)$	0	-5.773329	-6.300643	6.099638	-5.36395	2.298233
第 1 节龙身 $x(m)$	8.363824	7.455390	-1.452253	-5.229685	4.810833	2.480900
第 1 节龙身 $y(m)$	2.826544	-3.443281	-7.404474	4.368312	-3.575548	4.389951
第 51 节龙身 $x(m)$	-9.518732	-8.685392	-5.537876	2.901017	5.970599	-6.299130

第 51 节 龙身 $y(m)$	1.341137	2.543160	6.382430	7.244931	-3.842032	0.490494
第 101 节 龙身 $x(m)$	2.913983	5.684500	5.356255	1.887624	-4.930781	-6.224235
第 101 节 龙身 $y(m)$	-9.918311	-8.003207	-7.561587	-8.473992	-6.369281	3.956754
第 151 节 龙身 $x(m)$	10.861726	6.684757	2.395461	1.016495	2.981425	7.053580
第 151 节 龙身 $y(m)$	1.828753	8.132500	9.725700	9.423428	8.393860	4.371872
第 201 节 龙身 $x(m)$	4.555102	-6.617079	-10.626274	-9.292372	-7.467919	-5.243026
第 201 节 龙身 $y(m)$	10.725118	9.027435	1.366702	-4.236248	-6.167469	-6.419008

续表 1

	0s	60s	120s	180s	240s	300s
龙尾 (后) $x(m)$	-5.305444	7.362097	10.974821	7.391955	3.257155	1.809264
龙尾 (后) $y(m)$	-10.676584	-8.800018	0.836572	7.484286	9.463651	9.296248

表 2

	0s	60s	120s	180s	240s	300s
龙头 (m/s)	1	1	1	1	1	1
第 1 节龙身 (m/s)	0.999971	0.999962	0.999946	0.999918	0.999860	0.999712
第 51 节龙身 (m/s)	0.999750	0.999672	0.999552	0.999351	0.998973	0.998122
第 101 节龙身 (m/s)	0.999589	0.999471	0.999292	0.999003	0.998484	0.997384
第 151 节龙身 (m/s)	0.999466	0.999321	0.999107	0.998768	0.998174	0.996958
第 201 节龙身 (m/s)	0.999369	0.999207	0.998969	0.998598	0.997960	0.996681
龙尾 (后) (m/s)	0.999333	0.999164	0.998919	0.998537	0.997886	0.996587

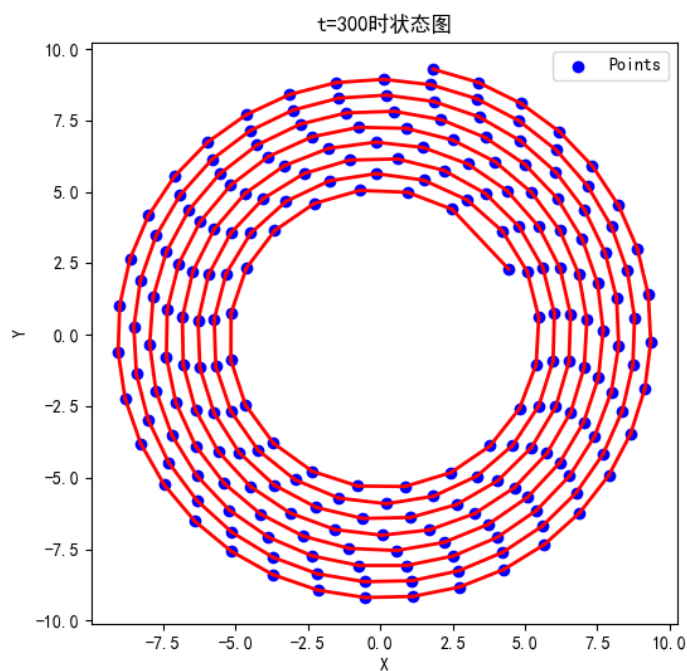


图 15 300s时各把手状态示意图

问题二:

表 3

	398s
龙头 $x(m)$	2.584109
龙头 $y(m)$	1.034044
第 1 节龙身 $x(m)$	0.375742
第 1 节龙身 $y(m)$	2.851381
第 51 节龙身 $x(m)$	-1.841626
第 51 节龙身 $y(m)$	-4.408672
第 101 节龙身 $x(m)$	4.753095
第 101 节龙身 $y(m)$	3.838445
第 151 节龙身 $x(m)$	6.160724
第 151 节龙身 $y(m)$	3.721621
第 201 节龙身 $x(m)$	2.601379
第 201 节龙身 $y(m)$	-7.714158
龙尾（后） $x(m)$	-8.520586
龙尾（后） $y(m)$	0.200369

表 4

	398s
龙头 (m/s)	1

第 1 节龙身(m/s)	0.996576
第 51 节龙身(m/s)	0.987806
第 101 节龙身(m/s)	0.985930
第 151 节龙身(m/s)	0.985111
第 201 节龙身(m/s)	0.984652
龙尾（后）(m/s)	0.984508

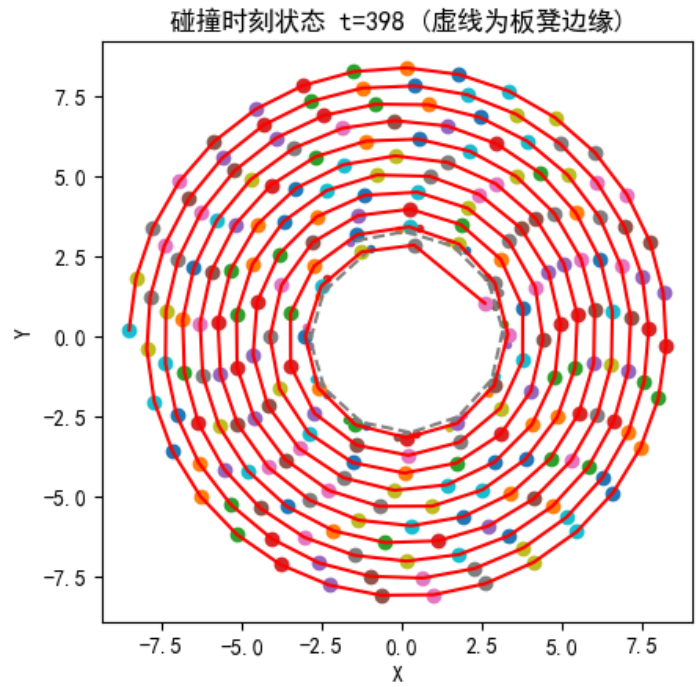


图 16 碰撞时刻各把手状态示意图

问题四:

表 5

	-100s	-50s	0s	50s	100s
龙头 $x(m)$					
龙头 $y(m)$					
第 1 节龙身 $x(m)$					
第 1 节龙身 $y(m)$					
第 51 节龙身 $x(m)$					
第 51 节龙身 $y(m)$					
第 101 节龙身 $x(m)$					
第 101 节龙身 $y(m)$					
第 151 节龙身 $x(m)$					

续表 5

	-100s	-50s	0s	50s	100s
第 151 节龙身 $y(m)$					
第 201 节龙身 $x(m)$					
第 201 节龙身 $y(m)$					
龙尾（后） $x(m)$					
龙尾（后） $y(m)$					

表 6

	-100s	-50s	0s	50s	100s
龙头 (m/s)					
第 1 节龙身 (m/s)					
第 51 节龙身 (m/s)					
第 101 节龙身 (m/s)					
第 151 节龙身 (m/s)					
第 201 节龙身 (m/s)					
龙尾（后） (m/s)					

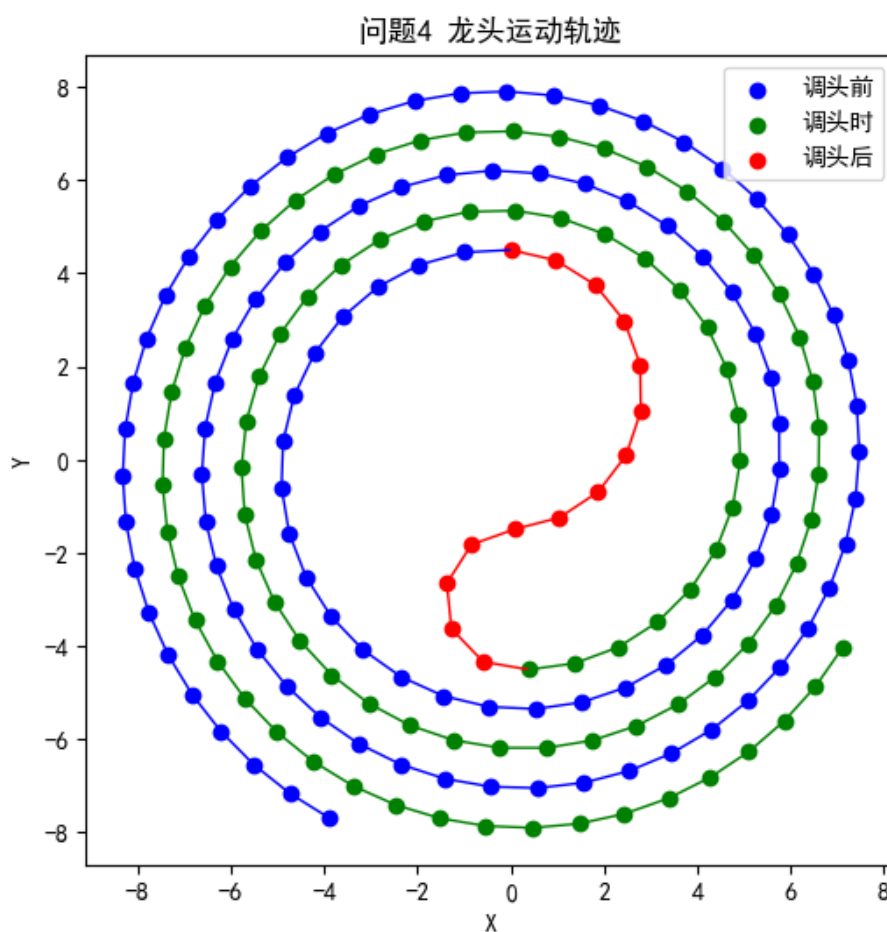


图 15 龙头运动轨迹示意图

第三部分：代码

附录 1

介绍： 问题一代码

```
"""
@Author: blankxiao
@file: A_1.py
@Created: 2024-09-05 21:24
@Desc: A 题第一问 所有点在 300 秒内的运动轨迹
"""

import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
import pandas as pd
from scipy.optimize import fsolve

import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei'] # 显示中文
plt.rcParams['axes.unicode_minus'] = False # 显示负号

# 模拟的时间 s
time = 300
# 螺线距离 m

# 板凳个数 第一节为龙头
num_point = 224

# 龙头初始位置 第一圈角度为 0
loong_head_speed = 1.0

def get_alpha_0(t: int, spiral_d = 0.55):
    """
    获取龙头的角度
    @param t: 时间 s
    @return: alpha0 角度
    """
    # spiral_distance 为螺距
    return 32 * np.pi - sp.sqrt((32 * np.pi) ** 2 - 4 * np.pi / spiral_d
* t)
```

```

def get_r_0(alpha_0: int, spiral_d = 0.55):
    """
    @param alpha_0: alpha0 角度
    @return: r0 极径
    """
    return 16 * spiral_d - spiral_d / (2 * np.pi) * alpha_0

def get_alpha_i(r_i: int, spiral_d = 0.55):
    """
    获取 r_i 的半径
    @param alpha_i: 角度
    @return: r_i 半径 m
    """
    return 32 * np.pi - 2 * np.pi / spiral_d * r_i

def get_r_i(r_i_pre: int, point_index: int, spiral_d = 0.55):
    """
    获取 alpha_i 的半径
    @param r_i: 半径 cm
    @return: alpha_i 角度
    """
    # 把手到边缘的距离
    AD_d = 0.275

    # 龙头两个把手的间距
    L_loong_head = 3.41 - 2 * AD_d
    # 龙身两个把手的间距
    L_loong_body = 2.20 - 2 * AD_d

    bandeng_len = L_loong_body
    if point_index == 1:
        bandeng_len = L_loong_head

    # 定义符号变量
    r_i = sp.symbols('r_i')

    def equation(r_i):
        return r_i_pre ** 2 + r_i ** 2 - 2 * r_i_pre * r_i * sp.cos(2 *
    sp.pi / spiral_d * (r_i - r_i_pre)) - bandeng_len ** 2

    eq = equation(r_i)

```

```

    r_i = sp.nsolve(eq, r_i, r_i_pre, tolerance=1e-6) # 使用 nsolve 求解,
    初始猜测值为 1.0

    return r_i

def plot_polar_scatter(theta, r, color='blue', marker='o', s=30,
alpha=0.8, figsize=(8, 6), line_color='red', line_width=1):
    """
    绘制极坐标散点图，并连接每个点

    theta: 角度值的列表（弧度制）
    r: 半径值的列表
    color: 散点颜色
    marker: 散点形状
    s: 散点大小
    alpha: 散点透明度
    figsize: 图像大小
    line_color: 连接线的颜色
    line_width: 连接线的宽度
    """

    plt.figure(figsize=figsize)
    ax = plt.subplot(111, projection='polar') # 创建极坐标子图

    # 绘制散点
    ax.scatter(theta, r, color=color, marker=marker, s=s, alpha=alpha)

    # 绘制连接线
    ax.plot(theta, r, color=line_color, linewidth=line_width)

    plt.grid(True)
    plt.show()

def get_x_y(alpha, r):
    """
    获取 x,y 坐标
    @param alpha: 角度
    @param r: 半径
    @return: x,y 坐标
    """

    return r * sp.cos(-alpha), r * sp.sin(-alpha)

def get_v_i(v_i_pre: int, r_i_pre: int, r_i: int, spiral_d = 0.55):

```

```

"""
    获取速度 v_i
    @param v_i_pre: 上一个速度
    @param r_i_pre: 上一个点的极径
    @param r_i: 当前极径
    @return: v_i
"""

    coeff_of_v_i_pre = (1 - r_i / r_i_pre * sp.cos(2 * np.pi / spiral_d *
(r_i - r_i_pre)) - 2 * np.pi * r_i / spiral_d * sp.sin(2 * np.pi /
spiral_d * (r_i - r_i_pre)))
    coeff_of_v_i = (1 - r_i_pre / r_i * sp.cos(2 * np.pi / spiral_d *
(r_i - r_i_pre)) + 2 * np.pi * r_i_pre / spiral_d * sp.sin(2 * np.pi /
spiral_d * (r_i - r_i_pre)))
    return - coeff_of_v_i_pre * v_i_pre / coeff_of_v_i

def plot_points_with_lines(x, y, title):
    """
    绘制散点图并将相邻的点连起来
    :param x: x 坐标的列表
    :param y: y 坐标的列表
    """

    # 创建图形
    fig, ax = plt.subplots(figsize=(6, 6))

    # 绘制散点图
    ax.scatter(x, y, color='blue', label='Points')

    # 将相邻的点连起来
    for i in range(len(x) - 1):
        ax.plot([x[i], x[i + 1]], [y[i], y[i + 1]], color='red',
linestyle='-', linewidth=2)

    # 设置图形属性
    ax.set_title(title)
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.legend()

    # 显示图形
    plt.show()

```

```

if __name__ == '__main__':
    df_xyv =
pd.DataFrame(index=pd.MultiIndex.from_product([range(num_point), ['x',
'y', "v"]]), columns=[f'{i} s' for i in range(time + 1)])

    df_rav =
pd.DataFrame(index=pd.MultiIndex.from_product([range(num_point), ['r',
'alpha', "v"]]), columns=range(time + 1))

    df_position = pd.DataFrame(index=[
        [f'龙头{s} (m)' for s in ["x", "y"]] +
        [f'第{i}节龙身{s} (m)' for i in range(1, num_point - 2) for s in
["x", 'y']] +
        [f'龙尾{s} (m)' for s in ["x", 'y']] +
        [f'龙尾(后){s} (m)' for s in ["x", 'y']]
    ],
    columns=[f'{t} s' for t in range(time + 1)])

    df_velocity = pd.DataFrame(index=[
        ['龙头 (m/s)'] + [f'第{i}节龙身 (m/s)' for i in range(1, num_point
- 2)] +
        ['龙尾 (m/s)', '龙尾(后) (m/s)']
    ], columns=[f'{t} s' for t in range(time + 1)])

    t = 300
    for point_index in range(num_point):
        if point_index != 0:
            r_i_pre=df_rav[t][point_index - 1, 'r']
            v_i_pre=df_rav[t][point_index - 1, 'v']

            cur_r = get_r_i(r_i_pre=r_i_pre, point_index=point_index)
            cur_alpha = get_alpha_i(r_i=cur_r)
            cur_r = get_r_i(r_i_pre=r_i_pre, point_index=point_index)
            cur_alpha = get_alpha_i(r_i=cur_r)

            cur_v = get_v_i(v_i_pre=v_i_pre, r_i_pre=r_i_pre, r_i=cur_r)
        else:
            cur_alpha = get_alpha_0(t)
            cur_r = get_r_0(cur_alpha)
            cur_v = loong_head_speed

    cur_x, cur_y = get_x_y(alpha=cur_alpha, r=cur_r)

```

```

# 将 x 和 y 值分别添加到对应的 t 列中
x_index = f"第{point_index}节龙身 x (m)"
y_index = f"第{point_index}节龙身 y (m)"
v_index = f"第{point_index}节龙身 (m/s)"
if point_index == 0:
    x_index = f"龙头 x (m)"
    y_index = f"龙头 y (m)"
    v_index = f"龙头 (m/s)"
elif point_index == num_point - 2:
    x_index = f"龙尾 x (m)"
    y_index = f"龙尾 y (m)"
    v_index = f"龙尾 (m/s)"
elif point_index == num_point - 1:
    x_index = f"龙尾 (后) x (m)"
    y_index = f"龙尾 (后) y (m)"
    v_index = f"龙尾 (后) (m/s)"

df_position.at[x_index, f"{t} s"] = round(cur_x, 6)
df_position.at[y_index, f"{t} s"] = round(cur_y, 6)
df_velocity.at[v_index, f"{t} s"] = round(cur_v, 6)

# 记录当前的 r 和 alpha
df_rav.at[(point_index, 'r'), t] = cur_r
df_rav.at[(point_index, 'alpha'), t] = cur_alpha
df_rav.at[(point_index, 'v'), t] = cur_v

df_xyv.at[(point_index, 'x'), t] = cur_x
df_xyv.at[(point_index, 'y'), t] = cur_y
df_xyv.at[(point_index, 'v'), t] = cur_v
x_list = df_xyv[t][:, 'x']
y_list = df_xyv[t][:, 'y']

plot_points_with_lines(x_list, y_list, f"{t=}时状态图")
# 将两个 DataFrame 写入同一个 CSV 文件的不同 sheet 中
# with pd.ExcelWriter('test.xlsx') as writer:
#     df_position.to_excel(writer, sheet_name='位置')
#     df_velocity.to_excel(writer, sheet_name='速度')

# df_xyv.to_csv('df_xy_10.csv')
# df_rav.to_csv('df_ra_10.csv')

# plot_polar_scatter(alpha_list, r_list)

```

附录 2

介绍： 问题二代码

```
"""
@Author: blankxiao
@file: A_2.py
@Created: 2024-09-06 14:03
@Desc: A_2 求出碰撞时间 和当时各个部分的状态(位置和速度)
主要是求碰撞时间 然后代入 A_1 模型 得出碰撞后各个部分的位置和速度
"""

import pandas as pd
import numpy as np
import sympy as sp

from A_1 import get_alpha_0, get_r_0, get_x_y, get_r_i, get_v_i,
get_alpha_i, loong_head_speed, L_loong_head, L_loong_body,
plot_polar_scatter

def get_pos_of_corner(alpha_i: int, alpha_i_pre: int, r_i: int, r_i_pre:
int, r_0: int, r_1: int, alpha_0: int, alpha_1: int):
    """
    获取三个点的坐标
    """

    # 孔距边界距离 m
    AD_d = 0.275
    # 板凳长度的一半
    CD_d = 0.15
    # A C 的距离
    AC_d = sp.sqrt(AD_d ** 2 + CD_d ** 2)

    # 龙头两个把手的间距
    D_1 = L_loong_head
    # 龙身两个把手的间距
    D_2 = L_loong_body

    Gama = sp.atan(CD_d / AD_d)

    gama_i_2 = - Gama + alpha_i_pre - alpha_i + sp.asin(r_i / D_2 *
sp.sin(alpha_i_pre - alpha_i))
```

```

    gama_i_1 = - Gama + alpha_i_pre - alpha_i + sp.asin(r_i_pre / D_2 *
sp.sin(alpha_i_pre - alpha_i))
    gama_0 = Gama + alpha_0 - alpha_1 + sp.asin(r_1 / D_1 *
sp.sin(alpha_0 - alpha_1))

    r_B_i = sp.sqrt(AC_d ** 2 + r_i ** 2 - 2 * AC_d * r_i *
sp.cos(gama_i_1))
    r_B_i_pre = sp.sqrt(AC_d ** 2 + r_i_pre ** 2 - 2 * AC_d * r_i_pre *
sp.cos(gama_i_2) )
    r_C_0 = sp.sqrt(AC_d ** 2 + r_0 ** 2 - 2 * AC_d * r_0 *
sp.cos(gama_0) )

    beta_B_i = alpha_i_pre + sp.asin(AC_d / r_B_i * sp.sin(gama_i_1))
    beta_B_i_pre = alpha_i - sp.asin(AC_d / r_B_i_pre *
sp.sin(gama_i_2))
    beta_C_0 = alpha_0 + sp.asin(AC_d / r_0 * sp.sin(gama_0))

    x_B_i, y_B_i = get_x_y(alpha=beta_B_i, r=r_B_i)
    x_B_i_pre, y_B_i_pre = get_x_y(alpha=beta_B_i_pre, r=r_B_i_pre)
    x_C_0, y_C_0 = get_x_y(alpha=beta_C_0, r=r_C_0)

    return (x_B_i, y_B_i), (x_B_i_pre, y_B_i_pre), (x_C_0, y_C_0)

def get_end_time():
    """
    获取龙头到达原点的时间 并将位置信息存入 df_rav
    """
    t = sp.var("t")

    # 将 get_alpha_0 代入 get_r_0, 得到关于 t 的函数
    alpha_0 = get_alpha_0(t)
    r_0 = get_r_0(alpha_0)
    # 求解方程 r_0 = 0
    equation = sp.Eq(r_0, 0)
    return float(sp.solve(equation, t)[0])

def forward_vaild(r_0: int, r_1: int, spiral_d: int):
    """
    @param r_0: 龙头极径
    @param r_1: 第一节龙身极径
    返回是否是锐角
    """

```



```

theta_0 = 2 * np.pi * r_0 / spiral_d
theta_1 = 2 * np.pi * r_1 / spiral_d

detal_x = (r_0 * sp.cos(theta_0) - r_1 * sp.cos(theta_1))
dx = sp.cos(theta_0 - theta_0 * sp.sin(theta_0))
detal_y = r_0 * sp.sin(theta_0) - r_1 * sp.sin(theta_1)
dy = sp.sin(theta_0 + theta_0 * sp.cos(theta_0))
return detal_x * dx + detal_y * dy >= 0

def are_collinear(A, B, C, atol=1e-1):
    """
    判断三个极坐标点 A、B、C 是否共线
    :param A: 点 A 的直角坐标
    :param B: 点 B 的直角坐标
    :param C: 点 C 的直角坐标
    :return: 如果 A、B、C 共线, 返回 True
    """
    # 将极坐标转换为笛卡尔坐标
    x_A, y_A = [A[0], A[1]]
    x_B, y_B = [B[0], B[1]]
    x_C, y_C = [C[0], C[1]]

    # 计算向量 AB 和 AC
    vector_AB = (x_B - x_A, y_B - y_A)
    vector_AC = (x_C - x_A, y_C - y_A)

    # 计算向量 AB 和 AC 的叉积
    cross_product = vector_AB[0] * vector_AC[1] - vector_AB[1] *
vector_AC[0]
    # 共线 且 A 在 BC 之间
    print(cross_product, (x_A - x_B) * (x_A - x_C))
    return np.isclose(cross_product, 0, atol=atol) and np.isclose((x_A -
x_B) * (x_A - x_C), 0, atol=atol * 10)

def get_min_t(num_point=10):
    """
    第一次碰撞的时间
    认为碰撞仅存在 2-num_point 这些点
    """

    end = get_end_time()
    print("龙头到达零点的时间", end)

```

```

time_range = np.arange(end - 100, end - 10, 0.1).tolist()

df_rav =
pd.DataFrame(index=pd.MultiIndex.from_product([range(num_point), ['r',
'alpha', "v"]]), columns=time_range)

df_xyv =
pd.DataFrame(index=pd.MultiIndex.from_product([range(num_point), ['x',
'y', "v"]]), columns=time_range)

df_coner =
pd.DataFrame(index=pd.MultiIndex.from_product([range(num_point), ['B_i',
'B_i_pre', "C_0"]]), columns=time_range)

crash_t = 0
crash = False

for t in time_range:
    for point_index in range(num_point):
        if point_index != 0:
            r_i_pre=df_rav[t][point_index - 1, 'r']
            v_i_pre=df_rav[t][point_index - 1, 'v']

            cur_r = get_r_i(r_i_pre=r_i_pre, point_index=point_index)
            cur_alpha = get_alpha_i(r_i=cur_r)

            cur_v = get_v_i(v_i_pre=v_i_pre, r_i_pre=r_i_pre,
r_i=cur_r)
        else:
            cur_alpha = get_alpha_0(t)
            cur_r = get_r_0(cur_alpha)
            cur_v = loong_head_speed

            cur_x, cur_y = get_x_y(alpha=cur_alpha, r=cur_r)

        # 记录当前的 r 和 alpha
        df_rav.at[(point_index, 'r'), t] = cur_r
        df_rav.at[(point_index, 'alpha'), t] = cur_alpha
        df_rav.at[(point_index, 'v'), t] = cur_v

        df_xyv.at[(point_index, 'x'), t] = cur_x
        df_xyv.at[(point_index, 'y'), t] = cur_y
        df_xyv.at[(point_index, 'v'), t] = cur_v

```

```

if point_index > 2:
    alpha_0 = df_rav.at[(0, 'alpha'), t]
    r_0 = df_rav.at[(0, 'r'), t]

    alpha_1 = df_rav.at[(1, 'alpha'), t]
    r_1 = df_rav.at[(1, 'r'), t]

    alpha_i_pre = df_rav.at[(point_index - 1, 'alpha'), t]
    r_i_pre = df_rav.at[(point_index - 1, 'r'), t]

    B_i, B_i_pre, C_0 = get_pos_of_corner(alpha_0=alpha_0,
r_0=r_0, alpha_1=alpha_1, r_1=r_1, alpha_i_pre=alpha_i_pre,
r_i_pre=r_i_pre, alpha_i=cur_alpha, r_i=cur_r )
    df_coner.at[(point_index, 'B_i'), t] = B_i
    df_coner.at[(point_index, 'B_i_pre'), t] = B_i_pre
    df_coner.at[(point_index, 'C_0'), t] = C_0
    if are_collinear(B_i, B_i_pre, C_0):
        print(f'{t}时刻发现碰撞点 r_0 为{r_0}')
        crash = True
        break

    # if forward_vaild(r_0=df_rav[t][0, 'r'], r_1=df_rav[t][1, 'r'],
spiral_d=0.55):
        # print(f'{t}时刻 r_0 r_1 为 锐角')
        # crash = True
        # break
    if crash:
        crash_t = t
        return crash_t
# 到结束了仍然没有碰撞
return crash_t

if __name__ == "__main__":
    get_min_t()

```

附录 3

介绍： 问题三代码

```
"""
@Author: blankxiao
@file: A_3.py
@Created: 2024-09-06 21:46
@Desc: A_3
"""

import numpy as np
import pandas as pd

from A_2 import get_end_time, get_pos_of_corner, are_collinear
from A_1 import get_alpha_0, get_alpha_i, get_r_0, get_r_i, get_v_i,
get_x_y, loong_head_speed


def get_crash_r_0(spiral_d, back_time=20, point_num=10):
    """
    第一次碰撞的时间
    """

    end = get_end_time()
    # print("到达零点时间", end)

    # 认为碰撞仅存在 2-10 这些点
    time_range = np.arange(end - back_time, end, 0.001).tolist()
    df_rav = pd.DataFrame(index=pd.MultiIndex.from_product([range(10),
['r', 'alpha', "v"]]), columns=time_range)

    for t in time_range:
        for point_index in range(point_num):
            if point_index != 0:
                r_i_pre=df_rav[t][point_index - 1, 'r']
                v_i_pre=df_rav[t][point_index - 1, 'v']

                cur_r = get_r_i(r_i_pre=r_i_pre, point_index=point_index,
spiral_d=spiral_d)
                cur_alpha = get_alpha_i(r_i=cur_r, spiral_d=spiral_d)
```

```

        cur_v = get_v_i(v_i_pre=v_i_pre, r_i_pre=r_i_pre,
r_i=cur_r, spiral_d=spiral_d)
    else:
        cur_alpha = get_alpha_0(t, spiral_d=spiral_d)
        cur_r = get_r_0(cur_alpha, spiral_d=spiral_d)
        cur_v = loong_head_speed

    cur_x, cur_y = get_x_y(alpha=cur_alpha, r=cur_r)

    # 记录当前的 r 和 alpha
    df_rav.at[(point_index, 'r'), t] = cur_r
    df_rav.at[(point_index, 'alpha'), t] = cur_alpha
    df_rav.at[(point_index, 'v'), t] = cur_v

    if point_index > 4:
        alpha_0 = df_rav.at[(0, 'alpha'), t]
        r_0 = df_rav.at[(0, 'r'), t]

        alpha_1 = df_rav.at[(1, 'alpha'), t]
        r_1 = df_rav.at[(1, 'r'), t]

        alpha_i_pre = df_rav.at[(point_index - 1, 'alpha'), t]
        r_i_pre = df_rav.at[(point_index - 1, 'r'), t]

        B_i, B_i_pre, C_0 = get_pos_of_corner(alpha_0=alpha_0,
r_0=r_0, alpha_1=alpha_1, r_1=r_1, alpha_i_pre=alpha_i_pre,
r_i_pre=r_i_pre, alpha_i=cur_alpha, r_i=cur_r )
        if are_collinear(B_i, B_i_pre, C_0, atol=1e-1):
            print(f'{t}时刻发现碰撞点')
            return r_0

def get_min_spiral_distance():
    # 螺旋线变化 尝试的步长
    step = 0.2
    spiral_d = 0.55
    times = 10
    while times := times - 1:
        crash_r_0 = get_crash_r_0(spiral_d=spiral_d)
        print(f'{crash_r_0}m')
        print(f'{spiral_d}m')

```

```

        if crash_r_0 > 4.5:
            spiral_d += step
            step /= 2
        else:
            spiral_d -= step
            step /= 2

if __name__ == "__main__":
    get_min_spiral_distance()

```

附录 4

介绍： 问题四代码

```

"""
@Author: blankxiao
@file: A_4.py
@Created: 2024-09-07 22:50
@Desc: A_4 圆内变向
"""

import sympy as sp
import numpy as np
import pandas as pd

PI = np.pi

class Loong_head_pos():
    def __init__(self, spiral_d=1.7, R_0=4.5):
        """
        初始化类，保存计算所需的参数
        :param R_0: 半径
        :param spiral_d: 螺距
        """
        temp_ = (2 * PI * R_0) / spiral_d
        X = - (sp.cos(temp_) - temp_ * sp.sin(temp_))

```

```

Y = - (sp.sin(temp_) + temp_ * sp.cos(temp_))
x_0, y_0 = (R_0 * sp.cos(temp_), R_0 * sp.sin(temp_))

self.cos_entrance_angle = - ((X * x_0) + (Y * y_0)) / (sp.sqrt(X
** 2 + Y ** 2) * sp.sqrt(x_0 ** 2 + y_0 ** 2))
self.R_0 = R_0
self.spiral_d = spiral_d
self.cos_theta_pre = sp.sqrt(1 - self.cos_entrance_angle ** 2)
self.theta_pre = sp.asin(self.cos_entrance_angle)

# 圆的半径
self.R_1 = R_0 * 2 / self.cos_theta_pre / 3
self.R_2 = R_0 / self.cos_theta_pre / 3

# 圆心极径
self.r_0_1 = sp.sqrt(self.R_1 ** 2 + R_0 ** 2 - 2 * R_0 *
self.R_1 * self.cos_theta_pre)
self.r_0_2 = sp.sqrt(self.R_2 ** 2 + R_0 ** 2 - 2 * R_0 *
self.R_2 * self.cos_theta_pre)

# 圆心极角
self.theta_0_1 = PI / 2 + sp.asin(self.R_1 / self.r_0_1 *
self.cos_entrance_angle)
self.theta_0_2 = - PI / 2 + sp.asin(self.R_2 / self.r_0_2 *
self.cos_entrance_angle)

def get_pos(self, t: int):
    """
    获取坐标
    四个区间
    alpha beta theta
    分别代表入、内、外，
    """
    alpha = 0
    beta = 0
    theta = 0

    outer_threshold = (self.R_1 + self.R_2) * (PI - 2 *
self.theta_pre)

    if t < 0:
        # 入 alpha 有效 beta theta 无效

```

```

        r_0 = sp.sqrt(self.R_0 ** 2 - self.spiral_d * t / PI)
        alpha = 2 * PI / self.spiral_d * (r_0 - self.R_0 +
self.spiral_d / 4)
        elif t > outer_threshold:
            # 外 beta 有效 alpha theta 无效
            r_0 = sp.sqrt(self.R_0 ** 2 + self.spiral_d / PI * (t -
(self.R_1 - self.R_2) * (PI - 2 * self.theta_pre)))
            beta = 2 * PI / self.spiral_d * (r_0 - self.R_0 +
self.spiral_d / 4)
        else:
            # 内 theta 有效 beta alpha 无效
            big_threshold = self.R_1 * (PI - 2 * self.theta_pre)
            if t < big_threshold:

                D1_01_A0 = t / self.R_1
                D1_01_0 = sp.acos((self.r_0_1 ** 2 + self.R_1 ** 2 -
self.R_0 ** 2) / (2 * self.r_0_1 * self.R_1))
                O_01_A0 = sp.Abs(D1_01_A0 - D1_01_0)
                r_0 = sp.sqrt(self.r_0_1 ** 2 + self.R_1 ** 2 - 2 *
self.r_0_1 * self.R_1 * sp.cos(O_01_A0))
                theta_threshold = self.R_1 * D1_01_0
                O1_0_A0 = sp.acos((r_0 ** 2 + self.r_0_1 ** 2 - self.R_1
** 2) / (2 * r_0 * self.r_0_1))
                if t < theta_threshold:
                    theta = self.theta_0_1 - O1_0_A0
                else:
                    theta = self.theta_0_1 + O1_0_A0 - 2 * PI
            else:
                O_02_A0 = sp.acos((self.r_0_2 ** 2 + self.R_2 ** 2 -
self.R_0 ** 2) / (2 * self.r_0_2 * self.R_2)) + t / self.R_2 - (1 +
self.R_1 / self.R_2) * (PI - 2 * self.theta_pre)
                r_0 = sp.sqrt(self.r_0_2 ** 2 + self.R_2 ** 2 - 2 *
self.r_0_2 * self.R_2 * sp.cos(O_02_A0))
                theta = self.theta_0_2 - O_02_A0
        return r_0, alpha, beta, theta

class Loong_body_pos():
    def __init__(self, spiral_d=1.7, R_0=4.5):
        """
        初始化类，保存计算所需的参数
        :param R_0: 半径
        :param spiral_d: 螺距

```



```

"""

self.R_0 = R_0

# 把手到边缘的距离
AD_d = 0.275

# 龙头两个把手的间距
L_loong_head = 3.41 - 2 * AD_d
# 龙身两个把手的间距
L_loong_body = 2.20 - 2 * AD_d

temp_ = (2 * PI * R_0) / spiral_d
X = - (sp.cos(temp_) - temp_ * sp.sin(temp_))
Y = - (sp.sin(temp_) + temp_ * sp.cos(temp_))
x_0, y_0 = (R_0 * sp.cos(temp_), R_0 * sp.sin(temp_))

self.cos_entrance_angle = - ((X * x_0) + (Y * y_0)) / (sp.sqrt(X
** 2 + Y ** 2) * sp.sqrt(x_0 ** 2 + y_0 ** 2))
self.R_0 = R_0
self.spiral_d = spiral_d
self.cos_theta_pre = sp.sqrt(1 - self.cos_entrance_angle ** 2)
self.theta_pre = sp.asin(self.cos_entrance_angle)

# 圆的半径
self.R_1 = R_0 * 2 / self.cos_theta_pre / 3
self.R_2 = R_0 / self.cos_theta_pre / 3

# 圆心极径
self.r_0_1 = sp.sqrt(self.R_1 ** 2 + R_0 ** 2 - 2 * R_0 *
self.R_1 * self.cos_theta_pre)
self.r_0_2 = sp.sqrt(self.R_2 ** 2 + R_0 ** 2 - 2 * R_0 *
self.R_2 * self.cos_theta_pre)

# 圆心极角
self.theta_0_1 = PI / 2 + sp.asin(self.R_1 / self.r_0_1 *
self.cos_entrance_angle)
self.theta_0_2 = - PI / 2 + sp.asin(self.R_2 / self.r_0_2 *
self.cos_entrance_angle)

const_dict = {}

```

```

func_list = [self.get_t1_r_1, self.get_t2_r_2, self.get_t3_r_3,
self.get_t4_r_4]
for func in func_list:
    for pos in ["head", "body"]:
        L_len = L_loong_head if pos == "head" else L_loong_body
        const_dict[func.__name__ + "_" + pos] = func(L_len)
        print(const_dict[func.__name__ + "_" + pos])
print()
print(const_dict)

def get_t1_r_1(self, L_len: int):
    r_1_pre = sp.symbols('r_1_pre')
    theta_1_pre = self.theta_0_1 - sp.acos((r_1_pre ** 2 + self.r_0_1
** 2 - self.R_1 ** 2) / (2 * r_1_pre * self.r_0_1))
    ep = self.R_0 ** 2 + r_1_pre ** 2 - 2 * self.R_0 * r_1_pre *
sp.cos(theta_1_pre) - L_len ** 2

    r_1_value = sp.nsolve(ep, r_1_pre, self.R_0 / 2)
    theta_1_pre_value = theta_1_pre.subs({r_1_pre: r_1_value})
    return (r_1_value, theta_1_pre_value)

def get_t2_r_2(self, L_len: int):
    r_2_pre = sp.symbols('r_2_pre')
    theta_2_pre = self.theta_0_2 - sp.acos((r_2_pre ** 2 + self.r_0_2
** 2 - self.R_2 ** 2) / (2 * r_2_pre * self.r_0_2))
    ep = self.r_0_1 ** 2 + r_2_pre ** 2 - 2 * self.r_0_1 * r_2_pre *
sp.cos(theta_2_pre - self.theta_0_1 + PI) - L_len ** 2

    r_2_value = sp.nsolve(ep, r_2_pre, self.R_0 / 2)
    theta_2_pre_value = theta_2_pre.subs({r_2_pre: r_2_value})
    return (r_2_value, theta_2_pre_value)

def get_t3_r_3(self, L_len: int):
    r_3_pre = sp.symbols('r_3_pre')
    theta_3_pre = self.theta_0_2 - sp.acos((r_3_pre ** 2 + self.r_0_2
** 2 - self.R_2 ** 2) / (2 * r_3_pre * self.r_0_2))
    ep = (self.R_0 / 3) ** 2 + r_3_pre ** 2 - 2 * (self.R_0 / 3) *
r_3_pre * sp.cos(theta_3_pre + PI / 2) - L_len ** 2

    r_3_value = sp.nsolve(ep, r_3_pre)[0]
    theta_3_pre_value = theta_3_pre.subs({r_3_pre: r_3_value})
    return r_3_value, theta_3_pre_value

```

```

def get_t4_r_4(self, L_len: int):
    r_4_pre = sp.symbols('r_4_pre')
    beta_4_pre = 2 * PI / self.spiral_d * (r_4_pre - self.R_0 -
self.spiral_d / 4)
    ep = self.R_0 ** 2 + r_4_pre ** 2 - 2 * self.R_0 * r_4_pre *
sp.cos(beta_4_pre + PI / 2) - L_len ** 2

    r_4_value = sp.nsolve(ep, r_4_pre)[0]
    theta_4_pre_value = beta_4_pre.subs({r_4_pre: r_4_value})
    return r_4_value, theta_4_pre_value

if __name__ == '__main__':
    # loong_head_pos = Loong_head_pos()

    # time_range = np.linspace(-100, 100, 201)
    # df_rav = pd.DataFrame()
    # df_xyv = pd.DataFrame()
    # for t in time_range:
    #     r_0, alpha, beta, theta = loong_head_pos.get_pos(t)
    #     new_row = pd.DataFrame({'t': [t], 'r_0': [r_0], 'alpha':
[alpha], 'beta': [beta], 'theta': [theta]})
    #     df_rav = pd.concat([df_rav, new_row], ignore_index=True)
    #     angle = alpha
    #     if beta > 0:
    #         angle = beta
    #     elif theta > 0:
    #         angle = theta
    #     new_row = pd.DataFrame({'color': [True if alpha or beta else
False], 'x': [sp.cos(angle) * r_0], 'y': [sp.sin(angle) * r_0]})
    #     df_xyv = pd.concat([df_xyv, new_row], ignore_index=True)

    loong_body_pos = Loong_body_pos()

```

附录 5
介绍： 问题五代码