基于运动学分析的板凳龙路径优化设计

摘要

“板凳龙”是浙中及江南沿海一带流行的民间文化传统，作为我国非物质文化遗产的重要组成部分，在提倡文化传承的当下，其形式的科学化、现代化推广无疑具有重要的研究价值。本文从运动学分析角度出发建立模型，对舞龙队盘入，调头与盘出路径进行了优化设计，为该项目的观赏性提高提供了可行的参考方案。

针对问题一，首先根据问题要求对板凳尺寸与等距螺线进行参数设定，在此基础上建立坐标系。根据螺线弧长公式建立**龙头前把手位置计算模型**，得到任意时刻龙头前把手位置。考虑到所有把手位置均位于螺线路径上，故可通过**递推方式**确定龙身和龙尾把手的位置。类似的，根据速度合成法建立**把手速度计算模型**。对部分结果的可视化分析展示了各把手位置坐标随时间逐步减弱的正弦型振荡变化，且在龙尾处呈速度减小趋势。

针对问题二，考虑到板尺寸与行进轨迹的固定，首先对舞龙队位置随**不同时间分布的高度相似性**进行研究，得出第一次板凳间碰撞只可能发生在龙头或第一节龙身与相邻的外圈板凳。其次，以龙头和第一节龙身作为基准，对外圈同圆心角角度的弧上包含把手位点进行搜寻，得到潜在碰撞板凳。出于方便计算考虑，将判定区域从矩形细分为两三角形，并用**计算几何的向量合成法**进行判断接着，根据上述**碰撞检测模型**，从初始时刻开始以**遍历法**寻找终止时间，得到结果为$412.6s$，并通过可视化分析验证了模型的准确性。

针对问题三，首先将问题转化为：求解满足第一次碰撞发生在调头区域内的最小螺距，并确定终止位置的约束条件。随后采取**变步长搜索法**进行求解，采取三次遍历，得到满足要求的最小螺距为$0.4503m$.

针对问题四，首先根据题意构建调头路径与盘出螺线，根据各节点的相切关系与两条路线的中心对称性，得到调头曲线长度函数。通过对板凳运动的分析，根据**速度投影定理**，判定板凳保持行进状态的临界情况。综上，以开始调头位置为决策变量，建立**调头曲线长度优化模型**，得到满足约束条件的最短调头曲线与调头起始位置半径分别为$12.8561m$, $4.256m$。参考问题一计算模型，对把手位置进行分情况讨论，建立调头区域内的位置计算模型，通过递推得到舞龙队板凳把手分布情况。根据龙头前把手恒定速度的初始条件，采用速度投影原理，进一步递推得到其余各把手速度。

针对问题五，首先对问题四结果数据初步分析，得到一组前把手恒定速度与时间范围内舞龙队各把手上限速度。考虑到问题四模型中速度递推公式两端速度均为一阶，推出两速度间呈**正比例关系**，解得龙头前把手的**最大恒定速度为**$1.0788m/s$.

关键词：运动学分析；遍历搜索；递推；计算几何

一、 问题重述

1.1 问题背景

2024 年 04 月 16 日习近平总书记于《求是》杂志发表的重要文章中明确指出，要传承弘扬中华优秀传统文化，加强对国粹传承和非物质文化遗产保护的支持和扶持。此建议无疑推动了民俗文化的发展，为加强文化强国提供了内在要求。“板凳龙”作为我国第一批认定的国家级非物质文化遗产，其保留了中国尤其是浙中和江南沿海一带“龙信仰”的民间文化传统，融合了书画、剪纸等民间艺术的民间形态，传承了群众体育和广场舞蹈的艺术形式，具有民俗、历史研究价值和民间工艺传承功能，在浦江和浙中及江南沿海一带产生了深远而广泛的影响。

由于传统民俗通过经验传承的特性，其在社会发展带来的功利化思潮下容易面临着传承消逝的局面。今日的板凳龙也正面临这个问题。这是过去放任民间发展的必然结果，也同样是重新肃正、诠释传统民俗文化内涵，破而后立的好时机。通过数学建模的方法对板凳龙盘绕的路线进行精化设计，以科学化理论进行改造与说明，有助于板凳龙文化的传播与传承。

1.2 问题要求

假设舞龙队行进路线为等距螺线，考虑由223 节板凳组成的板凳龙，龙头的板长为341 cm，龙身和龙尾的板长均为220 cm，所有板凳的板宽均为30 cm。每节板凳两头各有一个连结用的把手孔，孔径为5.5 cm，孔的中心距离最近的板头27.5 cm.

问题一：在给定间距的顺时针螺线上，龙头初始时于螺线最外圈以恒定速度盘入，计算从初始时刻到300s 为止，每秒整个舞龙队（即各把手）的位置和速度，并将6 个特定时刻的龙头前把手、6 个特定节点前把手以及龙尾后把手数据汇聚以表格呈现；

问题二：在问题一的基础上，确定舞龙队盘入的终止时刻，使得板凳之间不发生碰撞（即舞龙队不能再继续盘入的时间），求出此时刻舞龙队的各把手参数，并将特定把手数据汇聚成表；

问题三：考虑舞龙队将由顺时针盘入调头切换为逆时针盘出，假设调头空间是以螺线中心为圆心、直径为9 m 的圆形区域，确定使得龙头前把手能够沿着相应螺线盘至区域边界的最小螺距；

问题四：设盘入螺线的螺距为1.7 m，盘出螺线与盘入螺线关于螺线中心呈中心对称，在问题三给出调头空间内设计尽量短的S 型调头路径，满足前一段圆弧的半径是后一段的2 倍，且各转接点均相切的要求。同时给出恒定龙头前把手速度为1 m/s 情形下，$-100s$ 开始到100s 为止，每秒舞龙队的各把手参数，并将特定时刻的特定把手数据汇聚成表；

问题五：在问题四基础上，考虑龙头前把手恒定速度的最大值，使得各把手速度均不超过2 m/s.

二、 问题分析

2.1 问题一的分析

问题要求求解给定条件下一段时间内每秒舞龙队各把手的位置与速度。首先根据给定条件对龙头、龙身/龙尾、舞龙队行进路线等已知量进行模型建立。由于龙头前把手行进速度固定，结合等距螺线弧长方程，可对某一时刻龙头前把手位置进行求取。考虑到同板上两把手间由固定长度刚体连结，且所有把手位置均位于螺线路径上，可通过单位方向向量与切向量，以递推的方式进一步在约束条件中确定龙身和龙尾把手的位置与速度

2.2 问题二的分析

问题要求以舞龙队发生第一次碰撞为终止时间，求解该时刻下舞龙队各把手的位置与速度。考虑到把手行进轨迹均固定以及各板凳具有统一的固定形制，不同时刻下舞龙队位置排布应当存在高相似性。在此性质判定基础上可简化碰撞模型至龙头或第一节板凳与其外围板凳之间。将碰撞转化为板凳端点是否落在潜在碰撞对象四端点围成平面内，并将判定区域从矩形再分化为三角形，用计算几何的向量合成法进行判定

2.3 问题三的分析

问题要求求取能使舞龙队顺利盘入掉头区域内的最小螺距。考虑构建终止时刻龙头前把手距离原点距离关于行进时间与螺距的函数。随后在该函数基础上采取变步长搜索法对螺距进行遍历寻找，从而求得满足条件的最小螺距。

2.4 问题四的分析

问题要求在给定螺距与形制的情况下设计尽量短的调头路径，并在给定龙头前把手恒定速度的基础上要求求取调头时刻周边一定时间范围内每秒舞龙队各把手的位置与速度。首先在坐标系中寻找夹角、切向量、圆弧半径等变量间几何关系，构建调头曲线长度计算公式。然后，根据速度投影定理对各情况下同板把手的运动情况进行评估，寻找边界数值，并据此建立约束条件。结合所有约束条件，以掉头开始位置作为决策变量建立优化模型，遍历求得最小调头路径长度。随后参考问题一模型，对不同情况下龙头前把手位置、龙身间把手连接情况进行分类讨论，通过递推的方式即可确定全舞龙队各把手位置与速度数据。

2.5 问题五的分析

问题要求在各把手速度存在给定上限条件下，求取龙头前把手位于问题四中设计路径内的最大恒定速度。对问题四模型中的速度递推公式进行分析，建立龙头前把手最大恒定速度与舞龙队各把手行进最大速度之间的比例关系，与前问结果数据联立，即可解得要求的龙头前把手最大恒定速度。

三、 模型假设

1. 将板凳视为刚体，不发生形变；
2. 假设板凳在移动时把手中心始终在螺旋线或调头曲线上；

3. 不考虑板凳间的摩擦力；
4. 假设各板凳始终在同一平面上；
5. 假设板凳始终保持前进状态，不会中途停止或后退。

四、 符号说明

表 1 符号说明

符号	说明	单位
L	板凳长度	m
h	板凳两把手距离	m
d	板凳宽度	m
δ	螺距	m
(ρ,θ)	极坐标	/
s	弧长	m
t	时间	s
v_i	第i个把手速度	m/s
P_i	第i个把手编号	/
φ	相切圆弧夹角	rad
r	圆弧半径	m
R	调头空间半径	m
λ	调头曲线长度	m

五、 模型建立与求解

5.1 问题一的模型建立与求解

5.1.1 板凳参数说明

根据题意，对板凳龙各节参数进行设定如下，如图 1 所示：龙头板长$L_0=3.41m$，板上两把手孔孔心间距$h_0=2.86m$，龙身和龙尾板长$L_1=2.20m$，同板上两把手孔孔心间距$h_1=1.65m$. 两种板板宽均为$d=0.3m$，板上把手孔孔心距离最近板头距离$l=0.275m$.

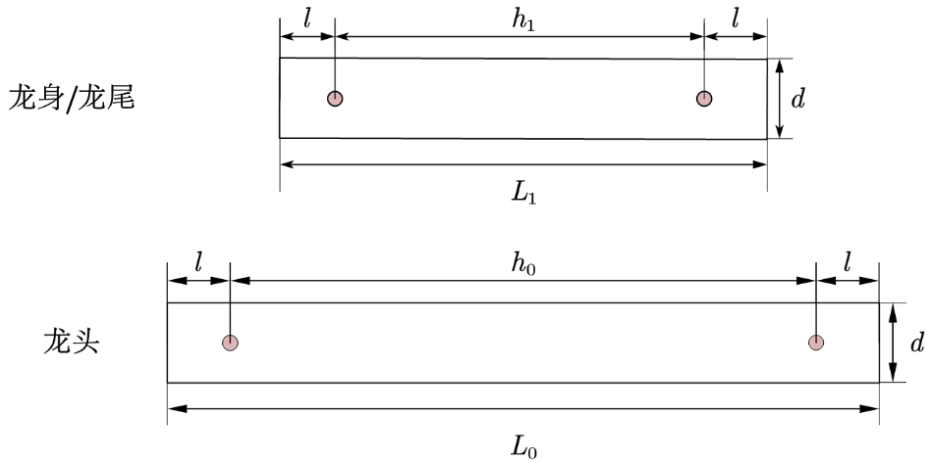

图 1 板凳参数示意图

5.1.2 等螺旋线方程建立

以舞龙队龙头行进路线终点为坐标原点，水平方向为坐标平面，坐标原点与龙头初始位置连线为x轴正方向，建立平面直角坐标系。规定舞龙队行进路线为等距螺线，其基本微分方程形式为：

$$\frac{dr}{d\theta}=a \tag{1}$$

解微分方程可得等距螺线的极坐标基本表达式：

$$r=a\theta+b \tag{2}$$

由于螺线旋转起点位于坐标原点，故方程可简化为：

$$\rho=a\theta \tag{3}$$

其中ρ表示从原点到螺线上任意一点的距离，θ表示原点到该点连线与x轴正半轴的夹角，参数a可由螺距δ求得，关系式为

$$a=\frac{\delta}{2\pi} \tag{4}$$

螺距δ已由题目给出$\delta=0.55\mathrm{m}$，故可根据上式求得参数$a=0.08735\mathrm{m}$。

龙头初始位置A位于x轴正方向与螺线第16圈交点处，其坐标可解得为：

$$A(n\delta,0)=(8.8\mathrm{m},0\mathrm{m}) \tag{5}$$

舞龙队行进路径如图 2 所示。

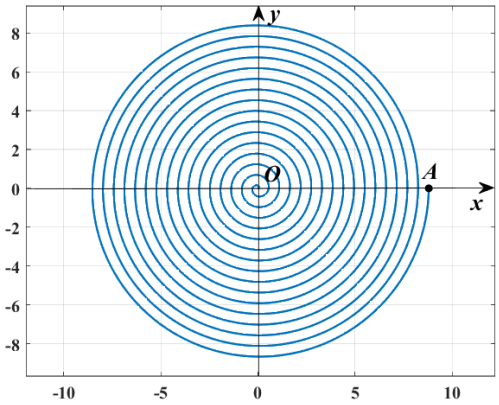

图 2 舞龙队行进路径示意图

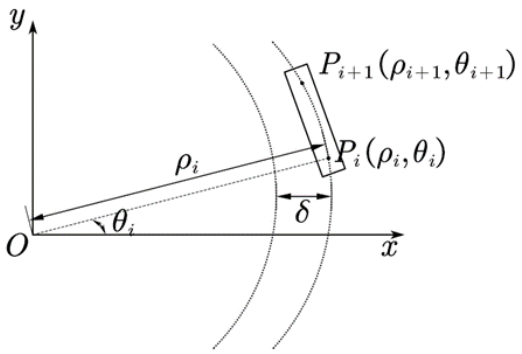

图 3 等距螺线上的板凳坐标

如图 3 所示，各板凳的把手中心均位于等距螺线上，即可将各把手位置用螺线上任一点$P_i(\rho_i,\theta_i)$表示。

为方便后续计算，对于螺线上任意一点，可将其极坐标转化为直角坐标，转换方程为：

$$\begin{cases} x = a\theta\cos\theta \\ y = a\theta\sin\theta \end{cases} \tag{6}$$

则龙头前把手位置可转换为$P_0(x_0,y_0)$表示，第i个龙身（包含龙尾）前把手位置可转换为$P_i(x_i,y_i)$表示。

5.1.3 龙头前把手位置计算模型

龙头前把手的行进速度始终保持$v_0=1\,\mathrm{m/s}$。考虑时刻t时情况，则龙头前把手的前进距离为

$$s = v_0 t \tag{7}$$

同时，由等距螺线的弧长计算公式可知

$$s = \int_\alpha^\beta \sqrt{\rho(\theta)^2+\rho'(\theta)^2}\,d\theta = \int_\alpha^\beta a\sqrt{\theta^2+1}\,d\theta \tag{8}$$

龙头前把手初始位置为A点，故有$\beta=32\pi,\alpha=\theta_0$. 将式(7) (8)联立：

$$a\left(\frac{\theta}{2}\sqrt{\theta^2+1}+\frac{1}{2}\ln\left(\theta+\sqrt{\theta^2+1}\right)\right)\Bigg|_{\theta_0}^{32\pi} = v_0 t \tag{9}$$

解得t时刻龙头前把手角度θ_0，代入式(3) (6)求得位置$P_0(x_0,y_0)$.

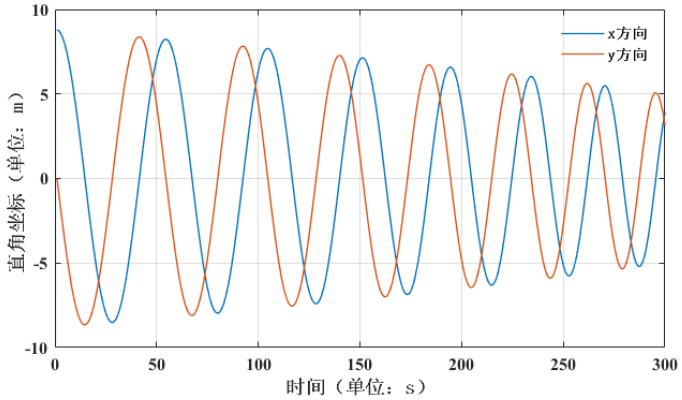

图 4 前 300s 龙头前把手位置

从图 4 可见，龙头前把手位置在x方向和y方向上均随时间呈正弦形振荡变化，且变化范围随时间递进而递减。结合实际评估，可证明本模型的合理性。

5.1.4 龙身和龙尾位置计算模型

考虑到同板上两把手间由固定长度刚体连结，且所有把手位置均位于螺线路径上，可通过递推的方式确定龙身和龙尾把手的位置。

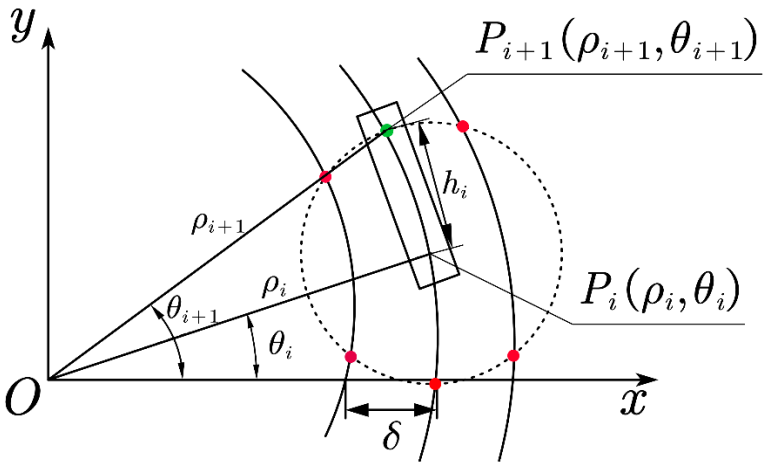

图 5 相邻把手位置关系

如图 5 所示，对于已知位置坐标的把手P_i，其后相邻把手P_{i+1}的位置坐标是唯一确定且可解的，可在如下约束条件中求解：

约束条件 1：所有把手位置均位于螺线路径上，即

$$\begin{cases} \rho_i = a\theta_i \\ \rho_{i+1} = a\theta_{i+1} \end{cases} \tag{10}$$

约束条件 2：由于每一块板板长固定，把手孔孔中心到最近板头距离固定，同板上两孔间间距固定，根据极坐标两点$(r_1,\theta_1),(r_2,\theta_2)$的距离公式

$$d = \sqrt{r_1^2 + r_2^2 - 2r_1 r_2 \cos(\theta_1 - \theta_2)} \tag{11}$$

可推得

$$\begin{cases} {h_0}^2 = \rho_i^2 + \rho_{i+1}^2 - 2\rho_i\rho_{i+1}\cos(\theta_{i+1} - \theta_i),\ i = 0 \\ h_1^2 = \rho_i^2 + \rho_{i+1}^2 - 2\rho_i\rho_{i+1}\cos(\theta_{i+1} - \theta_i), i \neq 0 \end{cases} \tag{12}$$

其中h_0为龙头板两孔间间距，h_1为龙身及龙尾板两孔间间距

约束条件 3：板凳龙按顺时针方向盘入，故对于前后两把手绕心旋转角度严格有

$$\theta_{i+1} > \theta_i \tag{13}$$

约束条件 4：相邻两把手必定位于同一圈层，即二者绕心旋转角度差值存在上限

$$|\theta_{i+1} - \theta_i| < \frac{\pi}{2} \tag{14}$$

综上，联立有

$$\begin{cases}\rho_i=a\theta_i\\ \rho_{i+1}=a\theta_{i+1}\\ \begin{cases}h_0^2=\rho_i^2+\rho_{i+1}^2-2\rho_i\rho_{i+1}\cos(\theta_{i+1}-\theta_i),i=0\\ h_1^2=\rho_i^2+\rho_{i+1}^2-2\rho_i\rho_{i+1}\cos(\theta_{i+1}-\theta_i),i\neq 0\end{cases}\\ \theta_{i+1}>\theta_i\\ |\theta_{i+1}-\theta_i|<\dfrac{\pi}{2}\end{cases}\tag{15}$$

通过上式，可解得第$i+1$个把手位置$P_{i+1}(\rho_{i+1},\theta_{i+1})$，为方便计算研究，再根据式(6)转化为直角坐标$P_{i+1}(x_{i+1},y_{i+1})$

5.1.5 舞龙队速度计算模型

由题已知龙头前把手的恒定速度为$v_0=1\,\mathrm{m/s}$，故其余各把手可通过递推的方式确定：

以第i个龙身为研究对象，取P_i点为基点，分析P_{i+1}点的速度，速度矢量图如图 6 所示，根据速度合成法有

$$\boldsymbol{v_{i+1}}=\boldsymbol{v_i}+\boldsymbol{u}\tag{16}$$

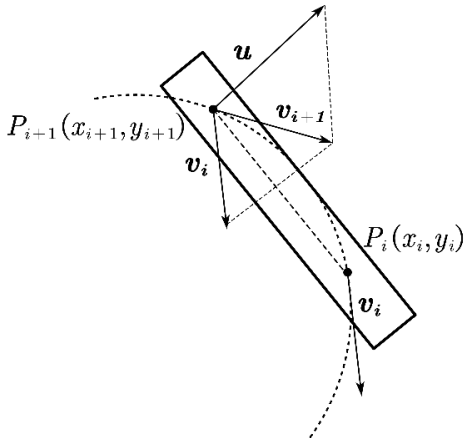

图 6 速度矢量图

式中，$\boldsymbol{u}$为P_{i+1}点相对于基点的速度，其方向与P_iP_{i+1}垂直。根据P_i,P_{i+1}两点坐标，可求得$\boldsymbol{u}$的方向向量为

$$\boldsymbol{k_u}=\begin{pmatrix}y_{i+1}-y_i\\ x_i-x_{i+1}\end{pmatrix}\tag{17}$$

$\boldsymbol{v_i},\boldsymbol{v_{i+1}}$为$P_i,P_{i+1}$两点速度，其方向与等距螺线相切，且$\boldsymbol{v_i}$大小已知。

对式(3)求导，有

$$\begin{cases}x'(\theta)=a\cos\theta-a\theta\sin\theta\\ y'(\theta)=a\theta\cos\theta+a\sin\theta\end{cases}\tag{18}$$

故可求得P_i处的切向量为

$$\boldsymbol{k_i}=\begin{pmatrix}x'(\theta_i)\\ y'(\theta_i)\end{pmatrix}=\begin{pmatrix}a\cos\theta_i-a\theta_i\sin\theta_i\\ a\theta_i\cos\theta_i+a\sin\theta_i\end{pmatrix}\tag{19}$$

将$\boldsymbol{v_i}, \boldsymbol{v_{i+1}}, \boldsymbol{k_u}$单位化，得到对应单位方向向量$\boldsymbol{n_i}, \boldsymbol{n_{i+1}}, \boldsymbol{n_u}$，代入式(16)，有

$$\boldsymbol{n_{i+1}} \cdot v_{i+1} = \boldsymbol{n_i} \cdot v_i + \boldsymbol{n_u} \cdot u \tag{20}$$

解得P_{i+1}速度为

$$v_{i+1} = \frac{\begin{vmatrix} n_{ix} & n_{ux} \\ n_{iy} & n_{uy} \end{vmatrix}}{\begin{vmatrix} n_{i+1x} & n_{ux} \\ n_{i+1y} & n_{uy} \end{vmatrix}} \cdot v_i \tag{21}$$

根据上述推导方法依次递推，可解得板凳龙任一把手的实时运动速度。

5.1.6 问题一求解结果

通过上述模型对300s内每秒整个舞龙队各把手的位置坐标与速度进行求解，结果保存至附录文件 result1.xlsx 中。针对要求的 7 个特殊把手在 6 个特定时刻位置与速度数据，求取结果如下所示。

表 2 问题一舞龙队把手位置

	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)	1.828754	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

表 3 问题一舞龙队把手速度

	0 s	60 s	120 s	180 s	240 s	300 s
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 151 节龙身(m/s)	0.999448	0.999299	0.999078	0.998727	0.998115	0.996861
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

取上述部分结果进行可视化分析，如图 7、8 所示。可以看到，各把手的位置坐标随时间均呈正弦型振荡变化，且随时间递进而递减。由于通过各段路径的时间分先后，振荡曲线随把手次序递增而整体右移。各把手速度总体均随时间呈

衰减趋势，次序越靠前的把手对应初始速度越大，曲线越平缓，即衰减趋势越弱，次序越靠后的把手对应初始速度越小，曲线变化率越大，即衰减趋势越强。综合现实因素考虑，本模型的合理性可以得到验证，且证实了其具有较高精度。

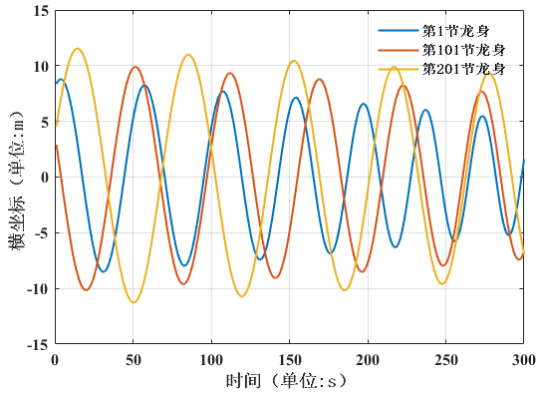

图 7 前 300s 部分把手位置变化趋势

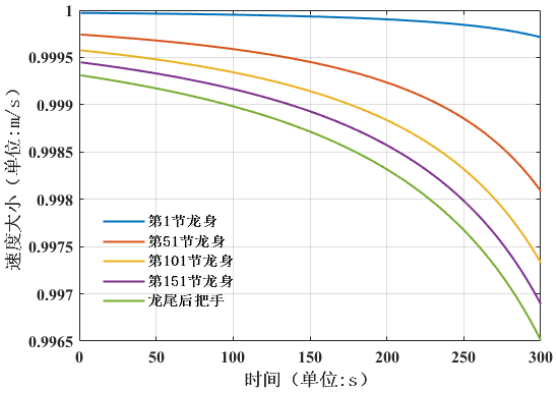

图 8 前 300s 部分把手速度变化趋势

5.2 问题二的模型建立与求解

5.2.1 潜在的板凳间碰撞情形

考虑到各把手行进轨迹均固定在同一条等距螺线上，且龙身与龙尾各板凳具有统一的固定形制，则对于任一时刻的舞龙队位置排布，必定有另一些时刻的舞龙队位置排布与其存在区域性的高度重合，称这一类重合为**相似情形**。取存在相似情形的不同时刻舞龙队位置进行模拟比对，结果如下图所示。

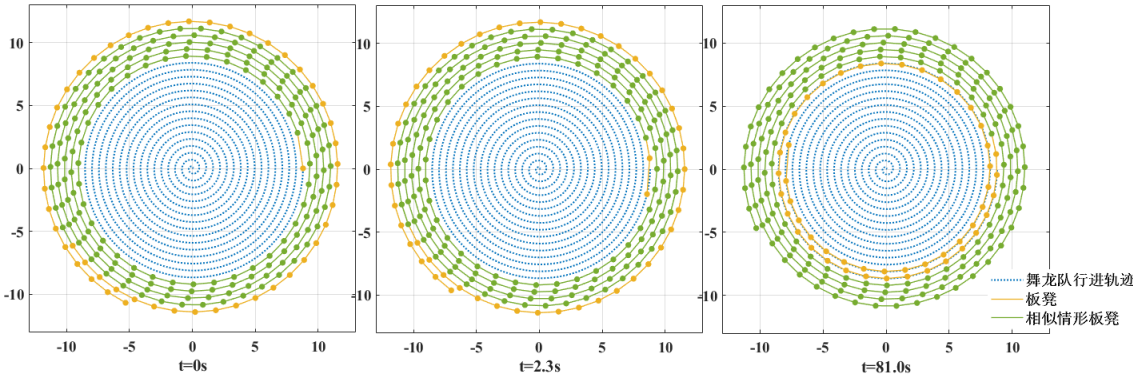

图 9 板凳排布相似情形

可以看到，随龙头逐渐向内圈盘入，舞龙队外围位置排布与前时刻完全重合，仅在最内圈新盘入区域逐渐产生新的排布情形。根据上述结论，第一次碰撞必不可能发生在舞龙队中间，即相似情形区域，因为在那之前必定有对应的相似情形时刻已然发生过。以此类推，**板凳间的碰撞只可能发生在龙头或第一节龙身与相邻的外圈板凳。**

潜在碰撞示意图如图 10 所示。将龙头与第一节龙身合并作为考察对象，对于任意t时刻，龙头前把手位置$P_0(\rho_0,\theta_0)$与第一节龙身后把手位置$P_2(\rho_2,\theta_2)$均为已知。查找$[\theta_0+2\pi,\ \theta_2+2\pi]$范围内所有把手位置，即相邻外圈板凳二点把手位置，依次记为$P_j,P_{j+1},\ldots,P_{j+n}$. 由于板的位置由把手位置决定，故可确定第$j,j+1\ldots j+n$个龙身，共$n+1$个潜在的可能碰撞对象。结合之前结论可知，**第**

一次碰撞只可能发生在龙头或第一节龙身与相邻外圈的第$j,j+1...j+n$个龙身之间。

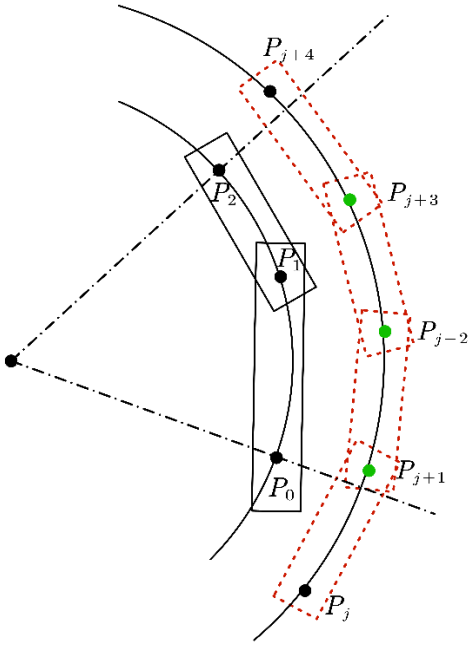

图 10 板凳间潜在碰撞示意图

5.2.2 板凳碰撞检测模型

考虑第i块板凳，按顺时针顺序依次设其四个端点为A_i,B_i,C_i,D_i。由于板形制固定，由已知把手位点P_i,P_{i+1}可确定第i块板全板位置，即可解出A_i,B_i,C_i,D_i四点位置坐标，如图 11 所示。

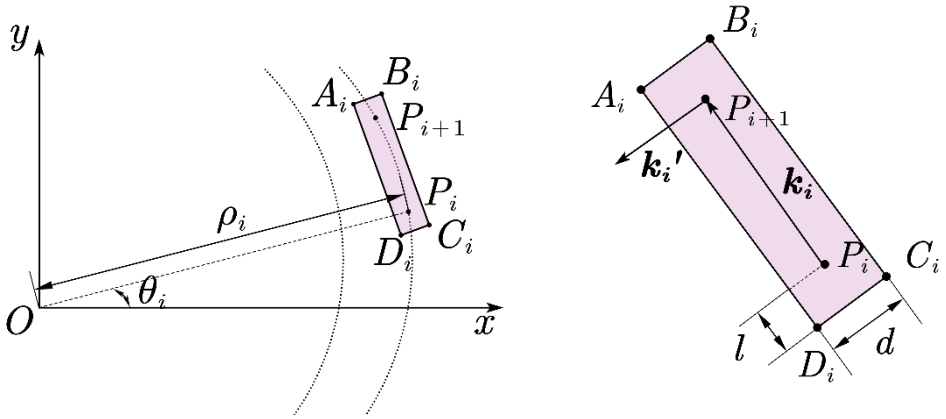

图 11 板凳端点参数示意图

由P_i,P_{i+1}两点坐标$(x_i,y_i),(x_{i+1},y_{i+1})$，得到由$P_i$指向$P_{i+1}$的方向向量为

$$\boldsymbol{k_i}=\begin{pmatrix}x_{i+1}-x_i\\y_{i+1}-y_i\end{pmatrix} \tag{22}$$

法向量为

$$\boldsymbol{k_i}'=\begin{pmatrix}y_i-y_{i+1}\\x_{i+1}-x_i\end{pmatrix} \tag{23}$$

将其单位化后，得到对应单位方向向量和法向量为$\boldsymbol{n_i}',\boldsymbol{n_{i+1}}'$.

根据几何关系，解得端点坐标为：

$$\begin{aligned} A_i &= (x_{i+1}, y_{i+1}) + l \cdot \boldsymbol{n_i} + \frac{1}{2} d \cdot \boldsymbol{n_i}' \\ B_i &= (x_{i+1}, y_{i+1}) + l \cdot \boldsymbol{n_i} - \frac{1}{2} d \cdot \boldsymbol{n_i}' \\ C_i &= (x_i, y_i) - l \cdot \boldsymbol{n_i} - \frac{1}{2} d \cdot \boldsymbol{n_i}' \\ D_i &= (x_i, y_i) - l \cdot \boldsymbol{n_i} + \frac{1}{2} d \cdot \boldsymbol{n_i}' \end{aligned} \tag{24}$$

对模型涉及所有板均进行上述求解，得到每个板对应的端点坐标。判断某时刻两板间是否碰撞，即对该时刻某板凳端点是否在另一板凳内部进行判定。

判断点是否在矩形内：

出于简化模型与方便计算考虑，将各矩形沿对角线均分为两个三角形，如图 12 所示。因此该问题可等价于判定端点是否落在其中任一三角形内。

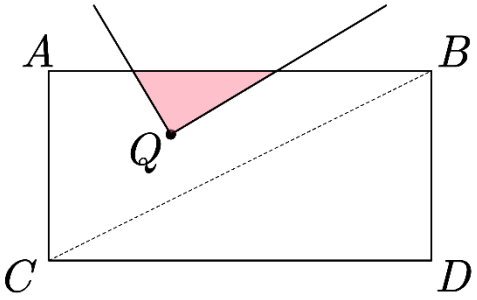

图 12 端点在矩形内

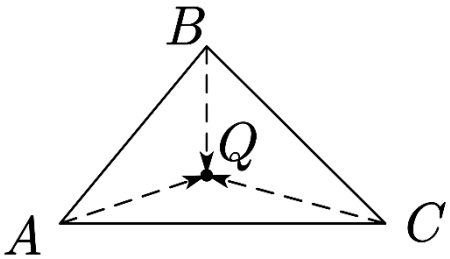

图 13 判断端点是否在三角形内

根据计算几何的向量合成法，如图 13 所示，点在三角形内部的判断条件为：

$$\begin{cases} \left(\overrightarrow{AB} \times \overrightarrow{AC}\right) \cdot \left(\overrightarrow{AB} \times \overrightarrow{AQ}\right) > 0 \\ \left(\overrightarrow{BC} \times \overrightarrow{BA}\right) \cdot \left(\overrightarrow{BC} \times \overrightarrow{BQ}\right) > 0 \\ \left(\overrightarrow{CA} \times \overrightarrow{CB}\right) \cdot \left(\overrightarrow{CA} \times \overrightarrow{CQ}\right) > 0 \end{cases} \tag{25}$$

根据上述模型，即可依次判断龙头与第一节龙身板端点是否落在其外围第 $j, j+1 \ldots j+n$ 板对应矩形内。

5.2.3 问题二的求解与结果

从 $t=0$ 时刻开始，以 $\Delta t = 0.1s$ 为步长，采用**遍历法**寻找终止时间，具体步骤如下：

STEP1: 考虑时刻 t 时的舞龙队板凳情况，基于问题一的位置计算模型，计算此时各把手的位置 $P_0, P_1, \ldots, P_{224}$。

STEP2: 查找可能产生碰撞的相邻外圈板凳 $P_j, P_{j+1}, \ldots, P_{j+n}$.

STEP3: 基于上述碰撞检测模型，判断龙头和第一节的龙身是否与相邻外圈板凳 $P_j, P_{j+1}, \ldots, P_{j+n}$ 发生碰撞。

STEP4: 若发生碰撞，则该时刻即为终止时间T；若没有碰撞，回到 STEP1，考虑$t+\Delta t$时情况。

求解结果：

计算得到，舞龙队的终止时间为$T=412.6s$.

根据问题一建立的模型对舞龙队位置与速度进行求解，将结果存放至附录文件 result2.xlsx 中，针对要求的 7 个特殊把手数据，求取结果如表 4 所示。

表 4 问题二终止时间把手位置和速度

	x(m)	y(m)	速度(m/s)
龙头 (m/s)	1.312391	1.869198	1.000000
第 1 节龙身(m/s)	-1.547380	1.833081	0.991466
第 51 节龙身(m/s)	1.398137	4.287713	0.976700
第 101 节龙身(m/s)	-0.658360	-5.86589	0.974389
第 151 节龙身(m/s)	0.846861	-6.971840	0.973446
第 201 节龙身(m/s)	-7.909810	-1.109140	0.972934
龙尾（后(m/s)	1.077883	8.306562	0.972775

对该时刻全舞龙队位置数据进行可视化分析，结果如图 14 所示。

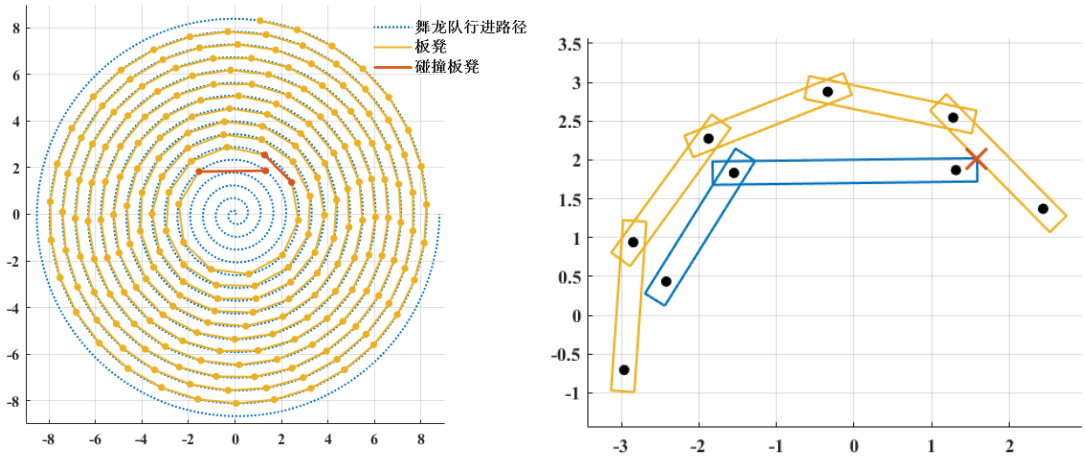

图 14 $T=412.6s$ 时板凳间发生碰撞

可以看到，发生碰撞时龙头已经位于内圈，较为靠近坐标轴原点。碰撞发生在龙头端点与相邻外圈板的内侧边缘之间，经结合实际与资料比对，认为该结果较为合理，验证了本模型的准确性。

5.3 问题三的模型建立与求解

5.3.1 最小螺距的模型建立

为实现舞龙队由顺时针盘入到逆时针盘出的调头切换，需在螺线路径内圈预留出一定空间进行缓冲。根据问题要求，将调头空间设置为以螺线中心为圆心、半径$R=4.5m$的圆形区域（如图 15 所示）。同时要求在满足龙头前把手能够顺利沿着相应盘入螺线盘入到调头空间区域内前提下，取螺距的最小值。即求解满足第一次碰撞发生（终止位置）在调头区域内的最小螺距。将调头空间与螺线交点记为B，碰撞情形如图 16 所示。

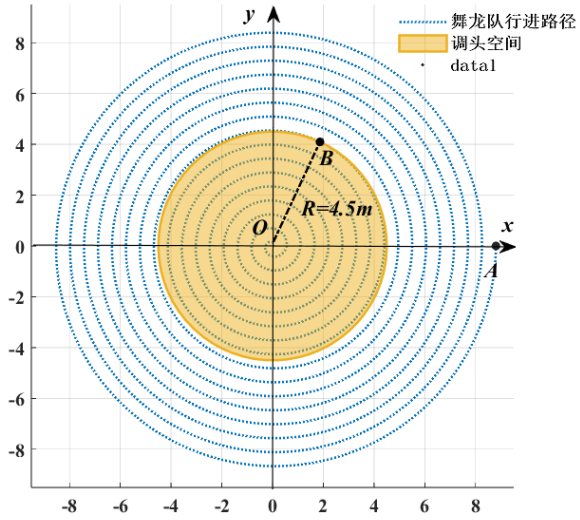

图 15 调头空间

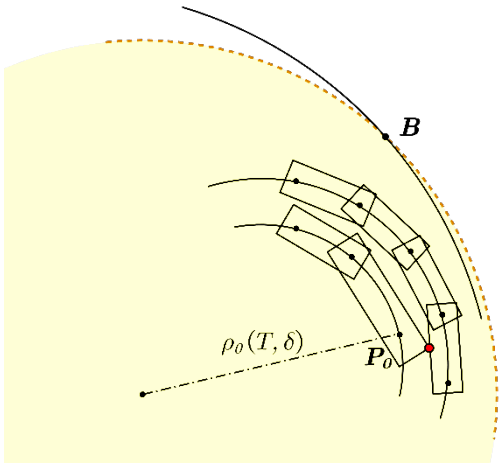

图 16 碰撞时终止位置

终止时刻龙头前把手距离原点距离ρ_0可写作关于时间t和螺距δ的函数$\rho_0(t,\delta)$，将该式作为核心函数，则问题可转化为如下关系进行求解：

$$\min\ \delta \quad s.t.\ \rho_0(T,\delta) < R \tag{26}$$

5.3.2 基于变步长搜索的模型求解

在上述模型基础上，本文采用变步长搜索法进行求解，以满足龙头前把手能够沿着相应螺线盘入到调头空间的边界为边界约束，遍历寻找最小螺距。具体算法步骤如下：

STEP1：确定参数初始值。参考问题二结果，设定初始数值参考范围，在该范围内采取较大步长进行遍历；

STEP2：对初始遍历结果进行筛选，将盘入终止位置在调头区域边界附近对应的结果范围作为下一次遍历范围，相应缩小步长再次进行遍历；

STEP3：重复 STEP2，最终确定较精确的最小螺距数据。

综合精细数据与效率评估，最终决定采用三次遍历，精度可达到$0.0001\,\mathrm{m}$，对应遍历范围与步长如表 5 所示：

表 5 遍历范围和步长选取

	遍历范围δ（单位：m）	遍历步长$\Delta\delta$（单位：m）
第一次遍历	$[0.40, 0.55]$	0.01
第二次遍历	$[0.44, 0.46]$	0.001
第三次遍历	$[0.449, 0.452]$	0.0001

求解结果：

解得满足要求的最小螺距$\delta = 0.4503\,\mathrm{m}$.

结果分析：

选取第二次遍历过程进行可视化分析，结果如图 17、18 所示。

可以看到，在第二次遍历范围内，最终碰撞位置到原点距离随螺距缩小整体呈缩小趋势，且有明显的分层现象。考虑到随螺距缩小带来的每圈弧长减小，每圈层容纳板数量减小，导致碰撞时龙头碰撞对象产生变化（如图中可见三层分别对应碰撞第16、17、18节龙身），最终碰撞位置存在较大差异。结合模拟结果与实际考虑，遍历结果具有较高合理性。在此结果基础上以调头区域半径 $R=4.5m$ 作为边界条件，在其附近选取一定区域作为参考范围，进行下一次遍历。

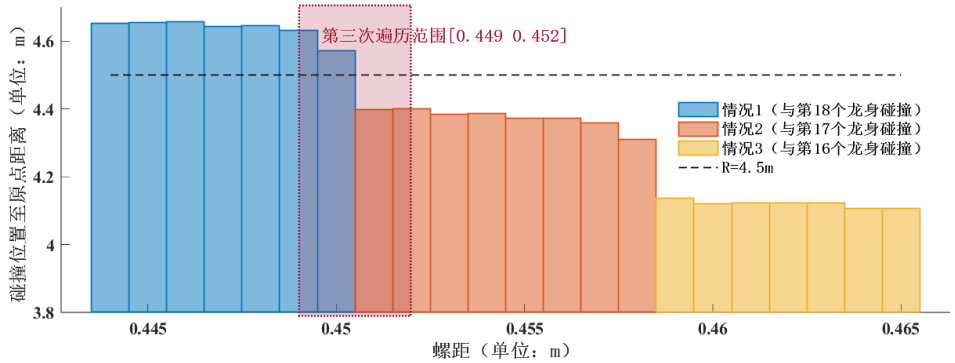

图 17 第二次遍历后碰撞时龙头前把手位置

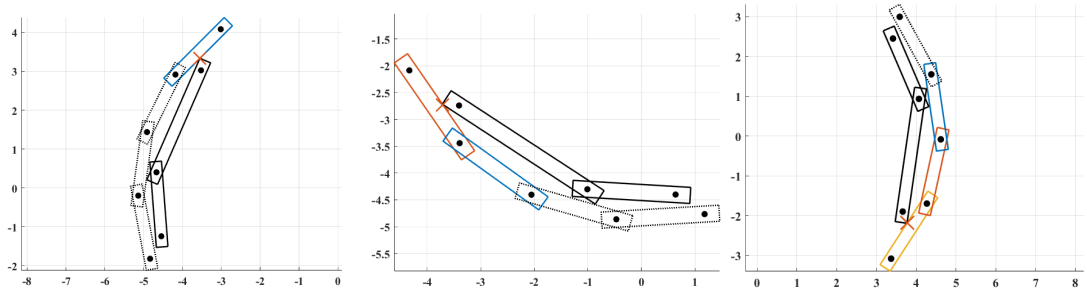

图 18 三种碰撞情况示意图

5.4 问题四的模型建立与求解

5.4.1 调头路径与盘出螺线

根据题意对调头路径与盘出螺线进行构建。如图 19 所示，设盘入、盘出螺线与调头区域边界分别交于点 B、F，与调头路径分别相切连接于点 C、E，调头路径由分别以 O_1,O_2 为圆心的两段圆弧组成，点 D 为两段圆弧切点。考虑到盘入螺线与盘出螺线关于螺线中心呈中心对称，故点 B、F 间，点 C、E 间均满足中心对称关系。

本模型最终求取对象为调头路径，为方便计算与函数关系建立，在上述构建基础上添加辅助线并假设变量，如图 20 所示。

假设 C 点极坐标为 (ρ_C,θ_C)，根据式(6)可得其直角坐标 (x_c,y_c)，由中心对称关系，可将 E 点坐标表示为 $(-x_c,-y_c)$，二者结合求得沿 CE 方向的单位向量 $\boldsymbol{l_{CE}}$.

由式(19)得 C 点的单位切向量 $\boldsymbol{l_c}$ 和单位法向量 $\boldsymbol{n_c}$，故夹角 φ 为

$$\varphi=\arccos|\boldsymbol{l_C}\cdot\boldsymbol{l_{CE}}| \tag{27}$$

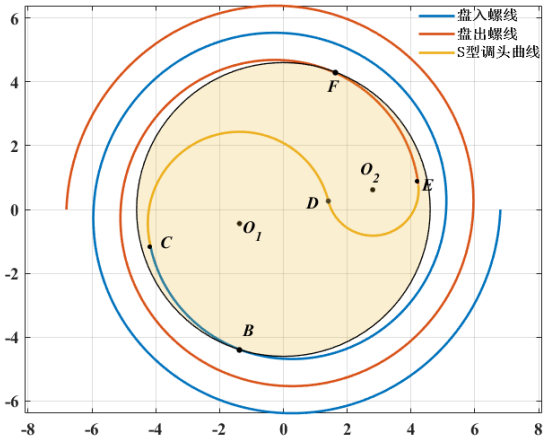

图 19 调头路径与盘出螺线示意图

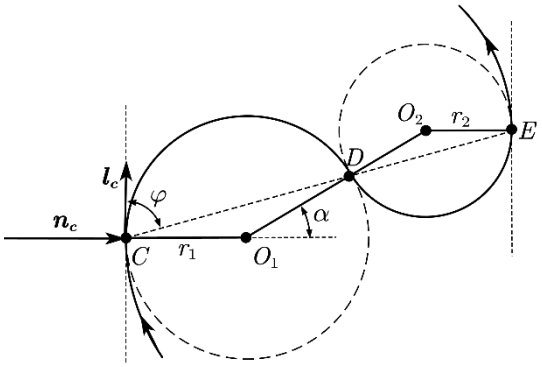

图 20 调头路径相关参数

考虑到圆弧和盘入、盘出螺线间存在的相切关系，可构建出如下约束关系：

$$\begin{cases} r_1 = 2r_2 \\ (r_1 + r_2)\,(1 + \cos\alpha) = CE \cdot \sin\varphi \\ (r_1 + r_2)\sin\alpha = CE \cdot \cos\varphi \end{cases} \tag{28}$$

其中r_1为前一段圆弧半径，r_2为后一段圆弧半径。由图中易知夹角α满足：

$$\alpha = \pi - 2\varphi \tag{29}$$

则两段圆弧半径可分别通过如下关系式求解：

$$\begin{cases} r_1 = \dfrac{2}{3} \cdot |CE| \cdot \dfrac{\cos\varphi}{\sin\alpha} \\ r_2 = \dfrac{1}{3} \cdot |CE| \cdot \dfrac{\cos\varphi}{\sin\alpha} \end{cases} \tag{30}$$

由于C、E两点位置已知，根据图中几何关系，两圆弧圆心坐标可各表示为

$$O_1 = (x_c, y_c) + r_1 \cdot \boldsymbol{n_c},\ O_2 = (x_e, y_e) - r_2 \cdot \boldsymbol{n_c} \tag{31}$$

则调头曲线长度λ可表示为

$$\lambda = (r_1 + r_2)\,(\pi - \alpha) \tag{32}$$

5.4.2 板凳锁死情况

结合标准尺寸与实际路径大小进行评估，当调头路径圆弧半径较小时，可能出现如图 21（a）所示情况，即当板前把手抵达E点，即将走出路径进入盘出螺旋时，板后把手尚停留在前一段圆弧上。

此时对该板运动进行分析，根据**速度投影定理**，平面图形内任意两点的速度在这两点连线上的投影相等，此刻把手位置两点连线与前把手速度V_1方向呈锐角，如若想保证前把手正常沿路径继续运动，则后把手必定会在调头路径上产生后退，这与题目要求与假设存在矛盾。

只有当前后把手基本同时位于同一半圆弧路径上时（如图 21（b）所示），连线与前把手速度方向夹角为钝角，才能保证板能顺利通过调头路径，进入盘出螺旋。

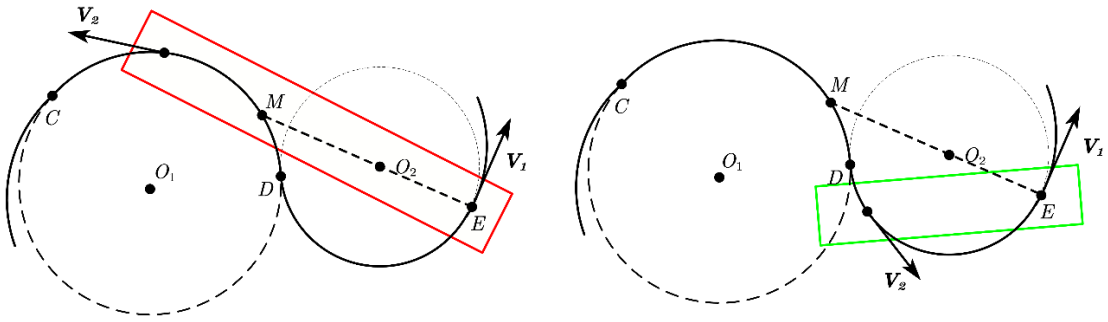

图 21 板凳调头时速度矢量图

将射线EO_2延长，将其与前一段圆弧交点命名为M。当板前把手落在点E上且后把手与M点重合时，把手位置连线与前把手速度方向呈直角结构（如图 22 所示），这意味着此刻在前把手运动的同时，后把手在其应有运动方向上不存在速度，在此时的瞬时速度为0。将此条件下情形作为临界情况，对圆弧半径进行范围限制。

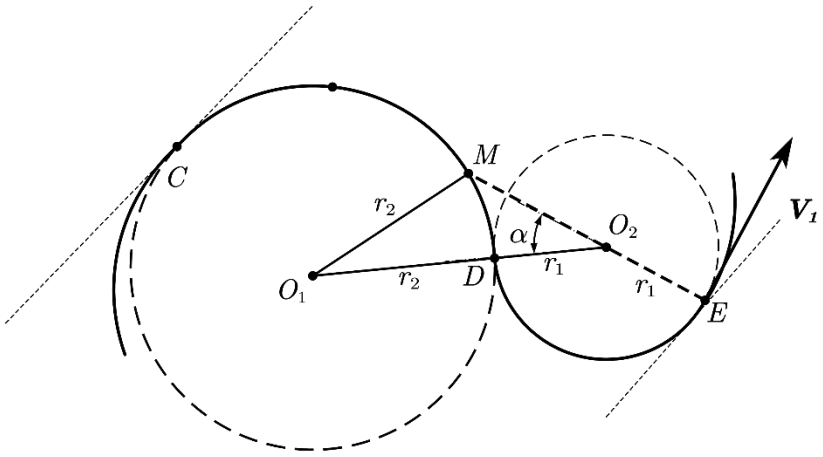

图 22 板凳锁死的临界情况

考虑$\triangle O_1O_2M$，根据余弦定理，有

$$O_1M^2=O_2M^2+O_1O_2{}^2-2MO_2\cdot O_1O_2\cdot\cos\alpha \tag{33}$$

解得O_2M长度为

$$O_2M=3r_2\cos\alpha-\sqrt{9r_2^2\cos^2\alpha-5r_2^2} \tag{34}$$

EM长度为

$$EM=O_2E+O_2M=r_2+3r_2\cos\alpha-\sqrt{9r_2^2\cos^2\alpha-5r_2^2} \tag{35}$$

为保证不出现把手后退的情况，应有约束条件：

$$EM\geqslant\max\{h_0,h_1\} \tag{36}$$

5.4.3 调头曲线长度优化模型

综上，本文建立了调头曲线长度优化模型，在圆弧相切、板凳保持行进等约束条件下，确定圆弧参数，使得调头曲线长度最短。

- **决策变量**

开始调头位置C点会影响到圆弧的排布情况，从而影响到调头曲线长度。因此，决策变量为ρ_C.

- **目标函数**

问题的优化目标为调头曲线长度最短。因此，目标函数为：

$$\min\ \lambda = f(\rho_C) \tag{37}$$

- **约束条件**

1. 调头起始位置C和结束位置E中心对称约束：盘出螺线与盘入螺线关于螺线中心呈中心对称，从而有

$$\begin{cases} x_c = -x_e \\ y_c = -y_e \end{cases} \tag{38}$$

2. S形调头曲线与螺线相切约束：

$$\begin{cases} \alpha = \pi - 3\varphi \\ r_1 = \dfrac{2}{3}\cdot|CE|\cdot\dfrac{\cos\varphi}{\sin\alpha} \\ r_2 = \dfrac{1}{3}\cdot|CE|\cdot\dfrac{\cos\varphi}{\sin\alpha} \end{cases} \tag{39}$$

3. 板凳保持行进状态约束：

$$EM \geqslant \max\{h_1, h_0\} \tag{40}$$

求解结果：

通过对决策变量ρ_C的取值范围进行遍历，得到满足约束条件的最小调头曲线长度为：$\lambda_{min} = 12.8561\,\mathrm{m}$，此时$\rho_C = 4.256\,\mathrm{m}$.

5.4.4 调头空间把手位置计算模型

参考问题一模型方法，考虑将龙头前把手的位置计算结果作为基准，通过递推的方式可逐级确认其后各节龙身/龙尾把手位置。以龙头前把手进入S型调头曲线作为零时刻，对**区域范围内龙头前把手位置**进行分情况讨论并确定。本模型涉及共计 4 种不同可能，分别为：

（1） 龙头前把手位于盘入螺线上：

此情况即为问题一讨论对象，故可直接采取相应方法与结论，在此不再赘述。

（2） 龙头前把手位于前一段圆弧上：

如图 23 所示，以前一段圆弧中心O_1为原点，水平方向为x轴建立平面直角坐标系。记前把手位点为P，可用$\psi_{1D}, \psi_{1C}, \psi_{1P}$分别表示该段圆弧上$D$、$C$、$P$三点关于原点的旋转偏移。对模型进行几何分析，可得到前把手走过路程关于圆弧半径及旋转角度的关系式：

$$s = vt_2 = r_1(\psi_{1P} - \psi_{1D}) \tag{41}$$

在此基础上即可对 P 点坐标进行求解：

$$P_0 = (O_{1x} + r_1\cos\psi_{1P}, O_{1y} + r_1\sin\psi_{1P}) \tag{42}$$

（3） 龙头前把手位于后一段圆弧上：

如图 24 所示，类似于(2)的思路，以圆弧中心O_2为原点建立坐标系。使用$\psi_{2D},\psi_{2C},\psi_{2E}$表示$D$、$C$、$E$三点的旋转偏移，则

此时龙头前把手走过路程可表示为：

$$s=vt_3=r_1(\pi-\alpha)+r_2(\psi_{1P}-\psi_{1D}) \tag{43}$$

对应可将此时 P 点坐标表示为：

$$P_0=(O_{2x}+r_2\cos\psi_{1P},O_{2y}+r_2\sin\psi_{1P}) \tag{44}$$

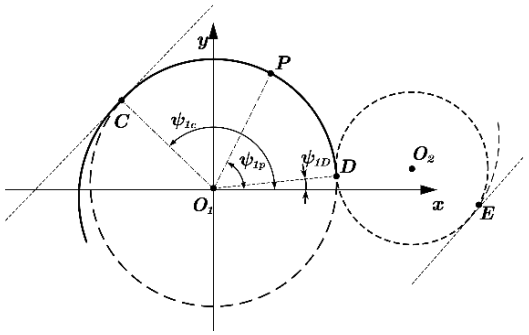

图 23 龙头前把手位于前一段圆弧上

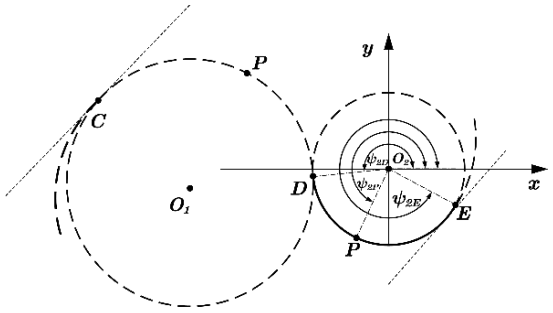

图 24 龙头前把手位于后一段圆弧上

（4） 龙头前把手位于盘出螺线上：

根据题意，盘入螺线与盘出螺线呈中心对称关系，对于龙头前把手位置P_0，可构造盘入螺线上一点P_0'与之中心对称，其坐标可用(ρ_0',θ_0')表示。考虑路程关系式:

$$s=vt_4=(r_1+r_2)(\pi-\alpha)+2\int_{\theta_C}^{\theta_0'}a\sqrt{\theta^2+1}\,d\theta \tag{45}$$

对上式进行求解可得$P_0'(\rho_0',\theta_0')$，根据式（6）将该坐标转化为直角坐标形式$P_0'(x_0',y_0')$，由中心对称关系，进一步得到前把手位置坐标$P_0(-x_0',-y_0')$。

根据确定的龙头前把手位置，通过递推的方式依次求解出**各节龙身把手位置**。同板上把手间连接关系共存在三种类型，分别为：

（1）两把手均位于螺线上

若两把手均位于盘入螺线上，则情况与问题一中讨论问题相同，此处不多赘述；若两把手均位于盘出螺线上，考虑到盘入螺线与盘出螺线的中心对称关系，可将两位点作中心对称变换到盘入螺线上进行计算。

（2）两把手均位于同一圆弧上

参考龙头前把手的位置确定流程，针对第P_i节龙身板上的两把手位点P_i,P_{i+1}，以二者所在的圆弧圆心为坐标原点，水平方向为x轴方向，构建如图 25 所示的平面直角坐标系，分别以ψ_{1i},ψ_{1i+1}表示两位点相对于原点的偏转角度。由图可见，针对两点的角度差γ，存在关系：

$$\gamma=2\arcsin\left(\frac{h}{2r}\right) \tag{46}$$

借助该关系，可通过已确定的把手位点P_i，在圆弧上准确定位下一把手位点P_{i+1}

（3）两把手位于不同曲线上

类似于位于同一圆弧时的判断方法，同样考虑构建坐标系，如图 26 所示。可以看到，当板的前把手行至后一段圆弧上时，此时该板的后把手应停留在前一段圆弧上，即针对位点P_{i+1}，存在约束：

$$\psi_{i+1}\in[\psi_{1D},\psi_{1C}] \tag{47}$$

考虑到该位点与前一段圆弧圆心间距离恒定为r_1，且该位点与已知的前把手位点间距离恒定为h，综合上述条件，本情况下后把手位点可被唯一确定，且可由已知的前把手位点求解得到。

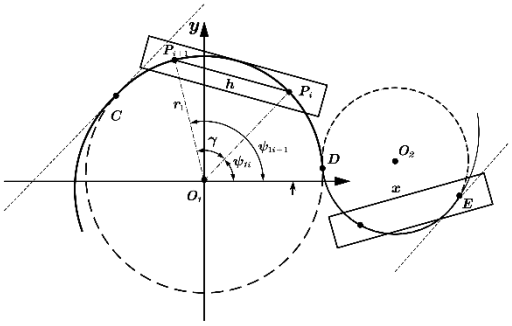

图 25 两把手均位于同一圆弧上

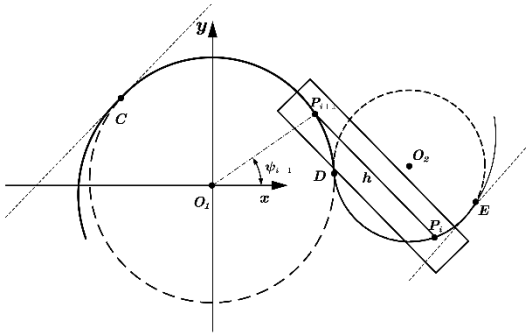

图 26 两把手位于不同曲线上

5.4.5 调头空间把手速度计算模型

参考问题一建立速度计算模型，已知龙头前把手恒定为1m/s，其余各把手速度可以通过递推的方式求出。

以第i节龙身上把手P_i,P_{i+1}为研究对象，根据速度投影原理，同板上两把手P_i,P_{i+1}的运动速度矢量$\boldsymbol{l_i},\boldsymbol{l_{i+1}}$应在把手间连线矢量$\boldsymbol{l_u}$的投影上具有相同长度，即满足等式：

$$v_{i+1}\cdot\boldsymbol{l_{i+1}}\cdot\boldsymbol{l_u}=v_i\cdot\boldsymbol{l_i}\cdot\boldsymbol{l_u}$$

其中由位置决定的l_{i+1},l_u,l_i已有求取结果，v_i亦为已知，故对于舞龙队中任一把手P_{i+1}，其速度均可由其前一把手速度P_i求得。

5.4.6 问题四求解结果

根据调头曲线长度优化模型结果，满足题目约束条件的最小调头路线起点距原点距离$\rho_C=4.256\text{m}$。考虑到计算误差与掉头流畅，取$\rho_C=4.4\text{m}$作为基本参数，将舞龙队龙头前把手抵达调头路线起点（C 点）作为零时刻，按题目要求取前$100s$至后$100s$区间内每秒舞龙队的位置和速度，并将结果存放到文件 result4.xlsx 中。针对要求的 7 个特定把手在 5 个特定时刻的位置与速度数据，求取结果如下表所示

表 6 问题四舞龙队把手位置结果

	-100 s	-50 s	0 s	50 s	100 s
龙头 x(m)	8.293430	6.805374	-3.740955	2.578346	-1.876964
龙头 y(m)	2.154748	0.266951	-2.316302	5.704520	7.918559
第 1 节龙身 x(m)	7.206654	6.209141	-1.649804	4.765843	0.982340

第 1 节龙身 y(m)	4.800220	3.064112	-4.267374	3.862114	7.981644
第 51 节龙身 x (m)	-10.027613	-5.073825	0.847532	-3.138518	3.126930
第 51 节龙身 y(m)	4.370813	-8.178780	-8.058906	-5.384248	3.187371
第 101 节龙身 x (m)	-12.405393	9.162921	4.520570	-6.526549	-6.617023
第 101 节龙身 y(m)	-3.229653	-7.304132	9.482183	6.398974	3.565061
第 151 节龙身 x(m)	-14.452370	12.385820	-5.572866	-6.018833	8.722934
第 151 节龙身 y(m)	-0.339006	-5.344679	11.133718	-9.591266	-5.009301
第 201 节龙身 x(m)	-10.781432	9.265039	-5.384991	-1.282669	9.583696
第 201 节龙身 y(m)	11.721967	-11.865624	13.064488	-13.085851	7.348918
龙尾（后）x(m)	-2.642885	1.827815	-3.536308	7.272701	-11.721576
龙尾（后）y(m)	-16.318934	15.586609	-14.381364	11.817096	-5.301994

表 7 问题四舞龙队把手速度结果

	-100 s	-50 s	0 s	50 s	100 s
龙头 (m/s)	1.000000	1.000000	1.000000	1.000000	1.000000
第 1 节龙身 (m/s)	0.999902	0.999753	0.998562	1.000377	1.000127
第 51 节龙身 (m/s)	0.999333	0.998600	0.994806	1.086437	1.004156
第 101 节龙身 (m/s)	0.999074	0.998200	0.994105	1.084720	1.260936
第 151 节龙身 (m/s)	0.998926	0.997997	0.993809	1.084203	1.259808
第 201 节龙身 (m/s)	0.998831	0.997874	0.993646	1.083955	1.259374
龙尾（后） (m/s)	0.998798	0.997834	0.993595	1.083882	1.259257

选取两任意时间点对应的舞龙队各把手位置数据进行可视化分析，结果如下

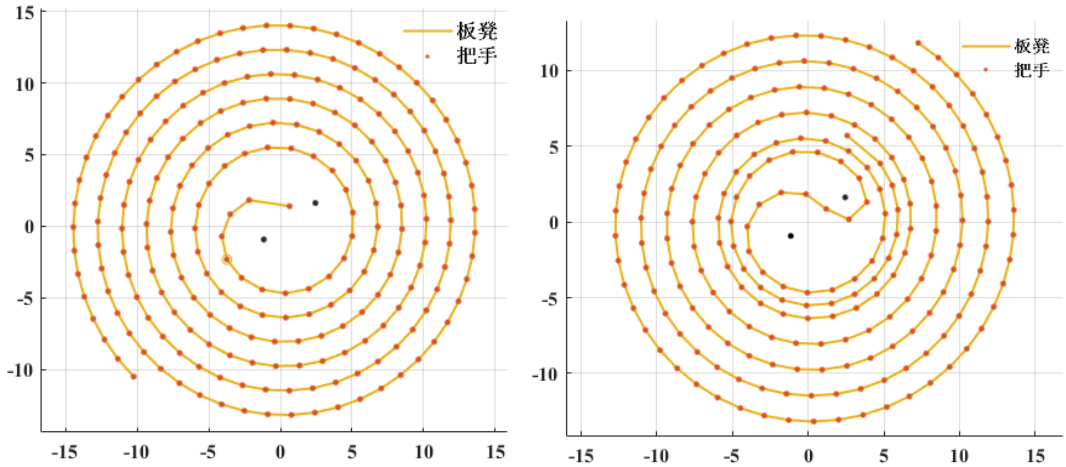

图 27 $T=8s,50s$ 时舞龙队各把手位置分布情况

可以看到，规划下的轨迹可按照题目要求沿与盘入路线中心对称的盘出路线进行盘出，且全流程中板凳间并不发生碰撞，验证了本模型的合理性。

5.5 问题五的模型建立与求解

本题要求龙头最大行进速度，使得舞龙队各把手的速度均不超过$2m/s$. 即求

$$\begin{aligned} &\max\ v_0 \\ &s.t.\ \forall i\in[1,223]\text{和}t\in[-100,100],\text{都有}v(i,t)\leqslant 2\text{m/s} \end{aligned} \tag{48}$$

问题四已解得在龙头前把手速度恒定为$1m/s$时，前$100s$至后$100s$区间内每秒舞龙队各个把手速度，对所有数据筛选，可发现该时间范围内全舞龙队把手最大速度可达$1.853881m/s$。考虑到速度递推公式两端速度均为一阶，易推得前把手恒定速度与时间范围内舞龙队各把手最大速度成正比关系，解得

$$\max\ v_0 = 1.0788m/s \tag{49}$$

六、 灵敏度分析

6.1 模型对板凳长度的灵敏度

在上文中，龙头和龙身的长度分别为$L_0 = 3.41\,\mathrm{m}, L_1 = 2.20\,\mathrm{m}$.考虑到实际情况中，板凳长度可能存在精度误差。为探究板凳长度的改变对发生碰撞时间的影响程度，我们使板凳长度的改变量在$[-10\%, 10\%]$范围，以5%为步长进行调整，并基于问题二的碰撞检测模型得到碰撞时间，如图 28 所示。

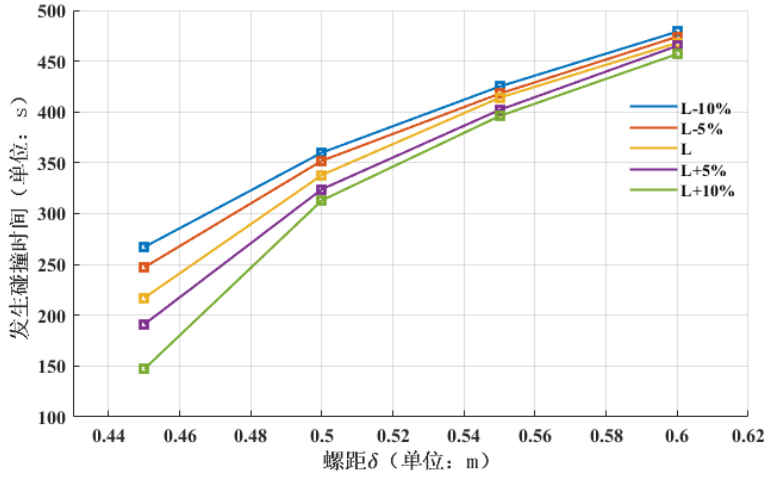

图 28 不同板凳长度下碰撞时间随螺距变化图

从图 28 可以看出，当螺距$\delta > 0.5\,\mathrm{m}$时，板凳长度的改变对发生碰撞时间的影响程度较小，且板凳长度越长，发生碰撞时间越早。当螺距较短时，碰撞时间会显著提早，符合实际情况，反映了模型的高精度和合理性。

七、 模型评价与推广

7.1 模型的优点

- 模型精度较高，仿真模拟速度快；
- 模型鲁棒性强，考虑尽可能全面的情况，如板凳间碰撞、板凳仅保持前进状态等，具有较强的实际意义；
- 模型数据结果全面，便于后续研究与拓展。

7.2 模型的缺点

- 对时间进行离散化处理，当对精度要求较高时可能会造成相应高的时间空间复杂度。

7.3 模型的推广

- 考虑加速度在舞龙过程中的变化，又便于更好的研究运动规律，实现更好的视觉感官效果，同时提供安全方面的建议和帮助。

参考文献

[1] https://www.ihchina.cn/project_details/12829/

附录

支撑材料目录

数据表

1. result1.xlsx 第一问具体数据
2. result2.xlsx 第二问具体数据
3. result4.xlsx 第四问具体数据

代码

1. T1.py base_model
2. T2.m 问题二主模型
3. T2_point.m 板凳端点坐标函数
4. T2_decision.m 判断是否碰撞
5. T4_calc.m 问题四调头主模型

主要程序

T1.py 解决问题一，作为所有模型的 base model

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

class SpiralSystem:
    def __init__(self, delta=0.55, theta_range=32*np.pi, theta_points
=3200,
                 L_0=341/100, L_1=220/100, bench_width=30/100, hole_di
ameter=5.5/100,
                 hole_head_distance=27.5/100, num_body=221):
        # Initialize parameters
        self.theta = np.linspace(0, theta_range, theta_points)
        self.delta = delta
        self.a = delta / (2 * np.pi)
        self.r = self.a * self.theta
        self.L_0, self.L_1 = L_0, L_1
```

```
        self.bench_width = bench_width
        self.hole_diameter = hole_diameter
        self.hole_head_distance = hole_head_distance
        self.h_0 = L_0 - 2 * hole_head_distance
        self.h_1 = L_1 - 2 * hole_head_distance
        self.num_body = num_body

    def plot_spiral(self):
        # Plot the spiral
        x = self.r * np.cos(self.theta)
        y = self.r * np.sin(self.theta)
        plt.figure()
        plt.plot(x, y, 'black', linewidth=1.5)
        plt.axis('equal')
        plt.grid(True)
        plt.title('Archimedean Spiral')
        plt.xlabel('X')
        plt.ylabel('Y')
        plt.show()

    def solve_theta_0(self, t_end=301, gap = 1):
        alpha = 32 * np.pi
        result1 = np.zeros((5, t_end*gap))

        for t in range(t_end*gap):
            k = (alpha / 2) * np.sqrt(alpha ** 2 + 1) + 0.5 * np.log(al
pha + np.sqrt(alpha ** 2 + 1)) - (t/gap / self.a)

            # Equation for fsolve
            equation = lambda theta_0: (theta_0 / 2) * np.sqrt(theta_0
 ** 2 + 1) + 0.5 * np.log(theta_0 + np.sqrt(theta_0 ** 2 + 1)) - k
            theta_0_initial_guess = 1
            theta_0_solution = fsolve(equation, theta_0_initial_guess)


            result1[0, t] = t/gap
            result1[1, t] = theta_0_solution

        result1[2, :] = self.a * result1[1, :]
        result1[3, :] = result1[2, :] * np.cos(result1[1, :])
        result1[4, :] = result1[2, :] * np.sin(result1[1, :])

        return result1
```

```
    def solve_theta_1(self, result1, t_end=301, gap = 1):
        Theta_0 = result1[1, :]
        Rho_0 = result1[2, :]
        result2 = np.zeros((self.num_body + 3, t_end*gap))

        for t in range(t_end*gap):
            h = self.h_0
            theta_0 = Theta_0[t]
            rho_0 = Rho_0[t]

            # Equation for fsolve
            equation = lambda theta_1: rho_0 ** 2 + (self.a * theta_1)
 ** 2 - 2 * rho_0 * self.a * theta_1 * np.cos(theta_0 - theta_1) - h *
* 2
            theta_1_initial_guess = theta_0 + 1
            result2[0, t] = fsolve(equation, theta_1_initial_guess)

            for i in range(1, self.num_body + 3):
                h = self.h_1
                theta_0 = result2[i - 1, t]
                rho_0 = theta_0 * self.a

                equation = lambda theta_1: rho_0 ** 2 + (self.a * theta
_1) ** 2 - 2 * rho_0 * self.a * theta_1 * np.cos(theta_0 - theta_1) -
h ** 2
                theta_1_initial_guess = theta_0 + 1
                result2[i, t] = fsolve(equation, theta_1_initial_gues
s)

        return result2

    def calculate_coordinates(self, result1, result2, t_end=301, gap
= 1):
        Result_theta = np.vstack([result1[1, :], result2])
        Result_rho = self.a * Result_theta
        Result_x = Result_rho * np.cos(Result_theta)
        Result_y = Result_rho * np.sin(Result_theta)
        return Result_x, Result_y, Result_theta, Result_rho

    def export_to_dataframe(self, Result_x, Result_y, t_end=301, gap
= 1):
        data = np.zeros((2 * (self.num_body + 3), t_end*gap))
        for i in range(self.num_body + 3):
            data[2 * i, :] = Result_x[i, :]
```

```
            data[2 * i+1, :] = Result_y[i, :]
        return pd.DataFrame(data)

    def velocity(self, Result_x, Result_y, Result_theta, t_end=301, g
ap = 1, num_body = 221):
        Result_t = np.zeros((num_body + 3, t_end*gap))
        Result_t[0, :] = 1

        for t in range(t_end*gap):
            temp = Result_theta[:, t]
            k = np.array([self.a * np.cos(temp) - self.a * temp * np.s
in(temp),
                          self.a * temp * np.cos(temp) + self.a * np.si
n(temp)])
            k_norm = k / np.linalg.norm(k, axis=0)
            tempx = Result_x[:, t]
            tempy = Result_y[:, t]
            n = np.zeros((223, 2))
            for i in range(223):
                n[i, 0] = -(tempy[i+1] - tempy[i])
                n[i, 1] = tempx[i+1] - tempx[i]
            n_norm = n / np.linalg.norm(n, axis=1)[:, None]
            for i in range(1, 224):
                Result_t[i, t] = Result_t[i-1, t] * (k_norm[0, i-1] *
n_norm[i-1, 1] - k_norm[1, i-1] * n_norm[i-1, 0]) / (k_norm[0, i] * n
_norm[i-1, 1] - k_norm[1, i] * n_norm[i-1, 0])

        return Result_t


spiral = SpiralSystem(delta=0.55)
t_end = 420
gap = 10

result1 = spiral.solve_theta_0(t_end=t_end, gap=gap)
result2 = spiral.solve_theta_1(result1, t_end=t_end, gap=gap)

Result_x, Result_y, Result_theta, Result_rho = spiral.calculate_coord
inates(result1, result2, t_end=t_end, gap=gap)
Velocity = spiral.velocity(Result_x, Result_y, Result_theta, t_end=t_
end, gap=gap, num_body = 221)


with pd.ExcelWriter(f'result_{spiral.delta:.2f}.xlsx') as writer:
```

```
    pd.DataFrame(result1).to_excel(writer, sheet_name='result1', inde
x=False)
    pd.DataFrame(result2).to_excel(writer, sheet_name='result2', inde
x=False)
    pd.DataFrame(Result_x).to_excel(writer, sheet_name='Result_x', in
dex=False)
    pd.DataFrame(Result_y).to_excel(writer, sheet_name='Result_y', in
dex=False)
    spiral.export_to_dataframe(Result_x, Result_y, t_end=t_end, gap=g
ap).to_excel(writer, sheet_name='result_1', index=False)
    pd.DataFrame(Velocity).to_excel(writer, sheet_name='velocity', in
dex=False)

import scipy.io

scipy.io.savemat(f'Result_{spiral.delta:.2f}.mat', {
    'Result_x': Result_x,
    'Result_y': Result_y,
    'Result_theta': Result_theta,
    'Result_rho': Result_rho
})
```

T2.m 计算问题二模型，判断碰撞情况，计算截止时间。

```
clear;
clc;

tic

delta = 0.447;

theta = linspace(0, 32*pi, 3200); % 角度范围

a = delta/(2*pi); % 参数
r = a * theta; % 阿基米德螺线的径向方程

% 龙参数设置
L_0 = 341 / 100;  % 龙头的板长
L_1 = 220 / 100;  % 龙身尾的板长

bench_width = 30 / 100;  % 板宽
hole_diameter = 5.5 / 100;  % 孔径
hole_head_distance = 27.5 / 100;  % 孔中心距最近的板头距离
```

```
h_0 = L_0 - 2*hole_head_distance;  %龙头的孔距
h_1 = L_1 - 2*hole_head_distance;  %龙身尾的孔距

num_body = 221;  % 龙身的数量
%%

t_end = 1500;
t = 0;
load('Result_0.55.mat')
while t<t_end*10
    t = t+1;
    headx = Result_x(1,t);
    heady = Result_y(1,t);
    headtheta = Result_theta(1,t);
    firstx = Result_x(2,t);
    firsty = Result_y(2,t);
    firsttheta = Result_theta(2,t);
    secx = Result_x(3,t);
    secy = Result_y(3,t);
    sectheta = Result_theta(3,t);
    start = 3;
    while headtheta +2*pi>Result_theta(start,t)
        start = start+1;
    end
    start = start-1;
    rear = start;
    while sectheta + 2*pi > Result_theta(rear, t)
        rear = rear +1;
    end

    B1 = T2_point(headx, heady, firstx, firsty);
    B2 = T2_point(firstx, firsty, secx, secy);

    for i = start:rear-1
        tempx = Result_x(i,t);
        tempy = Result_y(i,t);
        temp1x = Result_x(i+1,t);
        temp1y = Result_y(i+1,t);
        BB = T2_point(tempx, tempy, temp1x, temp1y);
        if T2_desicion(B1, BB) == false
            fprintf("螺距%.6f 米\t\t 终止时间%4f 秒\t\t 距离原点%8.6f 米",
 delta, t/10, Result_rho(1,t));
            break
        end
```

```
        if T2_desicion(B2, BB) == false
            fprintf("螺距%.6f 米\t\t 终止时间%4.2f 秒\t\t 距离原点%8.6f 米
", delta, t/10, Result_rho(1,t));
            break
        end
    end
    if T2_desicion(B1, BB) == false
        break
    end
    if T2_desicion(B2, BB) == false
        break
    end
end

%%
% hold on
% axis equal
% scatter(B1(:,1),B1(:,2))
% %scatter(B2(:,1),B2(:,2))
% %scatter(BB(:,1),BB(:,2))
%
% scatter(headx, heady, "filled")
% scatter(firstx, firsty, "filled")
% scatter(secx, secy,"filled")
toc
```

T2_point.m 计算板凳四角坐标的函数

```
function out = T2_point(x1,y1,x2,y2)
    k = [x1-x2,y1-y2];
    k = k/sqrt(k(1)^2 + k(2)^2);
    n = [y2-y1,x1-x2];
    n = n/sqrt(n(1)^2 + n(2)^2);
    A1 = [x2,y2];
    B1 = [x2,y2];
    C1 = [x1,y1];
    D1 = [x1,y1];
    A1 = A1 - 0.275*k - 0.15*n;
    B1 = B1 - 0.275*k + 0.15*n;
    C1 = C1 + 0.275*k + 0.15*n;
    D1 = D1 + 0.275*k - 0.15*n;
    out = [[A1,0];[B1,0];[C1,0];[D1,0]];
end
```

T2_desicion.m 判断碰撞函数

```
function BOOL = T2_desicion(BB1, BB)
    A1 = BB1(1,:);
    A2 = BB(1,:);
    B2 = BB(2,:);
    C2 = BB(3,:);
    if (dot(cross((B2-A2), (A1-A2)), cross((B2-A2), (C2-A2))) > 0)&&
(dot(cross((C2-B2), (A1-B2)), cross((C2-B2), (A2-B2))) > 0)&&(dot(cro
ss((A2-C2), (A1-C2)), cross((A2-C2), (B2-C2))) > 0)
        BOOL = false;
        return
    end

    A1 = BB1(1,:);
    A2 = BB(1,:);
    B2 = BB(3,:);
    C2 = BB(4,:);
    if (dot(cross((B2-A2), (A1-A2)), cross((B2-A2), (C2-A2))) > 0)&&
(dot(cross((C2-B2), (A1-B2)), cross((C2-B2), (A2-B2))) > 0)&&(dot(cro
ss((A2-C2), (A1-C2)), cross((A2-C2), (B2-C2))) > 0)
        BOOL = false;
        return
    end


    A1 = BB1(2,:);
    A2 = BB(1,:);
    B2 = BB(2,:);
    C2 = BB(3,:);
    if (dot(cross((B2-A2), (A1-A2)), cross((B2-A2), (C2-A2))) > 0)&&
(dot(cross((C2-B2), (A1-B2)), cross((C2-B2), (A2-B2))) > 0)&&(dot(cro
ss((A2-C2), (A1-C2)), cross((A2-C2), (B2-C2))) > 0)
        BOOL = false;
        return
    end


    A1 = BB1(2,:);
    A2 = BB(1,:);
    B2 = BB(3,:);
```

```
    C2 = BB(4,:);
    if (dot(cross((B2-A2), (A1-A2)), cross((B2-A2), (C2-A2))) > 0)&&
(dot(cross((C2-B2), (A1-B2)), cross((C2-B2), (A2-B2))) > 0)&&(dot(cro
ss((A2-C2), (A1-C2)), cross((A2-C2), (B2-C2))) > 0)
        BOOL = false;
        return
    end


    A1 = BB1(3,:);
    A2 = BB(1,:);
    B2 = BB(2,:);
    C2 = BB(3,:);
    if (dot(cross((B2-A2), (A1-A2)), cross((B2-A2), (C2-A2))) > 0)&&
(dot(cross((C2-B2), (A1-B2)), cross((C2-B2), (A2-B2))) > 0)&&(dot(cro
ss((A2-C2), (A1-C2)), cross((A2-C2), (B2-C2))) > 0)
        BOOL = false;
        return
    end


    A1 = BB1(3,:);
    A2 = BB(1,:);
    B2 = BB(3,:);
    C2 = BB(4,:);
    if (dot(cross((B2-A2), (A1-A2)), cross((B2-A2), (C2-A2))) > 0)&&
(dot(cross((C2-B2), (A1-B2)), cross((C2-B2), (A2-B2))) > 0)&&(dot(cro
ss((A2-C2), (A1-C2)), cross((A2-C2), (B2-C2))) > 0)
        BOOL = false;
        return
    end


    A1 = BB1(4,:);
    A2 = BB(1,:);
    B2 = BB(2,:);
    C2 = BB(3,:);
    if (dot(cross((B2-A2), (A1-A2)), cross((B2-A2), (C2-A2))) > 0)&&
(dot(cross((C2-B2), (A1-B2)), cross((C2-B2), (A2-B2))) > 0)&&(dot(cro
ss((A2-C2), (A1-C2)), cross((A2-C2), (B2-C2))) > 0)
        BOOL = false;
        return
    end
```

```
    A1 = BB1(4,:);
    A2 = BB(1,:);
    B2 = BB(3,:);
    C2 = BB(4,:);
    if (dot(cross((B2-A2), (A1-A2)), cross((B2-A2), (C2-A2))) > 0)&&
(dot(cross((C2-B2), (A1-B2)), cross((C2-B2), (A2-B2))) > 0)&&(dot(cro
ss((A2-C2), (A1-C2)), cross((A2-C2), (B2-C2))) > 0)
        BOOL = false;
        return
    end

    BOOL = true;
end
```

T4_calc.m 计算问题四相关数据，包含坐标，速度等

```
clc;
clear;

delta = 1.7;
a = delta/(2*pi);

rhoC = 4.4;
thetaC = rhoC/a;
v_0 = 0.1;

o_1 = [-1.15202961402234    -0.925458933386628];
o_2 = [2.44649250561632    1.62088022502350];
mid = [1.246985132403439,0.772100505553458];
r1 = 2.938873900374588;
r2 = 1.469436950187294;
alpha = 0.122828706503503;

L_first_circle = r1*(pi-alpha);
L_second_circle = r2*(pi-alpha);



% 龙参数设置
L_0 = 341 / 100;  % 龙头的板长
L_1 = 220 / 100;  % 龙身尾的板长
```

```
bench_width = 30 / 100;  % 板宽
hole_diameter = 5.5 / 100;  % 孔径
hole_head_distance = 27.5 / 100;  % 孔中心距最近的板头距离

h_0 = L_0 - 2*hole_head_distance;  %龙头的孔距
h_1 = L_1 - 2*hole_head_distance;  %龙身尾的孔距

t_max = 1000;
t = -t_max:1:t_max;

thetahead = zeros(224, 2*t_max+1);
X = zeros(224,2*t_max+1);
Y = zeros(224,2*t_max+1);
BOOL = zeros(224, 2*t_max+1);

% -100~0
for tt = -t_max:1:0
    equation = @(theta) a*((theta/2)*sqrt(theta^2+1)+(1/2)*log(theta+
sqrt(theta^2+1)))-a*((thetaC/2)*sqrt(thetaC^2+1)+(1/2)*log(thetaC+sq
rt(thetaC^2+1))) + v_0*tt;
    theta_0_initial_guess = 1;
    options = optimoptions('fsolve', 'TolX', 1e-6, 'TolFun', 1e-6, 'D
isplay', 'off');
    thetahead(1, tt+t_max+1) = fsolve(equation, theta_0_initial_gues
s, options);
    X(1,tt+t_max+1) = thetahead(1, tt+t_max+1)*a*cos(thetahead(1, tt+
t_max+1));
    Y(1,tt+t_max+1) = thetahead(1, tt+t_max+1)*a*sin(thetahead(1, tt+
t_max+1));
    BOOL(1,tt+t_max+1) = 1;

    h = h_0;
    i = 1;
    theta_0 = thetahead(i,tt+t_max+1);
    rho_0 = theta_0*a;
    equation1 = @(theta_1) rho_0^2 + (a*theta_1)^2 - 2*rho_0*a*theta_
1*cos(theta_0 - theta_1) - h^2;
    theta_1_initial_guess = theta_0 + 1;  % 初始猜测值略大于 theta_0
    thetahead(i+1, tt+t_max+1) = fsolve(equation1, theta_1_initial_gu
ess, options);
    X(i+1,tt+t_max+1) = thetahead(i+1, tt+t_max+1)*a*cos(thetahead(i+
1, tt+t_max+1));
    Y(i+1,tt+t_max+1) = thetahead(i+1, tt+t_max+1)*a*sin(thetahead(i+
```

```
1, tt+t_max+1));

    for i = 2:223
        h = h_1;
        theta_0 = thetahead(i,tt+t_max+1);
        rho_0 = theta_0*a;
        equation1 = @(theta_1) rho_0^2 + (a*theta_1)^2 - 2*rho_0*a*the
ta_1*cos(theta_0 - theta_1) - h^2;
        theta_1_initial_guess = theta_0 + 1;  % 初始猜测值略大于 theta_
0
        thetahead(i+1, tt+t_max+1) = fsolve(equation1, theta_1_initia
l_guess, options);
        X(i+1,tt+t_max+1) = thetahead(i+1, tt+t_max+1)*a*cos(thetahea
d(i+1, tt+t_max+1));
        Y(i+1,tt+t_max+1) = thetahead(i+1, tt+t_max+1)*a*sin(thetahea
d(i+1, tt+t_max+1));
    end
end

%龙头位置
for t = 1:t_max
    if v_0*t < L_first_circle
        % 第一段
        C = [rhoC*cos(thetaC), rhoC*sin(thetaC)];
        psiC = atan((C(2)-o_1(2))/(C(1)-o_1(1)));
        if (C(1)-o_1(1))<0
            psiC = psiC+pi;
        end
        if (C(2)-o_1(2))<0 && (C(1)-o_1(1))>0
            psiC = psiC+2*pi;
        end
        equation2 = @(psi_p) cos(psiC-psi_p) - 1+2*sin(v_0*t/(2*r1))^
2;
        psiP_guess = psiC - 1;
        PsiP = fsolve(equation2, psiP_guess, options);
        thetahead(1,t+t_max+1) = PsiP;
        P = o_1 + [r1*cos(PsiP ), r1*sin(PsiP )];
        X(1,t+t_max+1) = P(1);
        Y(1,t+t_max+1) = P(2);
        BOOL(1,t+t_max+1) = 3;
        continue
    end
    if v_0*t < L_first_circle + L_second_circle
        % 第二段
```

```
        S_2 = v_0*t-L_first_circle;
        psiD = atan((mid(2)-o_2(2))/(mid(1)-o_2(1)));
        if (mid(1)-o_2(1))<0
            psiD = psiD+pi;
        end
        if (mid(2)-o_2(2))<0 && (mid(1)-o_2(1))>0
            psiD = psiD+2*pi;
        end
        equation3 = @(psi_p) cos(psi_p-psiD) - 1+2*sin(S_2/(2*r2))^2;

        psiP_guess = psiD + 1;
        PsiP = fsolve(equation3, psiP_guess, options);
        thetahead(1,t+t_max+1) = PsiP;
        P = o_2 + [r2*cos(PsiP ), r2*sin(PsiP )];
        X(1,t+t_max+1) = P(1);
        Y(1,t+t_max+1) = P(2);
        BOOL(1,t+t_max+1) = 5;
        if (X(1, t+t_max+1)-mid(1))^2 + (Y(1, t+101)-mid(2))^2 < h_1^2

            BOOL(1,t+t_max+1) = 4;
        end
        continue
    end
    %盘出
    S_3 = v_0*t-L_first_circle - L_second_circle;
    equation4 = @(theta00) -a*((thetaC/2)*sqrt(thetaC^2+1)+(1/2)*log
(thetaC+sqrt(thetaC^2+1)))+a*((theta00/2)*sqrt(theta00^2+1)+(1/2)*lo
g(theta00+sqrt(theta00^2+1))) -S_3;
    theta_00_guess = 1;
    options = optimoptions('fsolve', 'TolX', 1e-6, 'TolFun', 1e-6, 'D
isplay', 'off');
    thetahead(1, t+t_max+1) = fsolve(equation4, theta_00_guess, optio
ns);
    X(1,t+t_max+1) = -thetahead(1, t+t_max+1)*a*cos(thetahead(1, t+t_
max+1));
    Y(1,t_max+1+t) = -thetahead(1, t+t_max+1)*a*sin(thetahead(1, t+t_
max+1));
    BOOL(1,t+t_max+1) = 7;
end
%%
%龙身

for t = 1:1:t_max
    for i = 2:224
```

```
        if i == 2
            h = h_0;
        else
            h = h_1;
        end
        x_i = X(i-1, t+t_max+1);
        y_i = Y(i-1, t+t_max+1);
        X_i = X(i-1, t+t_max+1);
        Y_i = Y(i-1, t+t_max+1);
        if BOOL(i-1, t+t_max+1) == 7
            P_ii = [-X(i-1, t+t_max+1), -Y(i-1, t+t_max+1)];
            rho_0 = sqrt(X(i-1, t+t_max+1)^2 + Y(i-1, t+t_max+1)^2);
            theta_0 = rho_0/a;

            equation5 = @(theta_1) rho_0^2 + (a*theta_1)^2 - 2*rho_0*a
*theta_1*cos(theta_0 - theta_1) - h^2;
            theta_5_initial_guess = theta_0 - 1;  % 初始猜测值略小于 th
eta_0
            thetahead(i, t+t_max+1) = fsolve(equation5, theta_5_initia
l_guess, options);
            X(i,t+t_max+1) = -thetahead(i, t+t_max+1)*a*cos(thetahead
(i, t+t_max+1));
            Y(i,t+t_max+1) = -thetahead(i, t+t_max+1)*a*sin(thetahead
(i, t+t_max+1));
            if sqrt(X(i,t+t_max+1)^2 + Y(i, t+t_max+1)^2)<rhoC
                [sol1, sol2] = Func(X_i, Y_i,o_2(1) ,o_2(2) , h,r2 );
                psiD = atan((mid(2)-o_2(2))/(mid(1)-o_2(1)));
                if (mid(1)-o_2(1))<0
                    psiD = psiD+pi;
                end
                if (mid(2)-o_2(2))<0 && (mid(1)-o_2(1))>0
                    psiD = psiD+2*pi;
                end
                E = [-rhoC*cos(thetaC), -rhoC*sin(thetaC)];
                psiE = atan((E(2)-o_1(2))/(E(1)-o_1(1)));
                if (E(1)-o_1(1))<0
                    psiE = psiE+pi;
                end
                if (E(2)-o_1(2))<0 && (E(1)-o_1(1))>0
                    psiE = psiE+2*pi;
                end
                psi1 = atan((sol1(1)-o_2(2))/(sol1(1)-o_2(1)));
                if (sol1(1)-o_2(1))<0
                    psi1 = psi1+pi;
```

```
                end
                if (sol1(2)-o_2(2))<0 && (sol1(1)-o_2(1))>0
                    psi1 = psi1+2*pi;
                end
                psi2 = atan((sol2(1)-o_2(2))/(sol2(1)-o_2(1)));
                if (sol2(1)-o_1(1))<0
                    psi2 = psi2+pi;
                end
                if (sol2(2)-o_1(2))<0 && (sol2(1)-o_1(1))>0
                    psi2 = psi2+2*pi;
                end
                if psi1>psiD && psi1 < psiE
                    X(i,t+t_max+1) = sol1(1);
                    Y(i,t+t_max+1) = sol1(2);
                else
                    X(i,t+t_max+1) = sol2(1);
                    Y(i,t+t_max+1) = sol2(2);
                BOOL(i, t+t_max+1) = 6;
                end
                continue;

            end
            BOOL(i, t+t_max+1) = 7;
            continue
        end
        if BOOL(i-1, t+t_max+1) == 6 || BOOL(i-1, t+t_max+1) == 5
                psii = atan((y_i-o_2(2))/(x_i-o_2(1)));
                if (y_i-o_2(2))<0 && (x_i-o_2(1))>0
                    psii = psii+2*pi;
                end
                if (x_i-o_2(1))<0
                    psii = psii+pi;
                end
                psiii = 2*asin(h/(2*r2));
                psi_i = psii-psiii;
                P = o_2 + [r2*cos(psi_i ), r2*sin(psi_i )];
                X(i, t+t_max+1) = P(1);
                Y(i, t+t_max+1) = P(2);
                BOOL(i,t+t_max+1) = 5;
            if (X(i-1, t+t_max+1)-mid(1))^2 + (Y(i-1, t+t_max+1)-mid
(2))^2 < h^2
                [sol1, sol2] = Func(X_i, Y_i,o_1(1) ,o_1(2) , h,r1 );
                psiD = atan((mid(2)-o_1(2))/(mid(1)-o_1(1)));
                if (mid(1)-o_1(1))<0
```

```
            psiD = psiD+pi;
        end
        if (mid(2)-o_1(2))<0 && (mid(1)-o_1(1))>0
            psiD = psiD+2*pi;
        end
        C = [rhoC*cos(thetaC), rhoC*sin(thetaC)];
        psiC = atan((C(2)-o_1(2))/(C(1)-o_1(1)));
        if (C(2)-o_1(2))<0 && (C(1)-o_1(1))>0
            psiC = psiC+2*pi;
        end
        if (C(1)-o_1(1))<0
            psiC = psiC+pi;
        end
        psi1 = atan((sol1(1)-o_1(2))/(sol1(1)-o_1(1)));
        if (sol1(1)-o_1(1))<0
            psi1 = psi1+pi;
        end
        if (sol1(2)-o_1(2))<0 && (sol1(1)-o_1(1))>0
            psi1 = psi1+2*pi;
        end
        psi2 = atan((sol2(1)-o_1(2))/(sol2(1)-o_1(1)));
        if (sol2(1)-o_1(1))<0
            psi2 = psi2+pi;
        end
        if (sol2(2)-o_1(2))<0 && (sol2(1)-o_1(1))>0
            psi2 = psi2+2*pi;
        end
        if psi1>psiD && psi1 < psiC
            X(i,t+t_max+1) = sol1(1);
            Y(i,t+t_max+1) = sol1(2);
        else
            X(i,t+t_max+1) = sol2(1);
            Y(i,t+t_max+1) = sol2(2);
        end
        BOOL(i, t+t_max+1) = 4;
        continue
    end
    continue
end
if BOOL(i-1, t+t_max+1) == 4 || BOOL(i-1, t+t_max+1) == 3
    psii = atan((y_i-o_1(2))/(x_i-o_1(1)));
        if (x_i-o_1(1))<0
            psii = psii+pi;
        end
```

```
                if (y_i-o_1(2))<0 && (x_i-o_1(1))>0
                    psii = psii+2*pi;
                end
                psiii = 2*asin(h/(2*r1));
                psi_i = psii+psiii;
                P = o_1 + [r1*cos(psi_i ), r1*sin(psi_i )];
                X(i, t+t_max+1) = P(1);
                Y(i, t+t_max+1) = P(2);
                BOOL(i,t+t_max+1) = 3;
            if (X(i-1, t+t_max+1)-thetaC*a*cos(thetaC))^2 + (Y(i-1, t+
t_max+1)-thetaC*a*sin(thetaC))^2 < h^2
                equation6 = @(theta)(a*theta*cos(theta )-x_i)^2 + (a*t
heta*sin(theta )-y_i)^2-h^2;
                rho_0 = sqrt(X(i-1, t+t_max+1)^2 + Y(i-1, t+t_max+1)^
2);
                theta_0 = rho_0/a;
                theta_1_initial_guess = theta_0 + 1;
                thetahead(i, t+t_max+1) = fsolve(equation6, theta_1_in
itial_guess, options);
                X(i,t+t_max+1) = thetahead(i, t+t_max+1)*a*cos(thetahe
ad(i, t+t_max+1));
                Y(i,t+t_max+1) = thetahead(i, t+t_max+1)*a*sin(thetahe
ad(i, t+t_max+1));
                BOOL(i, t+t_max+1) = 2;
                continue
            end
            continue


        end

        rho_0 = sqrt(X(i-1, t+t_max+1)^2 + Y(i-1, t+t_max+1)^2);
        theta_0 = rho_0/a;
        equation1 = @(theta_1) rho_0^2 + (a*theta_1)^2 - 2*rho_0*a*the
ta_1*cos(theta_0 - theta_1) - h^2;
        theta_1_initial_guess = theta_0 + 1;  % 初始猜测值略大于 theta_
0
        thetahead(i, t+t_max+1) = fsolve(equation1, theta_1_initial_g
uess, options);
        X(i,t+t_max+1) = thetahead(i, t+t_max+1)*a*cos(thetahead(i, t
+t_max+1));
        Y(i,t+t_max+1) = thetahead(i, t+t_max+1)*a*sin(thetahead(i, t
+t_max+1));
        BOOL(i, t+t_max+1) = 1;
```

```
    end



end
%%
N=224;
axis equal
hold on
t=17;
plot(X(1:N,t_max+1+t), Y(1:N,t_max+1+t))
scatter(X(1:N,t_max+1+t), Y(1:N,t_max+1+t))
scatter(thetaC*a*cos(thetaC),thetaC*a*sin(thetaC))

scatter(o_1(1), o_1(2));
scatter(o_2(1), o_2(2));

rectangle('Position',[o_1(1)-r1,o_1(2)-r1,2*r1,2*r1],'Curvature',[1,
1],'EdgeColor','m')
rectangle('Position',[o_2(1)-r2,o_2(2)-r2,2*r2,2*r2],'Curvature',[1,
1],'EdgeColor','m')

hold off
%%
Result = zeros(448, 2*t_max+1);
for i = 1:224
    Result(2*i-1, :) = X(i,:);
    Result(2*i, :) = Y(i,:);
end

velocity = zeros(224,2*t_max+1);
velocity(1,:) = 1;
for t = 1:2*t_max+1
    for j = 2:224
        if(BOOL(j,t) == 0 || BOOL(j,t) == 1)
            ku = [Y(j,t)-Y(j-1,t), X(j-1, t)-X(j, t)];
            thetai = sqrt(X(j-1,t)^2 + Y(j-1,t)^2)/a;
```

```
            thetaii = sqrt(X(j,t)^2 + Y(j,t)^2)/a;
            ki = [a*cos(thetai) - a*thetai*sin(thetai), a*thetai*cos(t
hetai) + a*sin(thetai)];
            kii = [a*cos(thetaii) - a*thetaii*sin(thetaii), a*thetaii*
cos(thetaii) + a*sin(thetaii)];
            ni = ki/norm(ki);
            nii = kii/norm(kii);
            nu = ku/norm(ku);
            lu = [nu(2), -nu(1)];
            velocity(j,t) = abs(velocity(j-1, t)*dot(ni, lu)/dot(nii,
lu));
            continue
        end
        if(BOOL(j,t) == 2)
            ku = [Y(j,t)-Y(j-1,t), X(j-1, t)-X(j, t)];
            thetaii = sqrt(X(j,t)^2 + Y(j,t)^2)/a;
            kii = [a*cos(thetaii) - a*thetaii*sin(thetaii), a*thetaii*
cos(thetaii) + a*sin(thetaii)];
            ki = [Y(j-1,t)-o_1(2), o_1(1)-X(j-1,t)];
            ni = ki/norm(ki);
            nii = kii/norm(kii);
            nu = ku/norm(ku);
            lu = [nu(2), -nu(1)];
            velocity(j,t) = abs(velocity(j-1, t)*dot(ni, lu)/dot(nii,
lu));
            continue
        end
        if(BOOL(j,t) == 3)
            ku = [Y(j,t)-Y(j-1,t), X(j-1, t)-X(j, t)];
            ki = -[Y(j-1,t)-o_1(2), o_1(1)-X(j-1,t)];
            kii = -[Y(j,t)-o_1(2), o_1(1)-X(j,t)];
            ni = ki/norm(ki);
            nii = kii/norm(kii);
            nu = ku/norm(ku);
            lu = [nu(2), -nu(1)];
            velocity(j,t) = abs(velocity(j-1, t)*dot(ni, lu)/dot(nii,
lu));
            continue
        end
        if(BOOL(j,t) == 4)
            ku = [Y(j,t)-Y(j-1,t), X(j-1, t)-X(j, t)];
            ki = -[Y(j-1,t)-o_2(2), o_2(1)-X(j-1,t)];
            kii = [Y(j,t)-o_1(2), o_1(1)-X(j,t)];
            ni = ki/norm(ki);
```

```
            nii = kii/norm(kii);
            nu = ku/norm(ku);
            lu = [nu(2), -nu(1)];
            velocity(j,t) = abs(velocity(j-1, t)*dot(ni, lu)/dot(nii,
lu));
            continue
        end
        if(BOOL(j,t) == 5)
            ku = [Y(j,t)-Y(j-1,t), X(j-1, t)-X(j, t)];
            ki = [Y(j-1,t)-o_2(2), o_2(1)-X(j-1,t)];
            kii = [Y(j,t)-o_2(2), o_2(1)-X(j,t)];
            ni = ki/norm(ki);
            nii = kii/norm(kii);
            nu = ku/norm(ku);
            lu = [nu(2), -nu(1)];
            velocity(j,t) = abs(velocity(j-1, t)*dot(ni, lu)/dot(nii,
lu));
            continue
        end
        if(BOOL(j,t) == 6)
            ku = [Y(j,t)-Y(j-1,t), X(j-1, t)-X(j, t)];
            kii = [Y(j-1,t)-o_2(2), o_2(1)-X(j-1,t)];
            thetai = sqrt(X(j-1,t)^2 + Y(j-1,t)^2)/a;
            ki = -[a*cos(thetai) - a*thetai*sin(thetai), a*thetai*cos
(thetai) + a*sin(thetai)];
            ni = ki/norm(ki);
            nii = kii/norm(kii);
            nu = ku/norm(ku);
            lu = [nu(2), -nu(1)];
            velocity(j,t) = abs(velocity(j-1, t)*dot(ni, lu)/dot(nii,
lu));
            continue
        end
        if(BOOL(j,t) == 7)
            ku = [Y(j,t)-Y(j-1,t), X(j-1, t)-X(j, t)];
            thetai = sqrt(X(j-1,t)^2 + Y(j-1,t)^2)/a;
            thetaii = sqrt(X(j,t)^2 + Y(j,t)^2)/a;
            ki = -[a*cos(thetai) - a*thetai*sin(thetai), a*thetai*cos
(thetai) + a*sin(thetai)];
            kii = -[a*cos(thetaii) - a*thetaii*sin(thetaii), a*thetaii
*cos(thetaii) + a*sin(thetaii)];
            ni = ki/norm(ki);
            nii = kii/norm(kii);
            nu = ku/norm(ku);
```

```
            lu = [nu(2), -nu(1)];
            velocity(j,t) = abs(velocity(j-1, t)*dot(ni, lu)/dot(nii,
lu));
            continue
        end
    end
end
%%
% Result_v = zeros(224, 2*t_max);
% for t = 1:2*t_max
%     for i = 1:224
%         Result_v(i,t) = sqrt((X(i,t+1)-X(i,t))^2+(Y(i,t+1)-Y(i,t))^
2);
%     end
% end
% hold on
% plot(-t_max:t_max-1, Result_v(100, :))
% plot(-t_max:t_max-1, Result_v(150, :))
% plot(-t_max:t_max-1, Result_v(200, :))

[x,y ] = meshgrid(-t_max:t_max, 1:224);
surf(x,y,velocity)
```