基于螺旋路径的动态位置与速度模型

摘要

针对“板凳龙”盘旋过程，本文建立了基于螺旋路径的动态位置与速度模型。通过合理假设，模拟盘入与调头过程中可能发生的碰撞风险，在构建螺旋线与调头空间的几何模型基础上，计算各节板凳的实时位置与速度。多点碰撞检测确保龙头与板凳边缘在行进中不发生碰撞。在满足各边界约束条件下，优化螺距与路径长度，利用迭代算法和几何约束确定最小螺距与最优路径，确保龙头以最大安全速度顺利完成调头，为实际表演过程提供理论依据和参考。

针对问题一，要求计算板凳龙舞龙队在 300 秒内，每秒钟各节板凳的前后把手的位置和速度情况。本文建立了舞龙队沿螺旋线运动的数学模型，使用极坐标方程描述轨迹，递推计算每节板凳的位置和速度。通过微分方程和差分方程，计算出各节板凳在不同时间点的精确位置和速度。位置和速度的具体结果见表 1 和表 2。

针对问题二，首先需要建立龙头和板凳边缘的碰撞模型。通过跟踪龙头的运动轨迹，结合其四个角的坐标，使用相对位置到全局坐标系的转换方法，分析龙头在盘入过程中的动态位置。模型通过极坐标与局部坐标系之间的转换，判定龙头轨迹与板凳边缘的相对位置变化，计算两者是否发生接触，确保在最大安全时刻前龙身与龙头无碰撞风险。最终确定碰撞时间为 422 s。位置和速度的具体结果见表 3。

针对问题三，需要综合考虑螺旋线的螺距与板凳龙的碰撞问题。首先求出螺旋线与调头空间的边界精确相切的螺距距离。在此基础上，通过迭代调整这些螺距，模拟龙头在螺旋线上的运动过程，验证是否与调头空间外部发生碰撞。经过多次优化，最终得出满足相切条件并避免碰撞的最小螺距为 39 cm。

针对问题四，本文在优化舞龙队调头路径时，通过结合随机搜索和几何约束条件，最小化路径长度，并确保路径在螺旋线与调头圆弧的切点处保持光滑过渡和曲率连续性。通过递归优化算法，确保调头路径不仅满足相切条件，还能最大程度缩短路径长度。在问题三建立的模型基础上进行碰撞检测，最终得出符合要求的最优调头路径长度为 13.3m。位置和速度的具体结果见表 4 和表 5。

针对问题五，在问题四的基础上确定龙头的最大行进速度。通过动态模拟舞龙队各节的速度变化，确保其速度始终在安全范围内（不超过 2 m/s）。通过逐步迭代调整龙头速度，分析螺旋线半径和节距对各节速度的影响，最终得出龙头的最大行进速度为 1.25 m/s。

关键词：动态位置与速度模型 微分方程 螺旋路径 碰撞检测 递归优化算法

一、 问题重述

1.1 问题背景

板凳龙是我国浙闽地区元宵及其他佳节期间的民俗活动，寄托着当地人民对于美好生活的向往[1]。板凳龙表演灵动壮观，通过将数十条至上百条板凳首尾相连可以组成绵延舞动的板凳龙。板凳龙表演时，舞龙队呈圆盘状，龙头在队首领头，龙身和龙尾随之盘旋。在舞龙队能够自由盘旋出入的前提下，板凳龙表演的观赏性随盘旋所占面积越小、行进速度越高而提高。

现有一板凳龙由 223 节板凳（队首 1 节龙头，后接 221 节龙身以及 1 节龙尾）组成。其中，龙头板凳长度 341 cm，其余板长均为 220 cm，宽度方面所有板凳宽度均为 30cm，每节板凳上分布有 2 个位于临近板头 27.5 cm 的孔径为 5.5 cm 的通孔用于安插把手以连接前后相邻的两条板凳。

1.2 问题提出

1、该板凳龙舞龙队表演时，所有把手的中心线均位于在一条螺距为 55 cm 的螺线上，龙头前把手以 1 m/s 的速度自螺线第 16 圈 A 点处起带领舞龙队以顺时针方向向内盘入。现要求通过数学建模的方法求解 300 s 内每秒板凳龙各节前把手和龙尾后把手的位置和速度情况。

2、随着舞龙队持续顺时针向内盘入，板凳间最终会因内部空间不足而发生碰撞。在此需要求解舞龙队的终止时刻（保证板凳不会相碰不能继续盘入的时间），并求解处此时舞龙队的板凳龙各节前把手和龙尾后把手的位置和速度情况。

3、当舞龙队由盘入切换到盘出时，其盘旋行进方向将由顺时针转为逆时针，这对舞龙队内部中心区域（即调头空间）的面积提出了一定的要求。现要求确定当调头空间为一以螺线中心为圆心的直径为 9 m 的圆形区域时能允许龙头前把手顺对应螺线盘进调头空间边界的最小螺距。

4、盘入螺线螺距为 1.7 m，盘出与盘入螺线以螺线中心呈对称分布，在问题 3 所述的调头空间内调头时，其调头路径为两端相切的圆弧（前一段圆弧半径为后一段的 2 倍）连接而成的 S 形，且于盘入和盘出螺线均相切。要求在保证各部分相切的前提下调整圆弧减短调头路径轨迹。并在龙头前把手为 1 m/s 移动时，以调头开始时刻为 0 时刻求解 -100~100s 内每秒板凳龙各节前把手和龙尾后把手的位置和速度情况。

5、在前一问路径和龙头行进速度的基础上求解龙头最大行进速度保证龙身各把手速度保持在 2 m/s 内。

二、 问题分析

2.1 问题一分析

问题一要求计算在 300 秒内，每秒钟舞龙队各前把手及龙尾后把手的位置和速度。其关键在于利用螺旋线方程描述舞龙队的运动轨迹。本文首先建立螺旋线的极坐标方程。随着时间推移，龙头沿着螺旋线运动，通过每秒钟更新龙头的极角与半径，再根据相邻节板凳的距离，建立板凳间的递推公式计算每一节龙身和龙尾的位置和速度。题干中所求的位置更新基于极坐标方程，而速度则通过位置随时间的变化率计算。

2.2 问题二分析

问题二要求确定舞龙队盘入的截止时刻，以避免板凳间发生碰撞，确定此时舞龙队的位置和速度。经分析易知，首次碰撞将发生在龙头与外圈沿着螺旋线盘入的龙身上。

以龙头上前后两个孔的连线为 y 轴，垂直于连线方向为 x 轴建立坐标系，可以确定龙头的四个角坐标。考虑孔运动过程中更新原点和方向向量，以实现从相对位置到全局坐标系中绝对位置的转换。同时画出舞龙队在盘入过程中板凳边缘的轨迹。结合问题一相关的计算公式，建立碰撞模型，碰撞判断条件为边缘轨迹与四个角坐标围成的区域面积相交。

2.3 问题三分析

问题三要求确定一个最小的螺距，使得龙头前把手能够沿螺旋线顺利进入直径为 9 米的调头空间边界。首先，本文针对螺旋线的路径与调头空间进行判断，确保两者相切。接着，通过模拟龙头沿螺旋线运动的过程，判断其是否与调头空间外的任何部分发生碰撞。如果出现碰撞，则需要调整螺距，并通过迭代不断优化，直至找到既满足进入调头空间的相切条件、又避免碰撞的最小螺距。

2.4 问题四分析

问题四要求调整调头路径的圆弧，使其更短，同时保持相切条件下，计算不同时间点的位置和速度。调头路径由两段相切圆弧构成，螺旋线与第一段圆弧、第二段圆弧分别相切，并要求切点处曲率连续性，确保路径光滑过渡。其次，路径优化过程采用迭代方法，通过随机搜索生成候选切点，满足几何约束（如距离条件）后，利用极坐标形式计算两个螺旋线的切点，并最小化几何目标函数。最后通过结合碰撞检测与动态计算，得到符合最优解的调头路径。

2.5 问题五分析

问题五要求确定龙头的最大行进速度，同时保证舞龙队各节的速度不超过 2 m/s。舞龙队的盘入和盘出过程中，由于螺旋线的变化的几何特性，速度在不同点上的变化存在显著差异。当龙头的速度较快时，龙尾及中部节数的速度可能会大幅增加。因此，在计算中需要逐步调整龙头速度，同时监控各点的速度，在螺线半径、节距等参数的约束条件下，动态模拟每个点的速度，从而得出龙头的最大安全速度。

三、 模型假设

(1) 忽略所有摩擦力，风力即操作者的操作误差，各把手的中心线均位于螺线上，人在行进过程过的路径视作精准的数学曲线。
(2) 每条板凳及开孔的尺寸与位置确定且不存在制造工艺误差，板凳间通过通过刚性连接且忽略由于受力而产生的形变。
(3) 龙头前把手以恒定匀速 v_0 延螺线方向行进。
(4) 忽略板凳的厚度，将各板凳视作在同一水平面上的矩形，不存在在空间高度上的差别。
(5) 舞龙队行进表演开始行进前，队伍排列在等距螺线上等待。

四、 符号说明

符号	含义	单位
r	极坐标系中的半径	m
θ	极坐标系中的角度	rad
d	板凳上相邻两孔的距离	m
p	螺旋线的螺距	m

t	时间	s
x	笛卡尔坐标系下的横坐标	m
y	笛卡尔坐标系下的纵坐标	m
W	龙头板凳的宽度	m
L	龙头板凳的长度	m
$\vec{v}_1$	螺旋线上的点到多边形边的向量	—

五、 问题一模型的建立与求解

5.1 模型准备

阿基米德螺线，又名等距螺线，描绘的是一个点在以恒定速度远离某一固定点的同时，以恒定角速度绕该点旋转所形成的轨迹[2]，该曲线可以通过极坐标方程表示为：

$$r = a + b\theta \tag{1}$$

式中r表示极坐标系中的半径，θ是极角，a和b是常数。

笛卡尔坐标(x,y)和极坐标之间的关系是：

$$\begin{cases} x = r\cos(\theta) \\ y = r\sin(\theta) \end{cases} \tag{2}$$

将(1)式极坐标方程代入笛卡尔坐标的公式(2)中，我们笛卡尔坐标系下的等距曲线：

$$\begin{cases} x = (a + b\theta)\cos(\theta) \\ y = (a + b\theta)\sin(\theta) \end{cases} \tag{3}$$

5.2 模型建立

现有一由 223 节板凳组成的舞龙队，存在 224 个连接孔（本文在建立模型过程中将孔视作点），各孔均分布在螺距为 0.55 米的螺旋线上，且舞龙队运动时的切向速度为 1 m/s，第一个孔和第二个孔之间的距离固定为 2.86 米，其余各孔间距固定为 1.65 米。本文将推导出所有孔的坐标和速度随时间变化的数学公式，并通过微分方程和差分方程建立数学模型来描述这些孔的位置和运动。

图 1 为问题一的具体流程图，本文首先建立螺旋线极坐标方程，并在此坐标系下，通过每秒更新龙头的极角和极径，根据相邻板的距离，计算舞龙队位置和速度信息，并将极坐标系下的结果转换回笛卡尔坐标系。

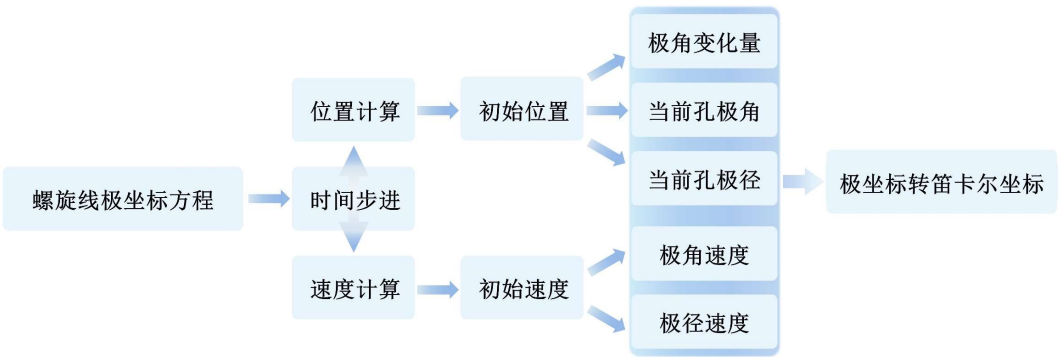

图 1 问题一建模流程图

5.2.1 孔的初始位置计算

孔的位置计算分为两个主要部分：初始位置的确定和后续孔位置的计算。

由题干可得，初始极角为 32π，同时初始极径r_0为 8.8m。对于后续的每个孔的位置，本文通过前一个孔的极坐标和固定距离（弧长）计算。

每个孔之间的距离d是固定的，极角变化量可通过如下公式计算：

$$\Delta\theta = \frac{d}{r_{\mathrm{p}}} \tag{4}$$

式中，r_{p}表示前一个孔的极径。

基于此，当前孔的极角可基于前一个极角为 p 得到为

$$\theta_{\mathrm{c}} = \theta_{\mathrm{p}} + \Delta\theta \tag{5}$$

当前孔的极角r_{c}可更新为：

$$r_{\mathrm{c}} = r_0 + p \cdot \theta_{\mathrm{c}} \tag{6}$$

式中p为螺距。

5.2.2 龙头的位置方程

龙头的运动是沿着螺距为 0.55 米的螺旋线，通过使用极坐标和参数方程表示龙头的所在位置的半径r随时间r的变化可表示为：

$$r(t) = r_0 + p \cdot t \tag{7}$$

式中，r_0为初始半径，为 8.8 米，即龙头的初始位置为(8.8,0)，p为螺距，为 0.55 米，t为时间，单位为秒

龙头的角度$\theta(t)$ 随时间的变化满足微分方程：

$$\frac{d\theta(t)}{dt} = \frac{v_{\mathrm{h}}}{r(t)} = \frac{1}{r(t)} \tag{8}$$

式中，v_h为龙头的速度，大小为 1m/s。

通过积分得到角度随时间变化的方程：

$$\theta(t) = \frac{v_{\mathrm{h}}}{p}\ln\left(1 + \frac{p \cdot t}{r_0}\right) \tag{9}$$

龙头的坐标为$((x_1(t), y_1(t))$：

$$\begin{cases} x_1(t) = r(t) \cdot \cos(\theta(t)) \\ y_1(t) = r(t) \cdot \sin(\theta(t)) \end{cases} \tag{10}$$

5.2.3 第二个点的位置方程

第二个点与龙头的距离保持 2.86 米，并且也沿着同一条螺旋线运动，设第二个点位置坐标为$((x_2(t), y_2(t))$，则有：

$$\sqrt{(x_2(t) - x_1(t))^2 + (y_2(t) - y_1(t))^2} = 2.86 \tag{11}$$

在此，本文引入一个时间差使得第二个点的位置滞后于龙头，并保持与龙头的固定距离：

$$r_2(t) = r(t - \Delta t_1) \tag{12}$$

式中，Δt_1 通过龙头的速度和两点间的固定距离 2.86 米来确定。

故第二个点的坐标可表示为：

$$\begin{cases} x_2(t) = r_2(t) \cdot \cos(\theta_2(t)) \\ y_2(t) = r_2(t) \cdot \sin(\theta_2(t)) \end{cases} \tag{13}$$

通过微分方程可将第二个点的运动和时间的关系描述为：

$$\frac{dx_2(t)}{dt}=\frac{dr_2(t)}{dt}\cdot\cos(\theta_2(t))-r_2(t)\cdot\sin(\theta_2(t))\cdot\frac{d\theta_2(t)}{dt} \tag{14}$$

$$\frac{dy_2(t)}{dt}=\frac{dr_2(t)}{dt}\cdot\sin(\theta_2(t))+r_2(t)\cdot\cos(\theta_2(t))\cdot\frac{d\theta_2(t)}{dt} \tag{15}$$

5.2.4 队伍中其余点的位置方程

由于自第三个点开始，所有点与前一个点的距均为 1.65 米，故通过差分方程可递推出每个点的坐标为。

第i个点与第$i-1$个点的距离固定为 1.65 米，故有：

$$\sqrt{(x_i(t)-x_{i-1}(t))^2+(y_i(t)-y_{i-1}(t))^2}=1.65 \tag{16}$$

本文根据前一个点的坐标可以推导出当前点的位置。假设第i个点在第$i-1$个点的运动方向上，可以通过调整角度来递推出位置。

由于第个点之后于第各店，故有：

$$\theta_i(t)=\theta_{i-1}(t-\Delta t_{i-1}) \tag{17}$$

式中，Δt_{i-1}是第i个点相对前一个点的时间差，用以保证距离为 1.65 米。

递推公式（差分方程）可表述为：

$$\begin{cases}x_i(t)=x_{i-1}(t)-d\cdot\cos(\theta_{i-1}(t))\\ y_i(t)=y_{i-1}(t)-d\cdot\sin(\theta_{i-1}(t))\end{cases} \tag{18}$$

式中，d为两点间的固定距离，为 1.65 米。

5.2.5 速度计算

如图 2 所示，点的速度可以通过位置随时间的变化来计算。

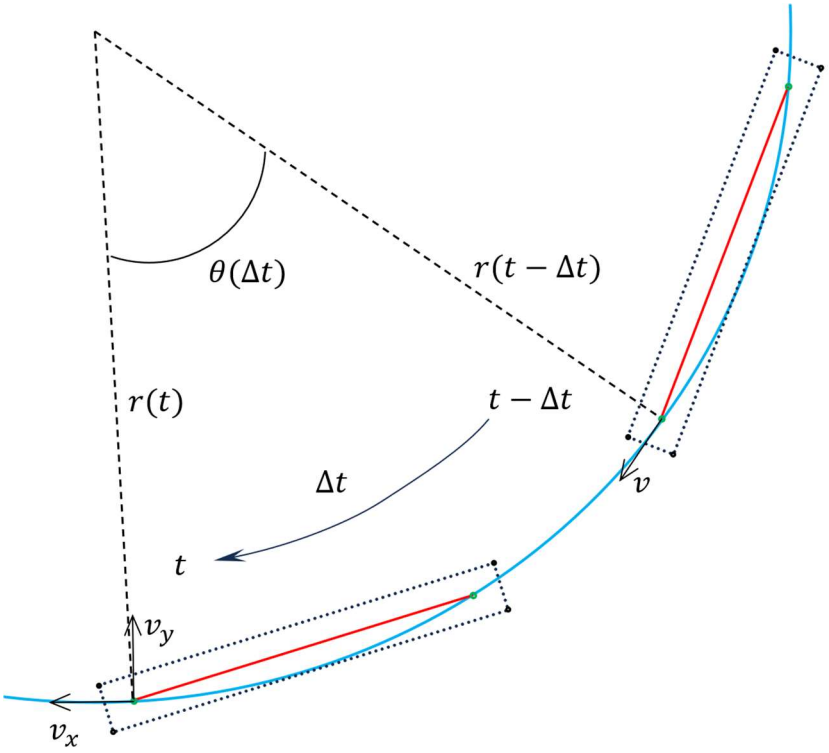

图 2 速度计算示意图

假设Δt是时间步长，则第i个点的分速度为：

$$\begin{cases} v_{x_i}(t) = \dfrac{x_i(t) - x_i(t-\Delta t)}{\Delta t} \\ v_{y_i}(t) = \dfrac{y_i(t) - y_i(t-\Delta t)}{\Delta t} \end{cases} \tag{19}$$

将式(20)中分速度合成可得总速度为：

$$v_i(t) = \sqrt{v_{x_i}(t)^2 + v_{y_i}(t)^2} \tag{20}$$

5.3 模型求解

通过本文所述建立的螺旋线极坐标方程，更新每秒龙头的位置信息，根据相邻板的距离，本文得以计算出舞龙队位置和速度信息，并将极坐标系下的结果转换回笛卡尔坐标系，舞龙队行进终止（t=300 s）时舞龙队的队伍分布情况如图 3 所示。

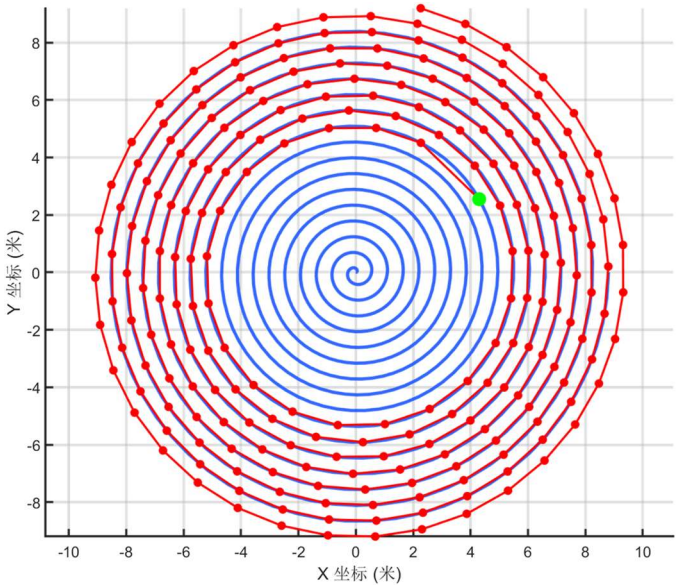

图 3 舞龙队行进终止时队伍排列情况

舞龙队不同位置不同时刻的部分具体位置和速度信息如表 1 和表 2 所示，完整数据可见附件 result1.xlsx。

表 1 问题 1 舞龙队位置结果

	0 s	60 s	120 s	180 s	240 s	300 s
龙头 x(m)	8.800000	5.812412	-4.062600	-2.955075	2.527542	4.479893
龙头 y(m)	0.000000	-5.753460	-6.311043	6.086924	-5.370861	2.149605
第 1 节龙身 x(m)	8.366429	7.459393	-1.432829	-5.217630	4.757750	2.625658
第 1 节龙身 y(m)	2.819050	-3.427540	-7.401693	4.366169	-3.619707	4.277820
第 51 节龙身 x(m)	-9.513257	-8.677045	-5.518904	2.937810	5.906739	-6.261461
第 51 节龙身 y(m)	1.405184	2.575037	6.396068	7.223870	-3.920949	0.712652
第 101 节龙身 x(m)	2.774790	5.598256	5.282236	1.786615	-5.049271	-6.063279
第 101 节龙身 y(m)	-9.965327	-8.069157	-7.615435	-8.494153	-6.268503	4.180160
第 151 节龙身 x(m)	10.910182	6.847363	2.573065	1.201768	3.197918	7.223371
第 151 节龙身 y(m)	1.582525	8.005877	9.685256	9.403482	8.311952	4.072396
第 201 节龙身 x(m)	4.900716	-6.360684	-10.592834	-9.413138	-7.673265	-7.710625
第 201 节龙身 y(m)	10.585256	9.222251	1.656484	-3.973556	-5.912867	-4.881854
龙尾（后）x(m)	-5.694251	7.075357	11.004596	7.634283	3.610253	2.274883
龙尾（后）y(m)	-10.489628	-9.046518	0.492662	7.246073	9.338004	9.192500

表 2 问题 1 舞龙队速度结果

	0 s	60 s	120 s	180 s	240 s	300 s
龙头(m/s)	1.000057	1.000000	1.000000	0.999999	0.999998	0.999825
第 1 节龙身(m/s)	1.000027	0.999961	0.999945	0.999916	0.999857	0.999538
第 51 节龙身(m/s)	0.999789	0.999662	0.999538	0.999330	0.998939	0.997957
第 101 节龙身(m/s)	0.999616	0.999453	0.999269	0.998970	0.998434	0.997223
第 151 节龙身(m/s)	0.999484	0.999299	0.999077	0.998727	0.998113	0.996800
第 201 节龙身(m/s)	0.999380	0.999180	0.998934	0.998551	0.997892	0.996524
龙尾（后）(m/s)	0.999341	0.999136	0.998882	0.998488	0.997815	0.996431

5.4 灵敏度分析

在问题一模型求参的过程中，通过模拟舞龙队在螺旋线上随时间的运动轨迹，重点分析了不同龙头速度对舞龙整体运动模式的影响。通过对不同时间点（0 到 300 秒）以及舞龙队中关键节点的位移和速度变化进行跟踪，我们得以清晰观察龙头速度的变化对整体运动的影响。

图 4 为灵敏度分析结果，随着龙头速度的增加，龙头、龙身、以及龙尾等关键节点的运动轨迹表现出明显的差异。具体而言，龙头和龙身的位移在更高的速度下显著增大，而龙尾的速度波动也随之加剧。速度的提升不仅影响了各节点的运动范围，还对整体的协调性产生了影响。

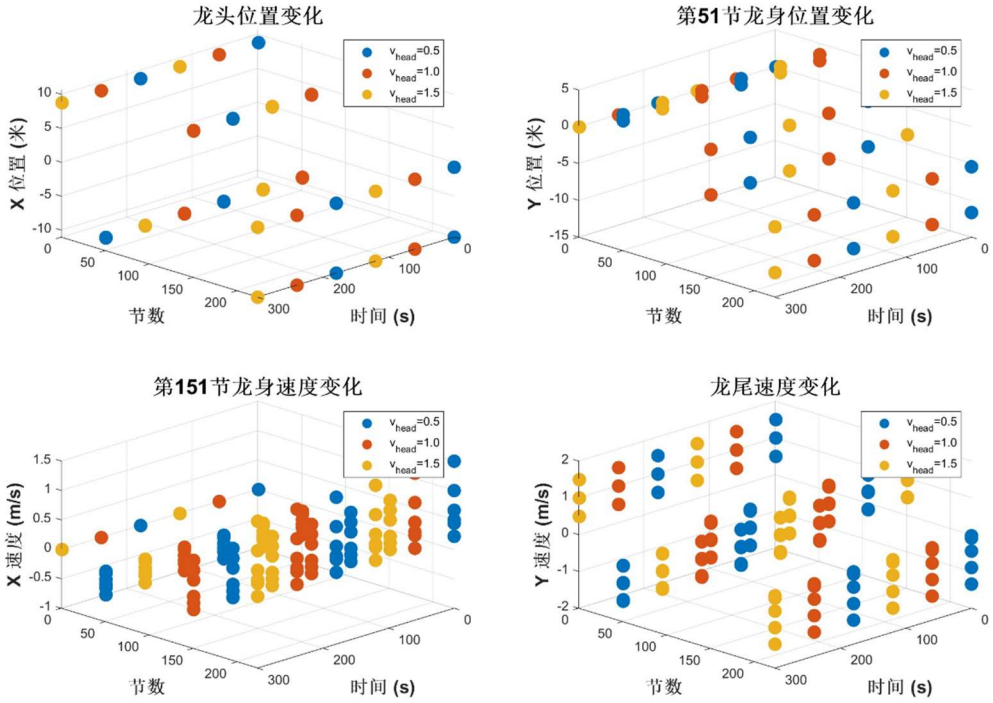

图 4 舞龙队关键点位置与速度随时间的变化

六、 问题二模型的建立与求解

6.1 模型准备

图 5 展示了问题二的建模流程图。经分析易得，初次碰撞发生于龙头盘入时其与外圈盘入的龙身之间，故本文一方面建立了龙身边缘的轨迹方程，同时建立了龙头坐标系，通过龙头四个顶点确立龙头在龙头坐标系下碰撞的区域后，将其装换到龙身轨迹方程的同一坐标系下进行碰撞判断。

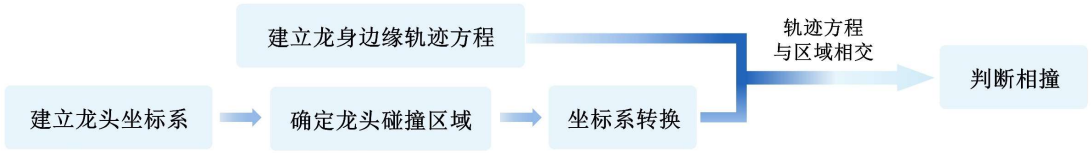

图 5 问题二建模流程图

6.2 模型建立

6.2.1 坐标系建立与龙头初始条件

如图 6 所示，在建立初始条件时，假设我们已将龙头的运动分解到一个等距螺线平面坐标系中，建立一龙头坐标系，其中龙头的前、后孔的连线方向定为局部坐标系的 y 轴，垂直于此连线的方向为 x 轴。由于舞龙队的运动具有强烈的旋转和螺旋轨迹特性，因此使用龙头坐标系便于计算龙头及后续板凳的几何位置和旋转。

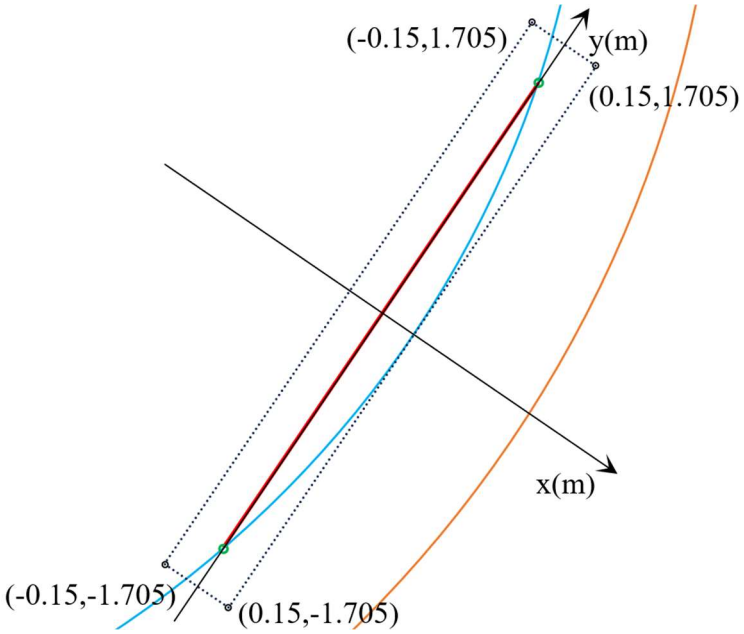

图 6 龙头坐标系的建立及碰撞区域位置

本文在此设龙头的四个角坐标在局部坐标系中的初始坐标为：

$$\begin{cases} (x'_1, y'_1) & = \left(\frac{W}{2}, \frac{L}{2}\right) \\ (x_2', y_2') & = \left(-\frac{W}{2}, \frac{L}{2}\right) \\ (x_3', y_3') & = \left(-\frac{W}{2}, -\frac{L}{2}\right) \\ (x'_4, y'_4) & = \left(\frac{W}{2}, -\frac{L}{2}\right) \end{cases} \tag{21}$$

式中，W 是龙头板凳的宽度，为 0.3 米，L 是龙头的板长，为 3.41 米。

6.2.2 坐标系的转换过程

在盘入过程中，龙头和每节龙身的方向和位置随时间不断变化，因此需要将每一节板凳的局部坐标转换为全局坐标。设局部坐标系的原点为龙头前把手的位置$(x_0(t), y_0(t))$，龙头的旋转角$\theta(t)$，则局部坐标系中的任意一点(x', y')转换到全局坐标系的公式为：

$$\begin{cases} x = x_0(t) + x'\cos(\theta(t)) - y'\sin(\theta(t)) \\ y = y_0(t) + x'\sin(\theta(t)) + y'\cos(\theta(t)) \end{cases} \tag{22}$$

6.2.3 碰撞模型的建立

为了判断舞龙队盘入过程中是否发生碰撞，我们引入了板凳间距和几何相交的判定标准，具体的碰撞检测为多边形相交判断。

建立一个龙身边缘的运动轨迹方程，即图6中橙色的等距螺旋线，其螺距为0.55米，该线上的每个点都需要检测是否位于四点围成的多边形内部。多边形由龙头的四个角坐标(x_1,y_1)、(x_2,y_2)、(x_3,y_3)和(x_4,y_4)组成。

数学上，检测螺旋线的点是否位于多边形内相当于计算该点与多边形每条边的叉积。对于每个点(x_i,y_i)，其到多边形边的叉积计算如下：

$$\vec{v}_1 \times \vec{v}_2 = (x_j - x_i) \cdot (y_k - y_i) - (y_j - y_i) \cdot (x_k - x_i) \tag{23}$$

式中，$\vec{v}_1 = (x_j - x_i, y_j - y_i)$和$\vec{v}_2 = (x_k - x_i, y_k - y_i)$为点到多边形边的向量。如果所有叉积的符号一致，则该点位于多边形内部，表示发生碰撞。

碰撞的判定条件为当螺旋线上的某个点位于多边形内时，判定此时发生碰撞。

6.3 模型求解

由题干可得龙头两孔之间的距离为3.41m，板凳宽为0.3m，基于两孔的中点建立坐标系可得四个点的坐标分别为(0.15,1.705)，(-0.15,1.705)，(-0.15,-1.705)，(0.15,-1.705)。根据上述建立的角坐标及龙头边缘轨迹方程，可以得结果如图7所示经过422 s后龙头与龙身发生碰撞。

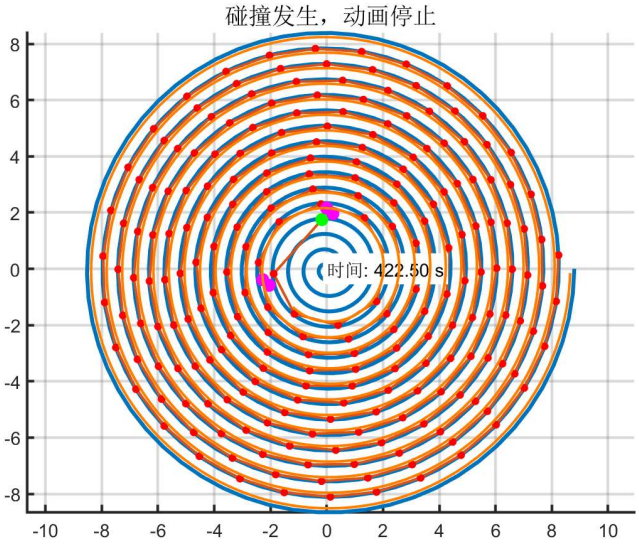

图 7 碰撞发生时舞龙队队列分布情况

碰撞发生时舞龙队位置和速度部分信息表3所示，完整信息可见附件result3.xlsx。

表 3 问题2舞龙队位置速度结果

	x坐标(m)	y坐标(m)	速度（m/s）
龙头	-1.269813	1.301454	1.000000
第1节龙身	-1.485627	-1.263916	1.000001
第51节龙身	1.728073	-3.910194	1.000002
第101节龙身	-5.329841	2.092845	0.999999
第151节龙身	-6.801250	-1.024180	0.999999
第201节龙身	-2.683383	7.390995	1.000004
龙尾（后）(m/s)	8.178962	1.149673	0.999996

6.4 灵敏度分析

在问题二的基础上本文进一步分析了龙头速度对舞龙队发生碰撞时间的影响。通过设定不同的龙头速度范围（1 m/s 到 3 m/s），我们模拟了舞龙队在螺旋路径上的运动，动态更新各节点的位置，并监控相邻节点的距离变化。当相邻节点之间的距离小于设定的安全距离时，即判定为发生碰撞，系统自动记录该时间点。结果如图 8 表明，随着龙头速度的增加，碰撞时间逐渐缩短，表明更高的速度使得节点更快地接近，并导致更早的碰撞发生。

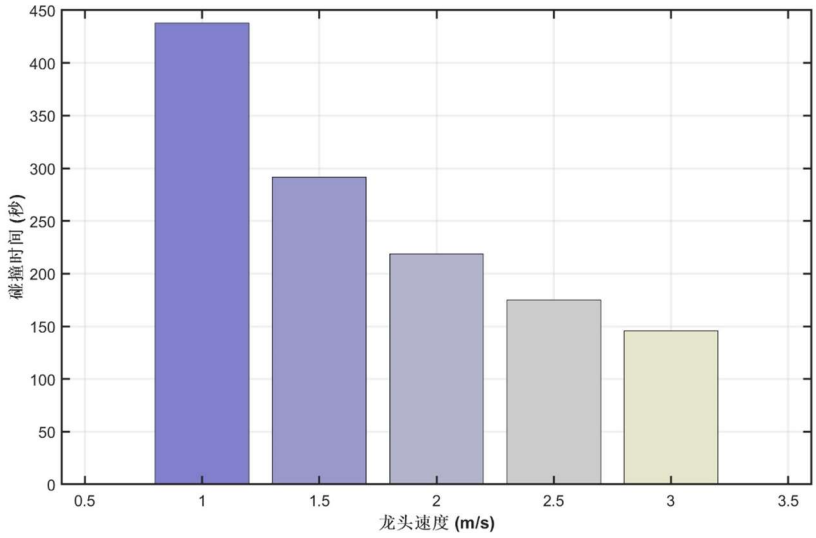

图 8 碰撞时间随龙头速度的变化

七、 问题三模型的建立与求解

7.1 模型准备

要确定一个最小的螺距，使得龙头前把手能够沿螺旋线顺利进入直径为 9 米的调头空间边界，本文首先对螺旋线的路径与调头空间进行判断，确保螺旋线与圆形区域的相切。随后，在一螺距情况下模拟龙头运动的过程以判断调头时是否会发生碰撞。若发生碰撞则持续迭代优化螺距，直至找到既满足进入调头空间的相切条件、又避免碰撞的最小螺距并将其输出。问题三模型建立流程图如图 9。

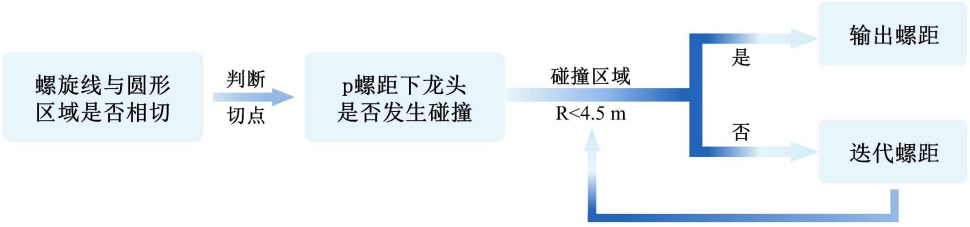

图 9 问题三建模流程图

7.2 模型建立

7.2.1 调头空间的定义

调头空间可以简单地建模为一个直径为 9 米的圆，其边界可以用如下方程来描述：

$$(x-x_c)^2+(y-y_c)^2=(4.5)^2 \tag{24}$$

式中，(x_c,y_c)为圆的中心坐标，其直径为 9 米

7.2.2 相切条件的判定

为了确保螺旋线能够顺利进入调头空间，螺旋线的路径与调头空间的边界必须相切。

相切意味着螺旋线在与调头空间接触的点上，两者的切线方向一致。数学上，这意味着螺旋线在某一时刻t_0满足距离条件和切线条件。

7.2.3 距离条件

螺旋线在t_0时的点 $(x(t_0),y(t_0))$ 距离调头空间的边界恰好为 4.5 米，即：

$$(x(t_0)-x_c)^2+(y(t_0)-y_c)^2=(4.5)^2 \tag{25}$$

7.2.4 切线条件

螺旋线的速度向量与调头空间边界的法向量平行，即：

$$\frac{dx}{dt}(t_0)\cdot(x(t_0)-x_c)+\frac{dy}{dt}(t_0)\cdot(y(t_0)-y_c)=0 \tag{26}$$

7.2.5 碰撞判定与螺距优化

为避免碰撞，除确保螺旋线与调头空间相切外，还需要在螺旋线运动过程中避免与调头空间外的任何部分发生碰撞。因此，需要在每一时刻对螺旋线的位置进行碰撞检测。通过计算螺旋线任意时刻t处的点$(x(t),y(t))$到调头空间中心的距离$D(t)$公式为：

$$D(t)=\sqrt{(x(t)-x_c)^2+(y(t)-y_c)^2} \tag{27}$$

如果在某一时刻，有$d(t)>4.5$（即大于调头空间的半径），则判定发生了碰撞。在此情况下，需要调整螺旋线的螺距p，改变螺旋线的形状，使其更加紧密，进而避免碰撞。通过迭代方法不断调整螺距，直到找到一个最小的螺距p，既保证螺旋线与调头空间边界相切，又避免与外部空间发生碰撞。

7.3 模型求解

基于该问题的模型建立假设，按照以下步骤进行模型求解：

1. 建立螺旋线方程，计算舞龙队的路径。
2. 判断螺旋线与调头空间的相切性，确保螺旋线在进入调头空间时满足距离和切线条件。
3. 模拟螺旋线运动，判断在调头过程中是否发生碰撞。
4. 通过迭代调整螺距，直至找到最小螺距$d_{\min}$，保证相切条件和无碰撞。
5. 输出最终螺距结果为 39 cm。

最终可以到达，在螺距为 39 cm 的条件下，碰撞发生时，距离原点的距离 4.3 m，小于调头空间的半径 4.5 m，故符合题干龙头能盘入调头空间的要求，结果如图 10 所示。

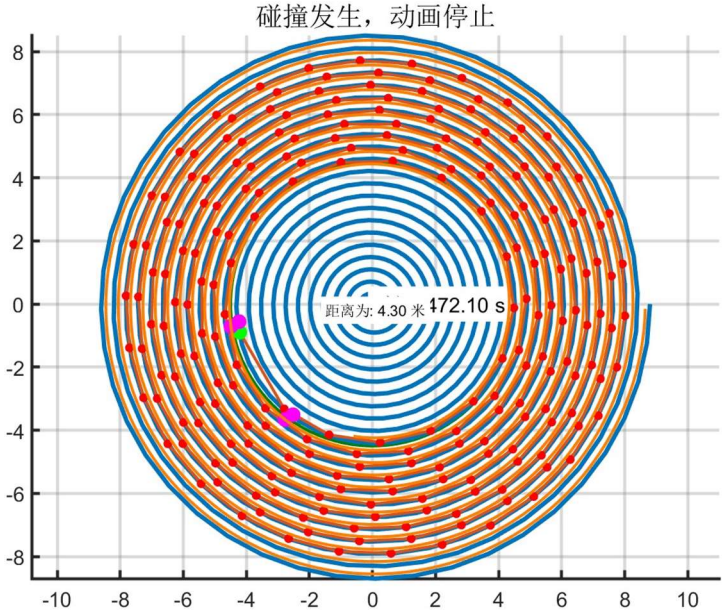

图 10 碰撞发生在小于 4.5m 区域的判定检测

八、 问题四模型的建立与求解

8.1 模型建立

8.1.1 几何模型与曲率连续性

在几何模型中，调头路径由两段相切的圆弧构成，设第一段圆弧的半径为r_A，第二段圆弧的半径为r_B，并且满足$r_B=2r_A$的关系。在舞龙队盘入螺线行进至调头点时，进入与螺线相切的第一段圆弧路径。螺线与第一段圆弧的相切点满足以下方程：

$$f'(x_1,y_1)=g'(x_1,y_1) \tag{28}$$

式中，$f(x_1,y_1)$为螺线的方程，$g(x_1,y_1)$为第一段圆弧的方程。通过求解该方程，可以确定相切点(x_1,y_1)的具体坐标，并由此推导出第一段圆弧的参数方程。

在第一段圆弧与第二段圆弧的连接处，要求两段圆弧的曲率在切点处保持连续性，以确保路径的光滑过渡。曲率连续条件可表示为：

$$\frac{1}{r_A}=\frac{1}{r_B} \tag{29}$$

该条件保证了两段圆弧的切线方向在切点处保持一致性，并避免路径的突变。通过该条件推导出第二段圆弧的方程，并确保调头完成时，第二段圆弧与盘出的螺线相切。最终，利用类似的相切条件，可以确定第二段圆弧与盘出螺线的最终切点坐标，从而完成调头路径的几何建模。

8.1.2 基于路径优化与最短调头路径的模型

在该迭代过程中，通过随机搜索来找到两个符合特定几何条件的切点A和B，并最小化一个几何目标函数。该过程可以描述为一个带有约束条件的优化问题，并通过迭代生成候选解逐步逼近最优解。

首先，设两个螺旋线分别为盘入螺线和盘出螺线，它们的方程由极坐标形式给出，螺距p确定每圈的间距。螺旋线上的点A和B的坐标通过随机生成极坐标角度θ_A和θ_B来确定，给定$r_A=p\cdot\theta_A$和 $r_B=p\cdot\theta_B$它们的笛卡尔坐标分别为：

$$\begin{cases} x_A=r_A\cdot\cos(\theta_A), \quad y_A=r_A\cdot\sin(\theta_A) \\ x_B=-r_B\cdot\cos(\theta_B), \quad y_B=-r_B\cdot\sin(\theta_B) \end{cases} \tag{30}$$

这两个点在每次迭代中根据随机角度生成，并且其位置需满足不发生碰撞和位于调头空间内的约束条件，其可表达为：

$$1\le\sqrt{x_A^2+y_A^2}\le 4.5, \quad 1\le\sqrt{x_B^2+y_B^2}\le 4.5 \tag{31}$$

此外，切点A和B间的距离必须大于 6 米，这一约束可以表达为：

$$\sqrt{(x_A-x_B)^2+(y_A-y_B)^2}\ge 6 \tag{32}$$

在每次迭代中，生成的点对如果不满足这些约束条件，则被立即舍弃，不再进行后续计算。一旦A和B点满足上述几何约束，接下来我们将引入中间点C，它位于A和B之间，具体坐标为：

$$x_C=\frac{2x_B+x_A}{3}, \quad y_C=\frac{2y_B+y_A}{3} \tag{33}$$

随后，构造两个中点D和E，分别为A与C以及B与C的中点：

$$\begin{cases} D_x=\dfrac{x_A+x_C}{2}, \quad D_y=\dfrac{y_A+y_C}{2} \\ E_x=\dfrac{x_B+x_C}{2}, \quad E_y=\dfrac{y_B+y_C}{2} \end{cases} \tag{34}$$

这些中点的坐标用于计算两个半径r_{DA}和r_{EB}，它们分别为到的距离和E到的距离：

$$\begin{cases} r_{DA}=\sqrt{(D_x-x_A)^2+(D_y-y_A)^2} \\ r_{ED}=\sqrt{(E_x-x_B)^2+(E_y-y_B)^2} \end{cases} \tag{35}$$

基于此，本问题需要优化的目标函数为：

$$\min\ l=\varphi\cdot r_b+(2\pi-\varphi)r_a \tag{36}$$

式中，l为题目要求的调头曲线长度，φ为调头曲线两段弧的圆心角，即∠ADC 和∠BEC，其大小相等。

迭代过程的核心目标是找到满足$r_{DA}=2\times r_{EB}$的点对，并在满足这一条件的同时最小化r_{da}。

在每次迭代中，首先根据随机角度生成A和B的坐标，计算它们是否满足几何约束条件。如果满足条件，则继续计算中间点C、中点D和E，并计算两个半径r_{DA}和r_{EB}。如果当前的r_{DA}值比之前迭代中的最小值更小，且满足$r_{DA}=2\times r_{EB}$的比例关系，则更新最优解。如图 11 所示，基于该问题所建立的模型可以得到圆弧上龙头与部分龙身调头轨迹。

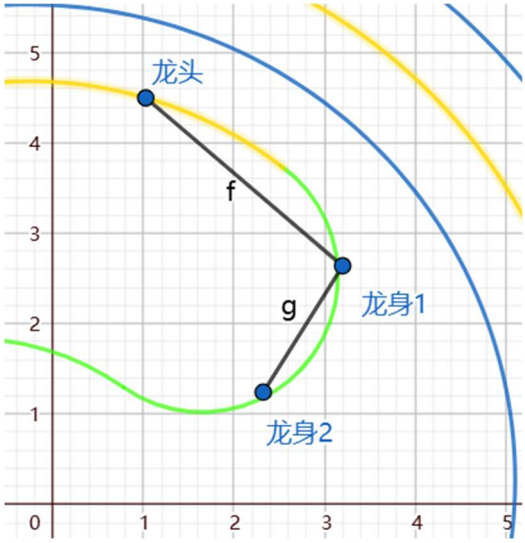

图 11 圆弧上龙头与部分龙身调头轨迹

8.2 模型求解

根据问题二建立的碰撞检测模型确保距离安全，在发生不碰撞的情况下，基于问题一建立的位置与速度的动态计算模型逐步迭代出相关数值。图 12 展示了问题四建模与求解的总体流程。

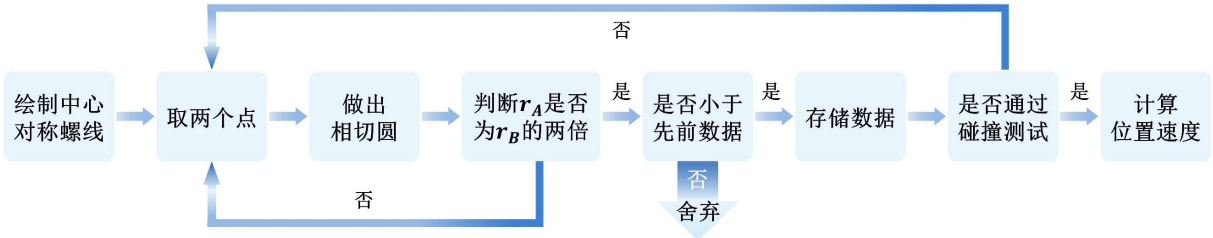

图 12 问题四建模的求解流程图

在满足个约束条件的前提下，通过上文所建立模型进行不断迭代后，得到的最短弧长如图 13 所示，最短弧长长度为 13.3 米。

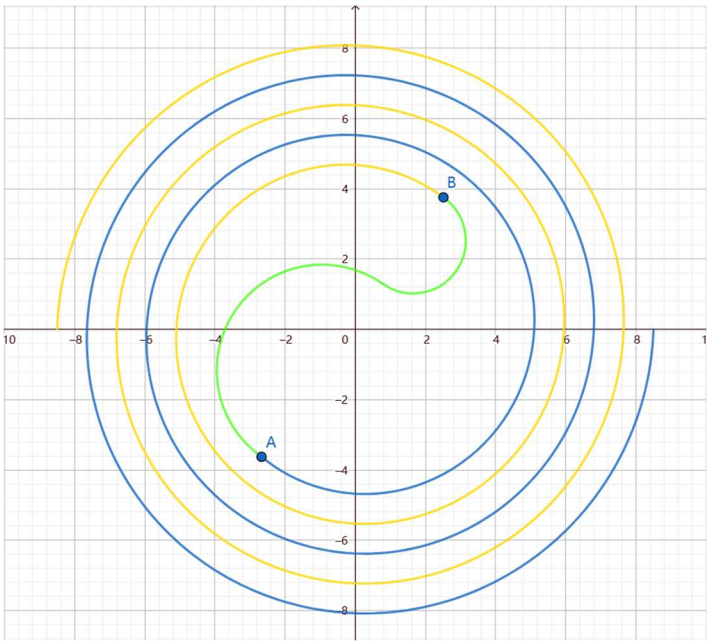

图 13 板凳龙完整盘入以及盘出的调头轨迹

根据该问题的模型求解，最终可以得到以调头开始前后 100 s 的舞龙队所求节点的位置与速度，如表 4 和表 5 所示。

表 4 问题 4 舞龙队位置结果

	-100 s	-50 s	0 s	50 s	100 s
龙头 x(m)	7.778033	6.608301	-2.711856	1.408772	-3.083475
龙头 y(m)	3.717164	1.898864	-3.591078	6.154956	7.576127
第 1 节龙身 x(m)	6.209273	7.459393	-0.063534	3.921315	-0.268126
第 1 节龙身 y(m)	6.108521	-3.629946	-4.670888	4.788674	8.079528
第 51 节龙身 x(m)	-10.608038	-8.963800	2.459962	-1.091642	2.162555
第 51 节龙身 y(m)	2.831491	2.575037	-7.778145	-6.356925	3.992649
第 101 节龙身 x(m)	-3.224889	10.125787	3.008493	-8.011888	-7.703658
第 101 节龙身 y(m)	-11.922761	-5.972247	10.108539	4.700838	0.915322
第 151 节龙身 x(m)	-14.351032	12.974783	-7.002789	-4.057692	9.947604
第 151 节龙身 y(m)	-1.980993	-3.810358	10.337482	-10.700207	-2.429277
第 201 节龙身 x(m)	-11.952942	10.522508	-6.872842	0.964946	7.704519
第 201 节龙身 y(m)	10.566998	-10.807426	12.382609	-13.217101	9.495343
龙尾（后）x(m)	-1.011059	0.189810	11.004596	5.313002	-10.390173
龙尾（后）y(m)	-16.527573	15.720588	0.492662	12.924603	-7.824092

表 5 问题 4 舞龙队速度结果

	-100 s	-50 s	0 s	50 s	100 s
龙头(m/s)	0.984886	0.984886	1.000000	0.948683	0.986701
第 1 节龙身(m/s)	0.948683	0.921954	1.000000	1.000000	0.989659
第 51 节龙身(m/s)	1.004988	1.004988	1.063015	0.412311	0.553376
第 101 节龙身(m/s)	0.989949	0.905539	1.063015	0.412311	0.457159
第 151 节龙身(m/s)	0.984886	1.000000	1.000000	0.412311	0.456475
第 201 节龙身(m/s)	0.984886	1.029563	1.063015	0.412311	0.452372
龙尾（后）(m/s)	0.995801	1.002715	0.998143	1.003117	0.999981

九、 问题五模型的建立与求解

9.1 模型建立

问题五要求在问题四的路径上，在保证舞龙队各把手速度不超过 2 m/s 的前提下确定出龙头前进的最大速度，故优化模型可写作：

$$\max\ v_h$$

$$s.t\begin{cases} v_i \leq 2 \\ 1 \leq \sqrt{x_A^2 + y_A^2} \leq 4.5, \\ 1 \leq \sqrt{x_B^2 + y_B^2} \leq 4.5 \\ \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \geq 6 \end{cases}$$

9.2 模型求解

为了确定龙头的最大行进速度$v_{\max}$，需要迭代地调整龙头速度，并实时更新舞龙队各节的速度。具体步骤如下：

（1） 初始条件：首先设定龙头速度v_h=1.5 m/s

（2） 速度检查：检查所有节的速度v_i是否都小于等于 2 m/s。

（3） 速度调整：如果存在某节速度v_i>2 m/s,则降低龙头速度，否则增加。

（4） 收敛条件：重复步骤和（3），直到找到使所有节的速度不超过 2 m/s 的最大龙头速度$v_{\max}$。

具体流程图如图 14 所示。

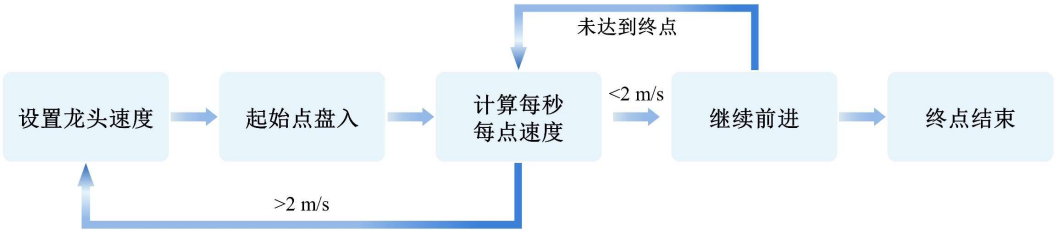

图 14 问题五建模与求解流程图

在问题五的场景下，为了确定龙头的最大行进速度$v_{\max}$，我们引入了一种基于速度约束的迭代优化算法，采用该模型方法确保舞龙队各节速度不超过 2 m/s。

通过如上步骤可以得到具体的龙头前孔完整运动轨迹情况如图 15 所示，最终解得龙头最大行进速度为 1.25 m/s。

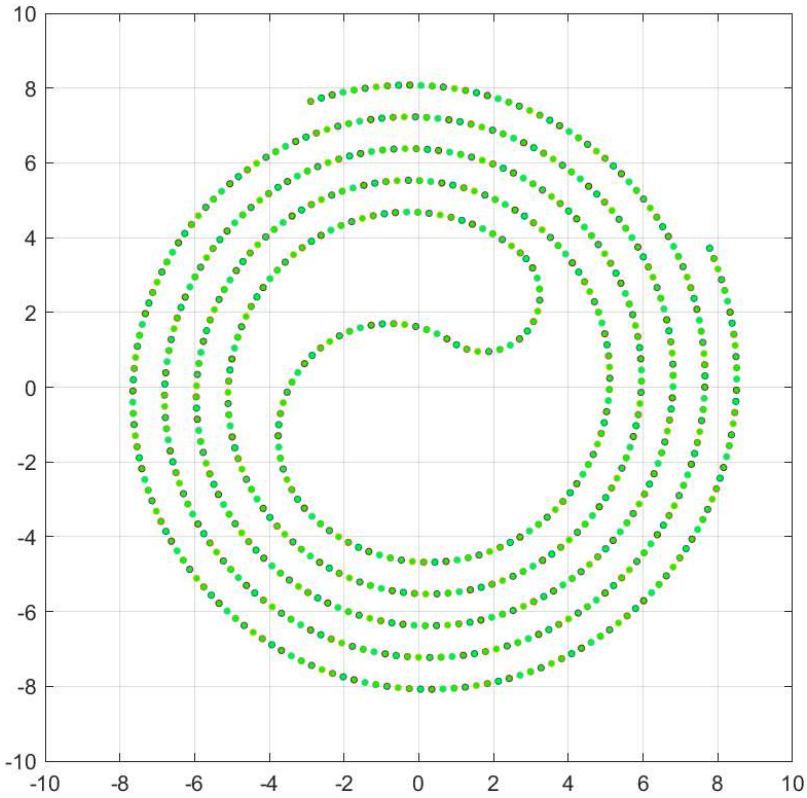

图 15 龙头前孔完整运动轨迹情况

十、 模型评价

10.1 模型的优点

1. 动态精确性： 通过极坐标和差分方程计算每节板凳的实时位置和速度，模型能够精确地模拟舞龙队在螺旋线运动中的动态行为。

2. 优化路径： 采用迭代算法和几何约束条件，确保螺旋线与调头路径的最小化，同时避免碰撞，路径设计具有较高的优化水平。

3. 多点碰撞检测： 多点碰撞检测确保舞龙队各节板凳不会发生碰撞，提高了模型的现实适用性和安全性。

10.2 模型的缺点

1. 忽略板凳制造误差和操作误差： 模型假设所有板凳尺寸一致、连接无误，未考虑实际中可能存在的制造偏差及人员在操作偏差，可能导致模型与实际情况不符。

2. 复杂计算难以实时应用： 模型在路径优化和碰撞检测过程中涉及多次迭代和几何约束求解，虽然提高了精度，但复杂的计算流程可能不适合需要实时反馈的场景。

3. 忽略动态变化： 模型假设舞龙队行进速度恒定，没有考虑物理层面的速度变化或加减速情况，可能影响对真实场景的模拟效果。

10.3 模型的改进与推广

为进一步提高模型的精确度和普适性，模型可以通过实验数据或文献研究来量化摩擦力和风力的影响，并在模型中引入相应的阻力项。同时在模型中引入随机变量，模拟制造误差对运动路径和稳定性的影响，进行敏感性分析。将匀速假设改为动态速度模型，考虑加速和减速的因素，使用微分方程描述运动状态的变化。在模型中引入板凳的厚度，分析其对运动路径和稳定性的影响，可能需要对运动轨迹进行三维建模。并引入队伍在

排队等待和演出过程中动态调整的模型，考虑队伍成员之间的相互影响和协调行为。

本文所建立数学模型的应用场景为板凳龙民俗演出，而等距螺线作为一种在工程、建筑、生物、计算机、艺术与设计中广泛应用的曲线，本问所建立的数学模型可以作为为上述领域的设计、建模和分析的参考依据，并推动科学研究的进步，并有望在实践中带来一定的经济和社会效益[3-5]。

十一、 参考文献

[1] 陈钰娉. 义乌板凳龙的传承与发展研究[D], 湖南师范大学, 2021.

[2] 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.

[3] 谭乐祖, 张诗, 盛文平, 杜军文. 直升机单机等距螺旋应召搜潜效能模型与仿真[J]. 火力与指挥控制, 2016, 41(08): 102-104.

[4] 高浩强. 含阿基米德螺旋线单相周期结构的低频减振特性研究[D], 天津大学,2022.

[5] 罗秀丽, 马俊卉, 陈东启, 蔡毅, 王岭雪, 黄攀, 王彩萍, 罗宏. 多波段热成像阿基米德螺旋线推扫滤光盘工程设计[J]. 红外技术, 2019, 41(09): 799-805.

十二、 附录

附录 1

介绍：支撑材料的文件列表

1. question1position.m
2. question1velocities.m
3. question2.m
4. question3.m
5. question4position.m
6. question4velocities.m
7. question5position.m
8. 问题四速度求解.ipynb
9. x.xlsx
10. y.xlsx
11. result1
12. result2
13. result4

附录 2

介绍：matlab 语言，用于计算问题 1 舞龙队每秒位置

```
clear; clc;

a = 224;
b = 2.86;
c = 1.65;
d = 0.55;
e = 0.088;
f = 8.80;
g = 0;
h = 1;
i = 301;
j = 1;

k = zeros(i, 1 + a * 2);

l = zeros(a, 2);

m = f;
n = g;
o = m * cos(n);
p = m * sin(n);
l(1, :) = [o, p];

for q = 2:a
```

```
    if q == 2
        r = b;
    else
        r = c;
    end

    s = atan2(l(q-1, 2), l(q-1, 1));
    t = sqrt(l(q-1, 1)^2 + l(q-1, 2)^2);

    u = r / t;
    v = s + u;
    w = t + e * u;

    x = w * cos(v);
    y = w * sin(v);
    l(q, :) = [x, y];
end

for z = 1:i
    k(z, 1) = z;

    for aa = 1:a
        k(z, 2*aa) = l(aa, 1);
        k(z, 2*aa+1) = l(aa, 2);

        ab = atan2(l(aa, 2), l(aa, 1));
        ac = sqrt(l(aa, 1)^2 + l(aa, 2)^2);
        ad = (h * j) / ac;

        ae = ab - ad;
        af = ac - e * ad;

        ag = af * cos(ae);
        ah = af * sin(ae);
        l(aa, :) = [ag, ah];
    end
end

ai = 'position_data_formatted.xlsx';
writematrix(k, ai);

disp(['Position data saved to ', ai, ' successfully.']);
```

附录 3

matlab 语言，用于计算问题 2 舞龙队不能盘入的时刻和此时的位置和速度

```
clear; clc;

a = 224;
b = 2.86;
c = 1.65;
d = 0.088;
e = 8.80;
f = 16 * 2 * pi;

g = zeros(a, 2);

h = e;
i = f;
j = h * cos(i);
k = h * sin(i);
g(1, :) = [j, k];

for l = 2:a
    if l == 2
        m = b;
    else
        m = c;
    end

    n = atan2(g(l-1, 2), g(l-1, 1));
    o = sqrt(g(l-1, 1)^2 + g(l-1, 2)^2);

    p = m / o;
    q = n + p;
    r = o + d * p;

    s = r * cos(q);
    t = r * sin(q);

    g(l, :) = [s, t];
end

u = 8.80;
v = 8.65;
w = 0.55;

figure('Position', [100, 100, 800, 600]);
```

```
x = gca;
x.FontSize = 14;
x.GridColor = [0.5 0.5 0.5];
x.GridAlpha = 0.3;
x.LineWidth = 2;
hold on;
grid on;
axis equal;

y = linspace(0, -u * 2 * pi / w, 1000);
z = u + w * y / (2 * pi);

aa = z .* cos(y);
bb = z .* sin(y);

plot(aa, bb, 'LineWidth', 3, 'Color', [0 0.45 0.74]);

cc = g(1, :);
dd = g(2, :);

ee = (cc + dd) / 2;

ff = (dd - cc);
ff = ff / norm(ff);

gg = [-ff(2), ff(1)];

hh = [-0.15, -1.705; 0.15, -1.705; 0.15, 1.705; -0.15, 1.705];

ii = zeros(4, 2);
for jj = 1:4
    ii(jj, :) = ee + hh(jj, 1) * gg + hh(jj, 2) * ff;
end

kk = plot(g(:, 1), g(:, 2), 'ro-', 'MarkerSize', 6, 'MarkerFaceColor', 'r', 'MarkerEdgeColor',
'none', 'LineWidth', 2, 'Color', [0.85 0.33 0.1]);
ll = plot(g(1, 1), g(1, 2), 'go', 'MarkerSize', 10, 'MarkerFaceColor', 'g', 'MarkerEdgeColor',
'none');
mm = plot(nan, nan, 'LineWidth', 3, 'Color', [1 0.5 0]);

nn = [];

oo = [j - 0.15, k - 0.15];
```

```
pp = 4500;
qq = 0.1;
rr = 1;

ss = text(0, 0, 'Time: 0 s', 'FontSize', 14, 'Color', 'k', 'BackgroundColor', 'w');

tt = plot(ii(:, 1), ii(:, 2), 'mo', 'MarkerSize', 10, 'MarkerFaceColor', 'm', 'MarkerEdgeColor',
'none');

for uu = 1:pp
    for vv = 1:a
        ww = atan2(g(vv, 2), g(vv, 1));
        xx = sqrt(g(vv, 1)^2 + g(vv, 2)^2);

        yy = (rr * qq) / xx;
        zz = ww - yy;
        aaa = xx - d * yy;

        bbb = aaa * cos(zz);
        ccc = aaa * sin(zz);

        g(vv, :) = [bbb, ccc];
    end

    ee = (g(1, :) + g(2, :)) / 2;
    ff = (g(2, :) - g(1, :));
    ff = ff / norm(ff);
    gg = [-ff(2), ff(1)];

    for jj = 1:4
        ii(jj, :) = ee + hh(jj, 1) * gg + hh(jj, 2) * ff;
    end

    set(kk, 'XData', g(:, 1), 'YData', g(:, 2));
    set(ll, 'XData', g(1, 1), 'YData', g(1, 2));
    set(tt, 'XData', ii(:, 1), 'YData', ii(:, 2));

    set(ss, 'String', sprintf('Time: %.2f s', uu * qq));

    if uu * qq >= 16
        ddd = atan2(oo(2), oo(1));
        eee = sqrt(oo(1)^2 + oo(2)^2);
        fff = (rr * qq) / eee;
        ggg = ddd - fff;
```

```
            hhh = eee - d * fff;

            iii = hhh * cos(ggg);
            jjj = hhh * sin(ggg);

            oo = [iii, jjj];
            nn = [nn; oo];

            kkk = linspace(ddd, ggg, 100);
            lll = eee + w * (kkk - ddd) / (2 * pi);
            mmm = lll .* cos(kkk);
            nnn = lll .* sin(kkk);

            plot(mmm, nnn, 'LineWidth', 2, 'Color', [1 0.5 0]);
        end

        if ~isempty(nn)
            ooo = inpolygon(nn(:, 1), nn(:, 2), ii(:, 1), ii(:, 2));

            if any(ooo)
                title('Collision detected, animation stopped', 'FontSize', 18);
                hold off;
                return;
            end
        end

        pause(qq);
end

title('Holes along the spiral path (clockwise)', 'FontSize', 18);
xlabel('X (meters)');
ylabel('Y (meters)');

hold off;
```

附录 4

matlab 语言，用于计算问题 3 舞龙队盘入调头边界时最小螺距

```
clear; clc;

a = 224;
b = 2.86;
c = 1.65;
d = 0.088;
e = 8.80;
```

```
f = 16 * 2 * pi;

g = zeros(a, 2);

h = e;
i = f;
j = h * cos(i);
k = h * sin(i);
g(1, :) = [j, k];

for l = 2:a
    if l == 2
        m = b;
    else
        m = c;
    end

    n = atan2(g(l-1, 2), g(l-1, 1));
    o = sqrt(g(l-1, 1)^2 + g(l-1, 2)^2);

    p = m / o;
    q = n + p;
    r = o + d * p;

    s = r * cos(q);
    t = r * sin(q);

    g(l, :) = [s, t];
end

u = 8.80;
v = 8.65;
w = 0.39;

figure('Position', [100, 100, 800, 600]);
x = gca;
x.FontSize = 14;
x.GridColor = [0.5 0.5 0.5];
x.GridAlpha = 0.3;
x.LineWidth = 2;
hold on;
grid on;
axis equal;
```

```
y = linspace(0, 2 * pi, 100);
z = 4.5 * cos(y);
aa = 4.5 * sin(y);
plot(z, aa, 'LineWidth', 2, 'Color', [0 0.5 0]);

bb = linspace(0, -u * 2 * pi / w, 1000);
cc = u + w * bb / (2 * pi);
dd = cc .* cos(bb);
ee = cc .* sin(bb);
plot(dd, ee, 'LineWidth', 3, 'Color', [0 0.45 0.74]);

ff = g(1, :);
gg = g(2, :);

hh = (ff + gg) / 2;

ii = (gg - ff);
ii = ii / norm(ii);

jj = [-ii(2), ii(1)];

kk = [-0.15, -1.705; 0.15, -1.705; 0.15, 1.705; -0.15, 1.705];

ll = zeros(4, 2);
for mm = 1:4
    ll(mm, :) = hh + kk(mm, 1) * jj + kk(mm, 2) * ii;
end

nn = plot(g(:, 1), g(:, 2), 'ro-', 'MarkerSize', 6, 'MarkerFaceColor', 'r', 'MarkerEdgeColor', 'none', 'LineWidth', 2, 'Color', [0.85 0.33 0.1]);
oo = plot(g(1, 1), g(1, 2), 'go', 'MarkerSize', 10, 'MarkerFaceColor', 'g', 'MarkerEdgeColor', 'none');
pp = plot(nan, nan, 'LineWidth', 3, 'Color', [1 0.5 0]);

qq = [];

rr = [j - 0.15, k - 0.15];

ss = 4500;
tt = 0.1;
uu = 1;

vv = text(0, 0, 'Time: 0 s', 'FontSize', 14, 'Color', 'k', 'BackgroundColor', 'w');
```

```
ww = plot(ll(:, 1), ll(:, 2), 'mo', 'MarkerSize', 10, 'MarkerFaceColor', 'm', 'MarkerEdgeColor',
'none');

for xx = 1:ss
    for yy = 1:a
        zz = atan2(g(yy, 2), g(yy, 1));
        aaa = sqrt(g(yy, 1)^2 + g(yy, 2)^2);

        bbb = (uu * tt) / aaa;
        ccc = zz - bbb;
        ddd = aaa - d * bbb;

        eee = ddd * cos(ccc);
        fff = ddd * sin(ccc);

        g(yy, :) = [eee, fff];
    end

    hh = (g(1, :) + g(2, :)) / 2;
    ii = (g(2, :) - g(1, :));
    ii = ii / norm(ii);
    jj = [-ii(2), ii(1)];

    for mm = 1:4
        ll(mm, :) = hh + kk(mm, 1) * jj + kk(mm, 2) * ii;
    end

    set(nn, 'XData', g(:, 1), 'YData', g(:, 2));
    set(oo, 'XData', g(1, 1), 'YData', g(1, 2));
    set(ww, 'XData', ll(:, 1), 'YData', ll(:, 2));

    set(vv, 'String', sprintf('Time: %.2f s', xx * tt));

    if xx * tt >= 16
        ggg = atan2(rr(2), rr(1));
        hhh = sqrt(rr(1)^2 + rr(2)^2);
        iii = (uu * tt) / hhh;
        jjj = ggg - iii;
        kkk = hhh - d * iii;

        lll = kkk * cos(jjj);
        mmm = kkk * sin(jjj);

        rr = [lll, mmm];
```

```
            qq = [qq; rr];

            nnn = linspace(ggg, jjj, 100);
            ooo = hhh + w * (nnn - ggg) / (2 * pi);
            ppp = ooo .* cos(nnn);
            qqq = ooo .* sin(nnn);
            plot(ppp, qqq, 'LineWidth', 2, 'Color', [1 0.5 0]);
        end

        if ~isempty(qq)
            rrr = inpolygon(qq(:, 1), qq(:, 2), ll(:, 1), ll(:, 2));

            if any(rrr)
                sss = g(1, 1);
                ttt = g(1, 2);
                uuu = sqrt(sss^2 + ttt^2);

                if uuu < 4.5
                    vvv = sprintf('Distance: %.2f meters', uuu);
                else
                    vvv = sprintf('Distance: %.2f meters', uuu);
                end

                text(mean(g(:,1)), mean(g(:,2)), vvv, 'FontSize', 10, 'Color', 'k',
'BackgroundColor', 'w', 'HorizontalAlignment', 'center');

                title('Collision detected, animation stopped', 'FontSize', 18);
                hold off;
                return;
            end
        end

        pause(tt);
end

title('Holes along the spiral path (clockwise)', 'FontSize', 18);
xlabel('X coordinate (meters)');
ylabel('Y coordinate (meters)');

hold off;
```

附录 5

matlab 语言，用于问题 4 舞龙队速度计算

```
a = readtable('y.xlsx');
b = a(:, [1, 11:10:width(a)]);

writetable(b, 'result_output2.xlsx');
disp('处理完毕，已保存至 result_output2.xlsx');



c = readtable('result_output2.xlsx');
c.Properties.VariableNames = arrayfun(@num2str, 1:width(c), 'UniformOutput', false);

writetable(c, 'header_output2.xlsx');
disp('表头插入成功，结果保存至 header_output2.xlsx');

file_processing('x.xlsx', 'processed_output1.xlsx');
file_processing('y.xlsx', 'processed_output2.xlsx');

final_processing('processed_output1.xlsx', 'processed_output2.xlsx',
'final_result_output.xlsx');

d = readtable('final_result_output.xlsx');
e = d(:, [1, 11:10:width(d)]);

writetable(e, 'final_data_output.xlsx');
disp('数据提取完成并保存至 final_data_output.xlsx');



function file_processing(a, b)
    c = readtable(a);
    c = round(c, 2);

    d = array2table((c{:, 1:end-1} - c{:, 2:end}).^2);

    for e = 1:size(d, 2)
        d.Properties.VariableNames{e} = ['DiffSquare_' num2str(e)];
    end

    writetable(d, b);
    disp(['处理结果已保存至 ', b]);
end

function final_processing(a, b, c)
    d = readtable(a);
```

```
    e = readtable(b);
    d(1,:) = [];
    e(1,:) = [];

    if ~isequal(size(d), size(e))
        error('文件维度不一致');
    end

    f = sqrt(d{:, :} + e{:, :}) / 0.1;

    writetable(array2table(f), c);
    disp(['最终结果已保存至 ', c]);
end
```

附录 6

matlab 语言，用于问题 5 舞龙队位置计算

```
clear,clc
warning off
a=1.7;
b=a/2/pi;
c=341e-2;
d=c-27.5e-2*2;
e=220e-2;
f=e-27.5e-2*2;
g=1;
h=223;
i=5*2*pi:-0.01:0*pi;
j=b*i;
k=j.*cos(i);
l=j.*sin(i);
figure(1)
set(gcf,'Position',[200 200 600 600]);
m=rand(1,3);
plot(k,l,'-','Color',m,'LineWidth',1.3)
axis equal
grid on
xlabel('x')
ylabel('y')
set(gca,'FontSize',18)
hold on
i=i-pi;
n=b*(i+pi);
o=n.*cos(i);
```

```
p=n.*sin(i);
q=rand(1,3);
plot(o,p,'m','Color',q,'LineWidth',1.3)
r=4.5;
s=r*cos(i);
t=r*sin(i);
u=sort(rand(1,3));
plot(s,t,'Color',u,'LineWidth',2)
v=r/b;
w=r/b-pi;
x=(b*sin(v)+r*cos(v))/(b*cos(v)-r*sin(v));
y=atan(-1/x)+pi;
z=atan(tan(v))+pi-y;
aa=r/cos(z);
ab=aa/3;
ac=ab*2;
ad=2*z;
ae=ac*(pi-ad);
af=ab*(pi-ad);
ag=y-ae/ac;
ah=ag-pi;
ai=ah+af/ab;
aj=r*cos(v)+ac*cos(y-pi);
ak=r*sin(v)+ac*sin(y-pi);
al=r*cos(w)-ab*cos(ai);
am=r*sin(w)-ab*sin(ai);
figure(1)
hold on
plot(aj+ac*cos(linspace(ag,y,50)),ak+ac*sin(linspace(ag,y,50)),'r','LineWidth',2)
plot(aj,ak,'r*')
plot(al+ab*cos(linspace(ah,ai,50)),am+ab*sin(linspace(ah,ai,50)),'b','LineWidth',2)
plot(al,am,'b*')
axis equal
an=@(ao,ap)1./(b*sqrt(1+ao.^2));
aq=v;
ar=0.1;
as_=0:ar:100;
[at,au]=ode45(an,as_,aq);
av=b*au.*cos(au);
aw=b*au.*sin(au);
ax=zeros(224,200/ar+1);
ay=zeros(224,200/ar+1);
az=zeros(224,200/ar+1);
ax(1,1:length(av))=av(end:-1:1);
```

```
ay(1,1:length(aw))=aw(end:-1:1);
az(1,1:length(au))=au(end:-1:1);
ba=ar:ar:ae;
bb=-ba/ac+y;
az(1,length(au)+(1:length(ba)))=bb;
ax(1,length(av)+(1:length(ba)))=ac*cos(bb)+aj;
ay(1,length(aw)+(1:length(ba)))=ac*sin(bb)+ak;
bc=ba(end)+ar:ar:ae+af;
bd=(bc-ae)/ab+ag-pi;
az(1,length(au)+length(ba)+(1:length(bc)))=bd;
ax(1,length(av)+length(ba)+(1:length(bc)))=ab*cos(bd)+al;
ay(1,length(aw)+length(ba)+(1:length(bc)))=ab*sin(bd)+am;
be=@(bf,bg)1./(b*sqrt(1+(bf+pi).^2));
bh=w;
bi=bc(end)+ar:ar:100;
[bj,bk]=ode45(be,bi,bh);
bl=b*(bk+pi).*cos(bk);
bm=b*(bk+pi).*sin(bk);
az(1,length(au)+length(ba)+length(bc)+1:end)=bk;
ax(1,length(av)+length(ba)+length(bc)+1:end)=bl;
ay(1,length(aw)+length(ba)+length(bc)+1:end)=bm;
figure(3)
set(gcf,'Position',[300 300 600 600])
clf
for bn=1:3:length(az(1,:))
    title({['t=',num2str((bn-1)*ar)],'头部把手中心的轨迹(-100 到 100s)'})
    plot(ax(1,bn),ay(1,bn),'Marker','o','MarkerSize',3,'MarkerFaceColor','g')
    hold on
    axis equal
    axis([-10 10 -10 10])
    grid on
    drawnow
end
bo=waitbar(0,'计算开始...');
bp=-100:ar:100;
for bq=1:length(bp)
    if bp(bq)<=0
        for br=2:h+1
            bs=d*(br<=2)+f*(br>2);
            bt=solve_theta1(a,ax(br-1,bq),ay(br-1,bq),az(br-1,bq),bs);
            az(br,bq)=bt;
            ax(br,bq)=b*bt*cos(bt);
            ay(br,bq)=b*bt*sin(bt);
        end
```

```
    elseif bp(bq)>0 && bp(bq)<=ae
        bu=2;
        for br=2:h+1
            bs=d*(br<=2)+f*(br>2);
            if bu==2
                [bv,bw,bx,bu]=solve_point_2_1(a,ax(br-1,bq),ay(br-1,bq),az(br-
1,bq),bs,ac,aj,ak,y);
                az(br,bq)=bx;
                ax(br,bq)=bv;
                ay(br,bq)=bw;
            else
                bt=solve_theta1(a,ax(br-1,bq),ay(br-1,bq),az(br-1,bq),bs);
                az(br,bq)=bt;
                ax(br,bq)=b*bt*cos(bt);
                ay(br,bq)=b*bt*sin(bt);
            end
        end
    elseif bp(bq)>ae && bp(bq)<=ae+af
        bu=3;
        for br=2:h+1
            bs=d*(br<=2)+f*(br>2);
            if bu==3
                [bv,bw,bx,bu]=solve_point_3_2(ax(br-1,bq),ay(br-1,bq),az(br-
1,bq),bs,ac,aj,ak,ab,al,am,ah);
                az(br,bq)=bx;
                ax(br,bq)=bv;
                ay(br,bq)=bw;
            elseif bu==2
                [bv,bw,bx,bu]=solve_point_2_1(a,ax(br-1,bq),ay(br-1,bq),az(br-
1,bq),bs,ac,aj,ak,y);
                az(br,bq)=bx;
                ax(br,bq)=bv;
                ay(br,bq)=bw;
            else
                bt=solve_theta1(a,ax(br-1,bq),ay(br-1,bq),az(br-1,bq),bs);
                az(br,bq)=bt;
                ax(br,bq)=b*bt*cos(bt);
                ay(br,bq)=b*bt*sin(bt);
            end
        end
    else
        bu=4;
        for br=2:h+1
            bs=d*(br<=2)+f*(br>2);
```

```
                if bu==4
                    [bv,bw,bx,bu]=solve_point_4_3(a,ax(br-1,bq),ay(br-1,bq),az(br-
1,bq),bs,ab,al,am,ai);
                    az(br,bq)=bx;
                    ax(br,bq)=bv;
                    ay(br,bq)=bw;
                elseif bu==3
                    [bv,bw,bx,bu]=solve_point_3_2(ax(br-1,bq),ay(br-1,bq),az(br-
1,bq),bs,ac,aj,ak,ab,al,am,ah);
                    az(br,bq)=bx;
                    ax(br,bq)=bv;
                    ay(br,bq)=bw;
                elseif bu==2
                    [bv,bw,bx,bu]=solve_point_2_1(a,ax(br-1,bq),ay(br-1,bq),az(br-
1,bq),bs,ac,aj,ak,y);
                    az(br,bq)=bx;
                    ax(br,bq)=bv;
                    ay(br,bq)=bw;
                else
                    bt=solve_theta1(a,ax(br-1,bq),ay(br-1,bq),az(br-1,bq),bs);
                    az(br,bq)=bt;
                    ax(br,bq)=b*bt*cos(bt);
                    ay(br,bq)=b*bt*sin(bt);
                end
            end
        end
        waitbar(bq/length(bp),bo,'已经完成...')
end
figure(100)
clf;
set(gcf,'Position',[200 200 600 600])
for bq=1:size(ax,2)
        plot(ax(:,bq),ay(:,bq),'k-
','LineWidth',1.2,'Marker','o','MarkerSize',6,'MarkerFaceColor','r')
        title({['t=',num2str(ar*(bq-1)-100)],'盘入-调头-盘出的轨迹'})
        axis equal
        grid on
        xlabel('x')
        ylabel('y')
        axis([-15 15 -15 15])
        drawnow
end
function by=solve_theta1(bz,ca,cb,cc,cd)
ce=bz/2/pi;
```

```
cf=@(cg)(ce*cg.*cos(cg)-ca).^2+(ce*cg.*sin(cg)-cb).^2-cd^2;
ch=0.01;
ci=optimoptions('fsolve','Display','off');
by=fsolve(cf,cc+ch,ci);
while by<=cc || abs(ce*by-ce*cc)>bz/2
    ch=ch+0.1;
    by=fsolve(cf,by+ch,ci);
end
end
function by=solve_theta2(bz,ca,cb,cc,cd)
ce=bz/2/pi;
cf=@(cg)(ce*(cg+pi).*cos(cg)-ca).^2+(ce*(cg+pi).*sin(cg)-cb).^2-cd^2;
ch=-0.1;
ci=optimoptions('fsolve','Display','off');
by=fsolve(cf,cc+ch,ci);
while by>=cc || abs(ce*by-ce*cc)>bz/2
    ch=ch-0.1;
    by=fsolve(cf,by+ch,ci);
end
end
function [cj,ck,cl,cm]=solve_point_2_1(cn,co,cp,cq,cr,cs,ct,cu,cv)
cw=cn/2/pi;
cx=2*asin(cr/2/cs);
if cx<=cv-cq
    cm=2;
    cl=cq+cx;
    cj=ct+cs*cos(cl);
    ck=cu+cs*sin(cl);
else
    cl=solve_theta1(cn,co,cp,4.5/cw,cr);
    cm=1;
    cj=cw*cl*cos(cl);
    ck=cw*cl*sin(cl);
end
end
function [cj,ck,cl,cm]=solve_point_3_2(co,cp,cq,cr,cs,ct,cu,cv,cw,cx,cy)
cz=2*asin(cr/2/cv);
if cz<=cq-cy
    cm=3;
    cl=cq-cz;
    cj=cw+cv*cos(cl);
    ck=cx+cv*sin(cl);
else
    da=sqrt((co-ct)^2+(cp-cu)^2);
```

```
        db=acos((da^2+cs^2-cr^2)/2/da/cs);
        dc=atan((cp-cu)/(co-ct));
        cl=dc+db;
        cm=2;
        cj=ct+cs*cos(cl);
        ck=cu+cs*sin(cl);
end
end
function [cj,ck,cl,cm]=solve_point_4_3(cn,co,cp,cq,cr,cs,ct,cu,cv)
cw=cn/2/pi;
cl=solve_theta2(cn,co,cp,cq,cr);
if cl>=4.5/cw-pi
        cm=4;
        cj=cw*(cl+pi)*cos(cl);
        ck=cw*(cl+pi)*sin(cl);
else
        cf=@(cd)(ct+cs*cos(cv-cd)-co).^2+(cu+cs*sin(cv-cd)-cp).^2-cr^2;
        ch=0.1;
        ci=optimoptions('fsolve','Display','off');
        db=fsolve(cf,cq+ch,ci);
        cl=cv-db;
        cm=3;
        cj=ct+cs*cos(cl);
        ck=cu+cs*sin(cl);
end
end
```