基于单目标优化的板凳龙二维路径规划与碰撞检测

摘　要

“板凳龙”是浙闽地区重要民俗之一，民众将上百条板凳首尾相连，进行独特的盘龙表演。本文通过建立板凳龙运动模型，综合运用多种优化算法，解决了碰撞检测、路径优化、速度控制等问题，为安全地提高表演观赏性提供策略。

针对问题一：建立模型模拟舞龙队的运动过程。首先，在极坐标系中建立螺线方程，使用微分方程描述运动，使用**梯形公式**近似求解运动时间关于极角的积分，进而求解任意时刻龙头的极角。接着，根据板凳连接的几何关系建立把手中心位置**递推**关系，将**连续时间离散化**处理，得到每秒舞龙队各把手中心角度，最后通过极坐标系和笛卡尔坐标系的转换关系得到位置坐标，并利用位置对时间的导数求得每个时间点每个把手的速度，在 300 秒的过程中，板凳龙完成约 7 圈盘入，把手速度呈现**逐渐下降**的趋势，且距离龙头越远的把手速度越小。每秒的位置与速度结果见文件 result1.xlsx。

针对问题二：建立**碰撞检测**模型，并与问题一的模拟运动模型结合。在每轮运动模拟中，选取龙头与其外圈龙身作为检测对象，利用位置坐标和几何关系计算相应顶点，再通过**点在矩形内**的方法判断碰撞发生。同样对时间进行离散化处理，利用迭代算法解得终止时刻为**第 413 秒**，此时舞龙队的位置和速度结果存在于文件 result2.xlsx 中。

针对问题三：建立了基于自适应迭代算法求解的**单目标优化模型**。基于问题一的运动模型，建立约束条件要求龙头把手中心与极点的距离小于调头空间半径；基于问题二的碰撞检测模型，要求龙头在进入调头空间前始终不发生碰撞。以螺距最小为优化目标，设置惩罚函数，在给定区间内进行**自适应寻优**，根据当前距离动态修改迭代步长，最终解得能使龙头前把手按要求盘入调头空间边界的**最小螺距为 0.445m**。

针对问题四：将运动路径的复杂几何关系转化为约束条件，建立以调头曲线最短为目标的**单目标优化模型**。首先确定盘入螺线、盘出螺线及调头曲线的方程，再用交点坐标与切线斜率刻画相交与相切关系，引入调头空间范围限制，同时防止运动过程中出现**碰撞或回退**。采用**遗传算法**求解该优化问题，得到更短的调头曲线，该曲线由半径分别为 **3.014m** 和 **1.507m** 的两个圆弧组成，**最短的调头曲线为 13.6148m**。最后，计算调头前后 100 秒范围内每秒舞龙队的位置和速度，该结果存放于文件 result4.xlxs 中。

针对问题五：首先采用与问题一类似的方法，分别确定螺线运动和圆周运动时龙身各把手速度与龙头的关系。接着以各把手速度限制作为约束条件，建立以龙头速度为优化目标的**单目标优化模型**，采用**黄金分割迭代优化算法**快速寻找最优解，得到**最大龙头速度为 1.24m/s**。

关键词：**优化模型　微分方程　自适应寻优　遗传算法　黄金分割迭代**

一、 问题重述

浙闽地区有一种传统民俗活动被称为“板凳龙”，其表演形式为将板凳的首尾相连，形成龙头、龙身、龙尾三部分，并随龙头盘旋移动，呈现圆盘状。通常，在舞龙队能够顺利盘入和盘出的前提下，盘龙时速度越快、所占面积越小，视觉效果越好。

已知某板凳龙由 223 节板宽 $30cm$ 的板凳组成，包括 1 节板长 $341cm$ 的龙头，221 节板长 $220cm$ 的龙身和 1 节板长 $220cm$ 的龙尾。每节板凳均有 2 个直径 $5.5cm$ 的孔，用于通过把手将板凳相互连接，孔中心距离板头近端 $27.5cm$。建立模型解决以下问题:

问题一： 板凳龙盘入过程中，舞龙队沿着螺距为 $55cm$ 的等距螺线行进，龙头前把手以恒定的 $1m/s$ 速度顺时针移动，初始位置位于第 16 圈的螺线上。在 300 秒时间范围内，求解每秒舞龙队各把手中心的位置和速度。

问题二： 在问题一的基础上，确定舞龙队在盘入过程中何时不能再继续盘入，并计算此时舞龙队的位置和速度。

问题三： 为了从顺时针盘入转换为逆时针盘出，舞龙队需要在螺线中心周围留出调头空间。该空间是以螺线中心为圆心，直径为 $9m$ 的圆形区域。求解龙头前把手能够沿相应路线盘入到调头空间边界时的最小螺距。

问题四： 假设盘入螺线的螺距为 $1.7m$，盘出螺线与盘入螺线关于螺线中心呈中心对称。舞龙队在问题三设定的空间内完成调头，调头路径为 S 形曲线，由两段相切的圆弧构成，前一段圆弧的半径为后一段的 2 倍，且该路径与盘入和盘出螺线均相切。探讨调整圆弧以缩短路径的可行性，并求解调头前后 100 秒范围内每秒舞龙队位置和速度。

问题五： 在问题四的路径设定下，假设龙头保持恒定行进速度，要求所有把手速度均不超过 $2m/s$，试确定此条件下龙头最大行进速度。

二、 问题分析

2.1 问题一的分析

问题一要求计算舞龙队沿螺线盘入时的位置和速度。在求解位置时，由于各板凳相互连接并跟随龙头运动，可以优先考虑龙头的位置。龙头把手中心在已知螺线上以恒定速度运动，因此，龙头的运动路程等于该点在螺线上运动的弧长，从而构建时间和角度之间的关系，进而求解每个时刻龙头的位置。

随后考虑板凳间的连接关系。图1显示了各板凳的连接关系及相关参数。将各板凳视为刚体，即内部各点的相对位置不变，不发生弹性形变和振动。因此板凳前后把手中心间距离恒定，可以通过几何关系逐个计算出各把手位置。在确定位置后，通过位置对时间的导数能够得到速度。在求解时，可以对时间进行离散化处理，并用梯形公式对积分表达式进行近似计算。而后采用二分迭代法，求解每个时间节点的近似速度。

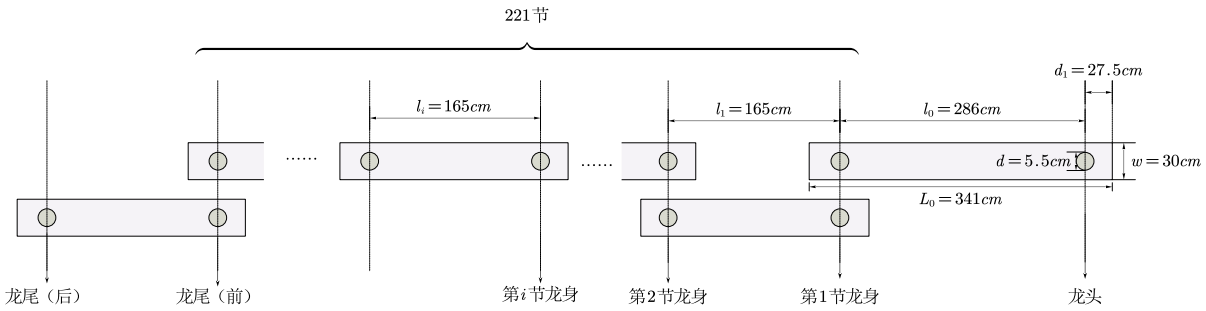


图 1　板凳龙结构示意图

2.2 问题二的分析

问题二要求确定运动的终止时刻，是一个碰撞检测问题。基于问题一建立的运动模型，能够计算运动过程中各板凳把手中心位置，进而根据几何关系计算出板凳的顶点位置，用于判断碰撞情况。

由于碰撞一定发生在龙头与某节龙身之间，考虑龙头与周围一圈的龙身之间可能发生的碰撞，即与其转动角度差值不超过 2π 的龙身之间的碰撞。如果龙头某顶点运动到龙身的矩形内部，则表明发生了碰撞，此时该点与矩形四边连线所形成的夹角和为 2π。可以与第一问采用同样的离散化和迭代的方式进行计算，得到发生碰撞的时刻，并求解此时舞龙队的位置和速度。

2.3 问题三的分析

问题三要求确定能使龙头前把手顺利盘入调头空间时的最小螺距。该问可被视为一个优化问题，以能够进入调头空间且此前不发生碰撞为约束条件，螺距最小为优化目标，建立单目标优化模型。

在一维搜索中，在每次试验中，考虑使用自适应步长，当龙头把手中心与原点距离接近 0.45 时，缩小时间步长，以更加高效和精确地靠近边界条件；此外，采用基于问题二的模型检测碰撞情况，确保得到的最小螺距满足约束条件。

2.4 问题四的分析

问题四要调整调头路径中的圆弧使得调头曲线变短，同样可视为一个单目标优化问题。这一优化问题有多个约束条件，包括盘入螺线与盘出螺线关于螺线中心呈中心对称，调头路径具有特定的形状和空间位置关系，舞龙队需要在给定调头空间完成调头，运动过程中不发生碰撞。根据这些约束条件可以建立以调头曲线长度为目标函数的优化模型。

该模型中存在多个决策变量，可以使用启发式搜索算法，如遗传算法，较快地求解这一复杂的优化问题。

2.5 问题五的分析

问题五在问题四的基础上研究龙头的最大行进速度。由于把手速度存在 $2m/s$ 的限制，需要得到把手速度的计算方式。舞龙队的运动包括螺线运动和圆周运动两个过程，其中螺线运动时龙身与龙头的速度关系可由问题一的模型给出。考虑圆周运动时，同样用板凳间连接关系描述各节点的位置关系，建立位置的递推求解模型，进而求解速度。对于每一个龙头速度，能够求解全过程中其余把手的速度，于是该问题被转化为在约束条件下寻找龙头速度最大值的问题，考虑使用黄金分割迭代算法来快速寻找最优解。黄金分割迭代优化算法是一种常用的优化方法，可以利用黄金分割法缩小螺距的搜索范围，更高效地确定最小螺距。

三、 符号说明

符号	含义	量纲
θ	极角	弧度
r	点到螺线中心的距离	m
b	螺线变化率	$-$
l_i	第 i 节龙身两孔中心之间的距离	m
v	把手中心的移动速度	m/s
t	运动时间	s
p	螺距长度	m
R	调头曲线中第二个圆弧的半径	m
γ	调头曲线中圆弧半径与 x 轴夹角	弧度
(x_i, y_i)	第 i 个把手的坐标	$-$
Δt	时间迭代步长	s
Δp	螺距迭代步长	m
Δv	速度迭代步长	m/s

四、 模型假设

1. 假设板凳龙在运动开始前，盘入和盘出阶段，各把手中心始终位于螺线上，且完全按照预定路线行进。
2. 假设各板凳为刚体，即内部各点的相对位置不变，同时忽略板凳运动时产生的弹性形变和振动。
3. 只考虑板凳龙在二维平面上的运动规律和现象，忽略竖直方向上的运动和倾斜，忽略相邻重合板凳间的碰撞。

五、 模型的建立与求解

5.1 问题一模型建立与求解

5.1.1 问题一模型建立

建立螺线方程 问题一需要求解盘入过程中舞龙队的位置和速度。由于舞龙队沿等距螺线行进，所以根据等距螺线定义，在行进过程中，螺线上的每个点到中心的距离，相对于转过的角度的变化率为一个定值，在极坐标系中有：

$$dr/d\theta \equiv C(C\text{为一定值}) \tag{1}$$

其中，令 r 表示螺线上一点到螺线中心的距离，θ 表示转过的角度，b 为表示螺线变化率的常数, 即 $b = C = dr/d\theta$。以螺线中心为原点，建立如2所示的极坐标系。

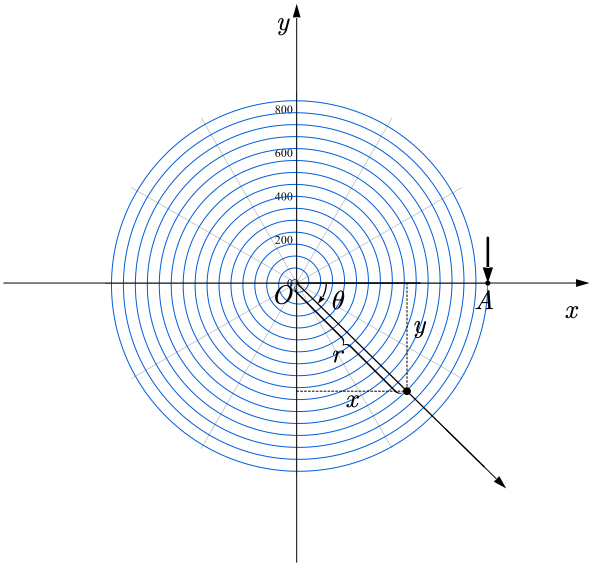


图 2　坐标系示意图

螺线的极坐标方程式形式如式 (2)：

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

为简化螺线方程，令 $a=0$，又有该等距螺线的螺距为 $0.55m$，所以按照螺线方程定义有 $b=0.55/2\pi$。初始时龙头所在的螺线圈数 $n=16$，所以不妨设 0 秒时 $\theta_0=32\pi$. 盘入过程中舞龙队顺时针行进，即龙头与原点的连线按照顺时针旋转，故而 θ 逐渐减小。

最终可得第一问的螺线方程为:

$$r=\frac{0.55}{2\pi}\theta \tag{3}$$

确定舞龙队位置 对于任意时间点的节点位置，采用先确定节点角度 θ, 再计算其在笛卡尔坐标系中对应的 x、y 的值的方式。

1) 首先，确定行进过程中龙头的位置，0 秒时，龙头角度 $\theta=32\pi$.
2) 设在极短时间 dt 内，龙头角度变化量为 $d\theta$, 推导过程如下:
   若曲线由极坐标方程 $r=r(\theta),\quad (\alpha\le\theta\le\beta)$ 给出，要导出它的弧长计算公式，只需要将极坐标方程化成参数方程，再利用参数方程下的弧长计算公式即可。
   曲线的参数方程为:

$$\begin{cases} x(\theta)=r(\theta)\cos\theta \\ y(\theta)=r(\theta)\sin\theta \end{cases} \tag{4}$$

   其中 θ 是参数，弧长元素 ds 为:

$$ds=\sqrt{r(\theta)^2+\left(\frac{dr(\theta)}{d\theta}\right)^2}d\theta. \tag{5}$$

   在本问中，弧长即龙头的运动路程，对上式积分得到 s 与 θ 的关系式:

$$s=\int_\alpha^\beta\sqrt{r(\theta)^2+\left(\frac{dr}{d\theta}\right)^2}d\theta \tag{6}$$

   又因为龙头以 $v=1m/s$ 的恒定速度前进，时间段 t 内龙头的运动路程 $s=v\cdot t$, 因此可以求得时间段 t 后的 θ。

$$-v\cdot t=\int_\alpha^\beta\sqrt{r(\theta)^2+\left(\frac{dr}{d\theta}\right)^2}d\theta \tag{7}$$

   即对于已知的初始角 α，可以解方程得到时间段 t 后的角 β.
   因此，可以得到在每一个时间，龙头把手的极坐标。再根据式 (4) 进行坐标系转换，得到龙头把手的 x,y 坐标。
3) 根据递推关系求解其余把手位置:
   对下一个板凳的前后把手进行考虑，如图3所示，这两个把手中心的位置在满足螺线方程的同时，还需要保证二者的欧几里得距离为板凳两孔中心之间的距离 l，这体现了板凳的刚性结构。

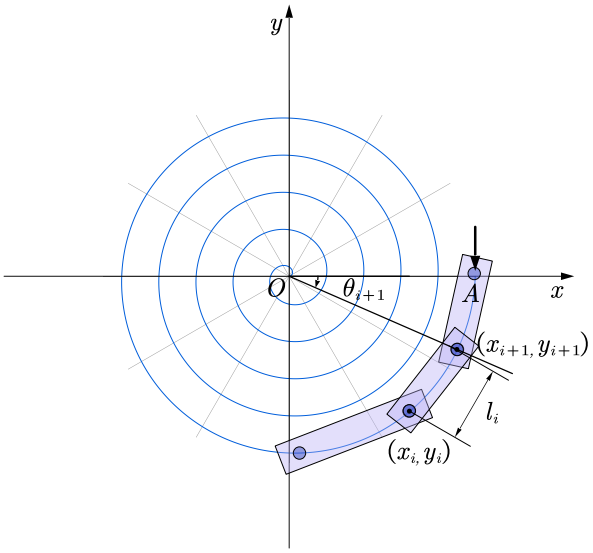


图 3　相邻把手位置示意图

不妨设在任意时刻 t 时，x_i，y_i 表示第 i 个节点的坐标，这一位置关系可以通过以下方程表示:

$$(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2 = l_i^2 \tag{8}$$

其中，l_i 表示第 i 个板凳的有效长度。龙头对应的的 l_0 值为 $2.86m$，龙身和龙尾的 l_i 值为 $1.65m$。

结合螺线方程，有:

$$(b\theta_i \cos\theta_i - x_{i-1})^2 + (b\theta_i \sin\theta_i - y_{i-1})^2 = l_i^2 \tag{9}$$

据此可递推计算出随后每个节点的 θ 值，再计算出相应的 x，y 值。

确定舞龙队速度　在物理学中，速度 v 的微分形式为:

$$v = \frac{ds}{dt} \tag{10}$$

在该问题中，路程变化量可用 dx 和 dy 进行表达，所以有:

$$v = \frac{\sqrt{(dx)^2 + (dy)^2}}{dt} \tag{11}$$

5.1.2 问题一模型求解

对于连续变量问题，考虑采用离散化进行处理。将时间划分成无数区间，时间步长 Δt 设置为 $0.1s$.

位置求解 对于式 (7) 中的积分求解，较为复杂，多轮时间迭代下，会大幅增加计算量，考虑使用梯形公式，进行近似求解。

式 (7) 可转化为:

$$-v\cdot\Delta t=\frac{\sqrt{b^2(\alpha)^2+b^2}+\sqrt{b^2(\beta)^2+b^2}}{2}\cdot(\beta-\alpha) \tag{12}$$

令 θ_i^j 表示第 i 个把手在第 j 个时间点转过的角度。对于龙头，其速度始终为 $v_0=1m/s$，则有:

$$-\Delta s=-v_0\cdot\Delta t=\int_{\theta_0^{j-1}}^{\theta_0^j}\sqrt{r^2+b^2}d\theta \tag{13}$$

与式 (12) 联立可得到 $\Delta\theta$ 与 Δt 的关系式:

$$\Delta\theta=-\frac{2v\Delta t}{b\left(\sqrt{1+\theta^2}+\sqrt{1+(\theta+\Delta\theta)^2}\right)} \tag{14}$$

其中 θ 为上一时间点的龙头 θ 值，$\Delta\theta$ 为下一时间点和上一时间点 θ 的差值。由于所有把手的 θ 都在减少，$\theta+\Delta\theta$ 即为龙头下一时间点的 θ 值。根据式 (14) 可以得到每个时间节点 j 时龙头的 θ 值，通过式 (4) 进行坐标系转换，得到龙头坐标。

已知每个时间节点的龙头坐标，由式 (9) 可以递推计算出接下来每个把手在该时间点 j 的极坐标 θ_i^j，通过式 (4) 坐标转换，得到每个时间点 j 每个把手的坐标 (x_i^j,y_i^j). 取出所有时间为整数时的把手坐标，即得舞龙队每个把手每秒的位置。

速度求解 使用式 (11) 求解每个时间点舞龙队各节点的速度。假设 0s 时所有节点（包括龙头）的速度均为 0m/s.

对于第一个时间节点 $j=0$ 的把手 i，选取下一个时间节点同一把手的 (x_i^{j+1},y_i^{j+1}) 进行计算，取一个 Δt.

$$v_i^0=\frac{\sqrt{(x_i^1-x_i^0)^2+(y_i^1-y_i^0)^2}}{\Delta t} \tag{15}$$

对于其余时间节点 $j>0$ 的把手 i，选取上一个时间节点和下一个时间节点同一把手的 $(x_i^{j-1},y_i^{j-1}),(x_i^{j+1},y_i^{j+1})$ 进行计算，取两个 Δt.

$$v_i^j=\frac{\sqrt{(x_i^{j+1}-x_i^{j-1})^2+(y_i^{j+1}-y_i^{j-1})^2}}{2\Delta t} \tag{16}$$

由此可得每个时间点 j 每个把手的速度 v_i^j. 取出所有时间为整数时的把手速度，即得舞龙队每个把手每秒的速度。

5.1.3 问题一结果与可视化

经程序计算，得到每秒舞龙队的位置与速度，其中部分关键节点如下表所示：

	0 s	60 s	120 s	180 s	240 s	300 s
龙头 x(m)	8.800000	5.875191	-3.987953	-3.077199	2.719822	4.347741
龙头 y(m)	0.000000	-5.695369	-6.367830	6.040052	-5.296507	2.459220
第 1 节龙身 x(m)	8.363823	7.502097	-1.331911	-5.318127	4.904173	2.322954
第 1 节龙身 y(m)	2.826543	-3.343184	-7.428510	4.263021	-3.450373	4.479080
第 51 节龙身 x(m)	-9.518732	-8.716800	-5.630777	2.773560	6.055194	-6.313100
第 51 节龙身 y(m)	1.341136	2.437296	6.302325	7.296324	-3.710957	0.309970
第 101 节龙身 x(m)	2.913983	5.774717	5.456528	2.021780	-4.807598	-6.321237
第 101 节龙身 y(m)	-9.918311	-7.939573	-7.490983	-8.444413	-6.464877	3.8040291
第 151 节龙身 x(m)	10.861726	6.599699	2.276484	0.879971	2.834596	6.958251
第 151 节龙身 y(m)	-2.047632	8.202854	9.755335	9.438429	8.446190	4.525555
第 201 节龙身 x(m)	4.555102	-6.706285	-10.642210	-9.235640	-7.368802	-7.368809
第 201 节龙身 y(m)	9.255098	8.962442	1.245146	-4.361302	-6.287727	-5.390994
龙尾（后）x(m)	-5.305444	7.446946	10.965795	7.294432	3.109857	1.631804
龙尾（后）y(m)	-10.676584	-8.729439	0.958827	7.580951	9.514505	9.330730

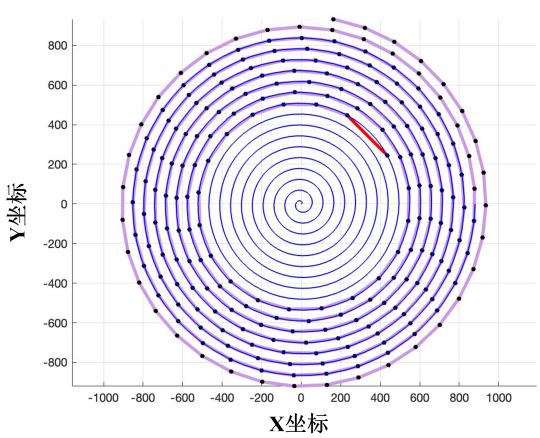
表 1　问题一中 300 秒内部分把手位置结果

	0 s	60 s	120 s	180 s	240 s	300 s
龙头 (m/s)	0	0.999844	0.999815	0.999773	0.999706	0.999582
第 1 节龙身 (m/s)	0	0.999805	0.999760	0.999690	0.999565	0.999294
第 51 节龙身 (m/s)	0	0.999511	0.999360	0.999113	0.998662	0.997677
第 101 节龙身 (m/s)	0	0.999305	0.999095	0.998759	0.998165	0.996926
第 151 节龙身 (m/s)	0	0.999153	0.998906	0.998519	0.997849	0.996492
第 201 节龙身 (m/s)	0	0.999036	0.998766	0.998346	0.997631	0.996210
龙尾（后）(m/s)	0	0.998993	0.998714	0.998284	0.997555	0.996115

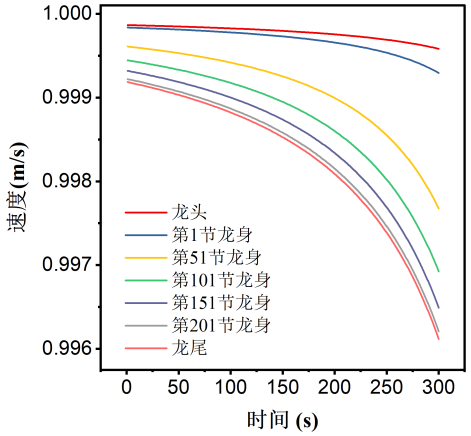
表 2　问题一中 300 秒内部分把手速度结果

对结果进行可视化和分析有助于更好地理解舞龙队的运动状态。

图 (a) 为舞龙队在 $300s$ 的位置示意图。该图中显示了板凳龙沿着等距螺线顺时针盘入后的运动轨迹。$300s$ 时，龙头从外向里经过约 7 圈，仍有板凳未进入第 16 圈 A 点后的螺线。



(a) 第 300 秒时的舞龙队位置



(b) 舞龙队速度随时间变化情况

图 (b) 为在这 $300s$ 的时间内，舞龙队几个重要节点的速度变化情况，由此可以得到以下结论：

a) 把手中心的运动速度，随着时间逐渐减小。

b) 与龙头相近的把手速度变化较慢，反之则变化较快。

c) 随着节数的增大，把手速度减小，但与此同时，相邻把手速度差值随之减小。

d) 龙头把手中心的速度应当始终保持为 $1m/s$，而随着时间的增长，龙头把手速度有小幅度的下降趋势，体现出模型的误差随着时间增大，但在可接受范围内。

5.2 问题二模型的建立与求解

5.2.1 问题二模型建立

当板凳之间发生碰撞，舞龙队不能再继续盘入。问题一中建立了舞龙队盘入的行进模型，可以计算出任意时间点各把手的位置。

为解决碰撞问题，需要考虑碰撞发生的位置和时刻。

首先考虑碰撞发生的位置。由于龙头、龙身宽度一致、龙头长度长于龙身，龙身跟随龙头前进，如龙头在行进过程中未发生碰撞，则龙身也不会发生碰撞。因此，碰撞一定发生在龙头与某节龙身之间。5展示了龙头与外圈龙身发生碰撞的状况。

从龙头周边考虑，不考虑相邻两节的碰撞，考虑龙头与外圈龙身间可能发生的碰撞，即考虑转动角度与龙头差值在 2π 之内的龙把手集合。

对于某一板凳，当其前后把手位置均确定时，可算出其四个角点的位置，方法如下：

假设前后把手的坐标分别为 (x_i^j, y_i^j) 和 (x_{i+1}^j, y_{i+1}^j)，板凳的长度为 l，宽度为 w。可由几何方法确定板凳的四个角点的位置。设前后把手的连线方向为向量 $\vec{v}$:

$$\vec{v} = (x_{i+1}^j - x_i^j, y_{i+1}^j - y_i^j) \tag{17}$$

设 $\vec{u}$ 为 $\vec{v}$ 的垂直单位向量（方向为宽度方向），可通过式 (18) 计算板凳四角点的位置：

$$\begin{cases} (x_1, y_1) = (x_i^j, y_i^j) + \frac{w}{2} \cdot \vec{u}, \\ (x_2, y_2) = (x_i^j, y_i^j) - \frac{w}{2} \cdot \vec{u}, \\ (x_3, y_3) = (x_{i+1}^j, y_{i+1}^j) + \frac{w}{2} \cdot \vec{u}, \\ (x_4, y_4) = (x_{i+1}^j, y_{i+1}^j) - \frac{w}{2} \cdot \vec{u}. \end{cases} \tag{18}$$

可以通过检测点在矩形内来判断是否发生碰撞。假设某一点 $P(x_p, y_p)$ 和矩形的四个角点 $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$，通过计算 P 与每个角点的夹角之和来判断点是否在矩形内部。

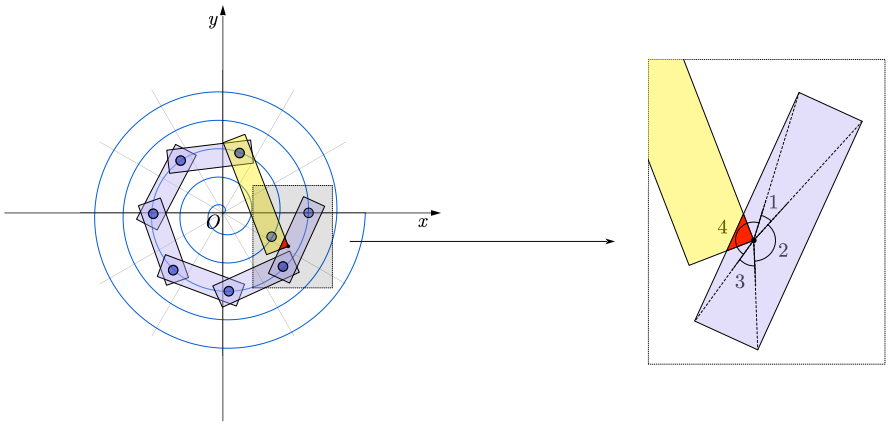


图 5　龙头与外圈龙身碰撞示意图

夹角的计算公式为：

$$\gamma = \arccos\left(\frac{\vec{v}_1 \cdot \vec{v}_2}{|\vec{v}_1||\vec{v}_2|}\right) \tag{19}$$

其中 $\vec{v}_1$ 和 $\vec{v}_2$ 分别是从点 P 指向矩形角点的向量。

如果龙头与龙身发生碰撞，则5中的四个夹角满足：

$$\gamma_1 + \gamma_2 + \gamma_3 + \gamma_4 = 2\pi \tag{20}$$

5.2.2 问题二模型求解

由问题一可知，前 300 秒内不会发生碰撞，因此从第 300 秒开始对 t 进行迭代。

***Step1:* 初始化**　设置一个布尔函数 $flag$，初始值为 0，用于每次迭代前判断是否要进行迭代。设置初始时间为第 300s，并引入第一问中 $300s$ 时龙头的位置。

***Step2:* 选取关键节点**　对每一秒龙头的 θ 值，采用与问题一相同的方法，通过勾股定理计算和龙头最近的 30 个节点位置。获取所有和龙头 θ 差值在 2π 里的 n 个节点 (x, y) 坐标。

***Step3:* 计算顶点坐标**　计算龙头四个顶点的 (x, y) 坐标。从关键节点中去除始终与龙头存在重合的第一节龙身前把手节点，由剩余 $n-1$ 个节点确定 $n-2$ 节可能发生碰撞的龙身。对每一节龙身，通过其前后把手中心位置计算对应的四个顶点 (x, y) 坐标。

***Step4:* 碰撞判断**　依次判断龙头的四个顶点是否在上述龙身的内部或边上，若是，则发生碰撞，记录下当前时刻，并设布尔函数 $flag$ 为 1。

***Step5:* 判断程序是否结束**　$flag$ 为 0 时，从第二步起重复上述步骤直到达到迭代最大次数；否则停止迭代，计算此时每个节点的位置和速度。

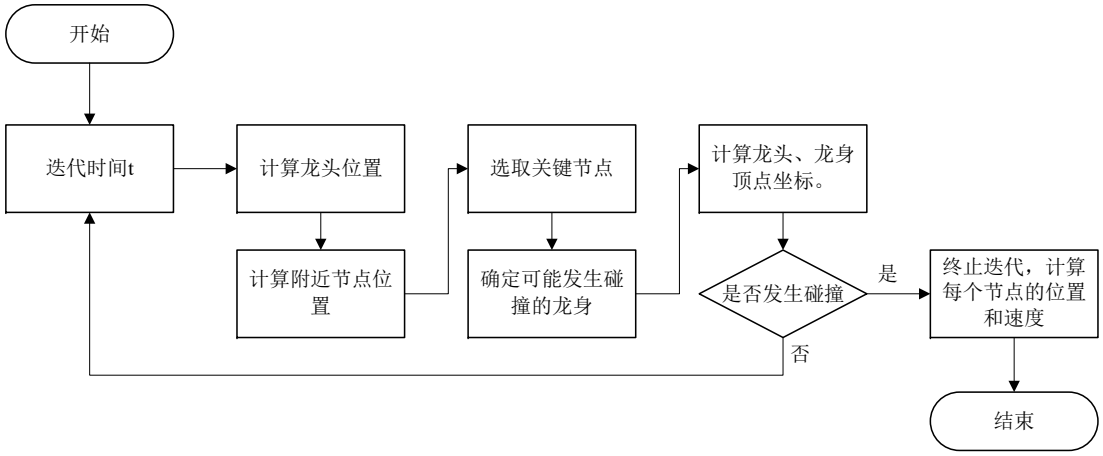


图 6　迭代算法流程图

5.2.3 问题二结果展示

解得舞龙队盘入的终止时刻为：$413s$。

计算出此时舞龙队几个重要节点的位置与速度，如下表所示：

	横坐标 x (m)	纵坐标 y (m)	速度 (m/s)
龙头	1.439959	1.764632	0.998002
第 1 节龙身	-1.415285	1.929488	0.989407
第 51 节龙身	1.549346	4.232091	0.974776
第 101 节龙身	-0.817515	-5.843423	0.972496
第 151 节龙身	0.687034	-6.987384	0.971567
第 201 节龙身	-7.928754	-0.949765	0.971062
龙尾（后）	1.236576	8.282728	0.970906

表 3　问题二结果：终止时刻各节点位置和速度

5.3 问题三模型的建立与求解

5.3.1 问题三模型建立

问题三要求确定最小螺距，使得龙头前把手能够沿着相应的螺线盘入到调头空间的边界，这个问题可以被视为一个优化问题。在保证板凳龙不发生碰撞的前提下，极小化螺线的螺距，使板凳龙能沿着原螺线进入调头空间。因此，本文根据前两题的运动分析，结合问题三的边界条件，构建了一个优化模型，考虑采用迭代方式进行最小螺距的搜索。该模型主要由约束条件和目标函数组成。

约束条件 模型在满足运动关系的前提下，主要由 2 个硬性条件进行约束：

1) 龙头能够进入调头空间

在改变螺距 p 的同时，将会使得前述的螺线方程发生变化，即改变螺线变化率 b，其中 b 满足：

$$b = \frac{dr}{d\theta} = \frac{p}{2\pi} \tag{21}$$

故本题螺线方程为：

$$r = \frac{p}{2\pi} \cdot \theta \tag{22}$$

龙头若进入调头空间，因为调头空间是以螺线中心为圆心、直径为 $9m$ 的圆形区域，则龙头把手中心与极点的距离将小于 $4.5m$，即 $r \leq 4.5$.

2) 碰撞避免

碰撞的检测采用问题二中的方法，即判断龙头的的一角是否在龙身所形成的矩形内部。同样引入布尔函数 $flag(t)$，满足：

$$flag(t) = \begin{cases} 1, & \text{若在时间 } t \text{ 前发生碰撞} \\ 0, & \text{若未发生碰撞} \end{cases} \tag{23}$$

需要保证对于每一个确定的螺距 p，在龙头进入调头空间前，$flag(t)$ 始终为 0.

目标函数 $\min \quad p$

5.3.2 问题三模型求解

遍历迭代算法 给定螺距 p 的取值区间为 $[a, b]$，给出迭代步长 Δp，进行遍历寻优。

自适应调整时间步长 例如初始迭代的 Δt 设置为 $0.5s$，在对于一个确定的螺距，进行时间迭代时，考虑龙头把手中心与原点的距离与 4.5 是否接近，若接近，则步长改为 $0.1s$，以提高搜索的精度。

设置惩罚函数 不妨设惩罚函数为 $Loss(p)$, 以刻画螺距 p 选择的好坏。

首先设置两个布尔函数 $flag1, flag2$, 初始值为 0, 分别表示是否有碰撞，龙头是否到达了调头空间。

每次时间迭代都判断龙头是否到达调头空间（计算龙头坐标和新原点距离判断），再判断龙头是否发生了碰撞（判断方法与第二问相同）。如符合条件成立则变化布尔函数 $flag1, flag2$ 的值。

通过判断 $flag1, flag2$ 的值，来确定惩罚函数值。

(1) $flag1 = 0, flag2 = 0$: 表示迭代时间结束前都没有到达调头空间或发生碰撞，这种情况螺距太大, 将该螺距对应的惩罚函数 $Loss(p)$ 函数值设置为极大数（代码中设为 1000）。

(2) $flag1 = 1, flag2 = 0$: 未达到调头空间就发生碰撞，该情况不成立, 将 x_1 的 $reward$ 函数值设置为大数（代码中设为 1000）。

(3) $flag1 = 0, flag2 = 1$: 到达调头空间还未发生碰撞, $Loss(p) = 0$, 并记录下该螺距值 p_{min}。

(4) $flag1 = 1, flag2 = 1$: 到达调头空间但发生了碰撞, 将 $Loss(p)$ 函数值设置为大数（代码中设为 1000）。

求解步骤 具体的运算步骤如下：

***Step1:* 确定搜索范围** 设置 $a = 0.1$，$b = 1$，以 $\Delta p = 0.005$ 为步长，进行逐步迭代。

***Step2:* 选择螺距，进行运动模拟** 对于确定的螺距 p，按照问题二中的运动方程再次进行时间迭代，步长 $\Delta t = 0.5s$，并判断龙头是否到达调头空间，以及是否发生了碰撞。如符合条件成立则变化布尔函数 $flag1, flag2$ 的值.

***Step3:* 返回惩罚函数值** 按照上述设置惩罚函数部分中的规则，返回 $Loss(p)$.

***Step4:* 判断是否终止迭代** 若 $Loss(p)$ 为 0，则终止迭代，返回 p_{min}，否则重复进行迭代。

5.3.3 问题三结果展示

经过编程与运算，得到最小螺距 p_{min} 为 **45.5**cm，即 **0.455**m. 当螺距 $p = 0.455m$ 时，在第 $227s$，龙头把手到达调头空间边界上坐标为 $(2.96035, -3.37344)$ 处。

图7展示了龙头进入调头空间时板凳龙的运动状态：

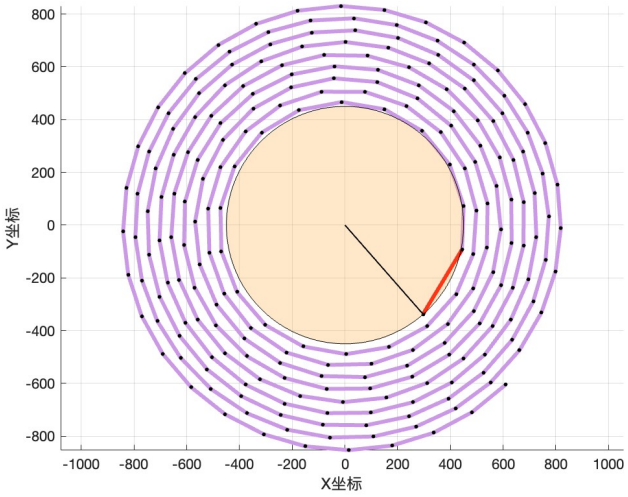


图 7　问题三进入调头空间示意图

5.4 问题四模型的建立与求解

5.4.1 问题四模型建立

问题四包括盘入、调头、盘出三个阶段，因此可将运动路径分为三部分，分别是盘入螺线部分，调头曲线部分和盘出螺线部分。其中，调头曲线又可以分为两段圆弧，分别记作圆弧一和圆弧二。

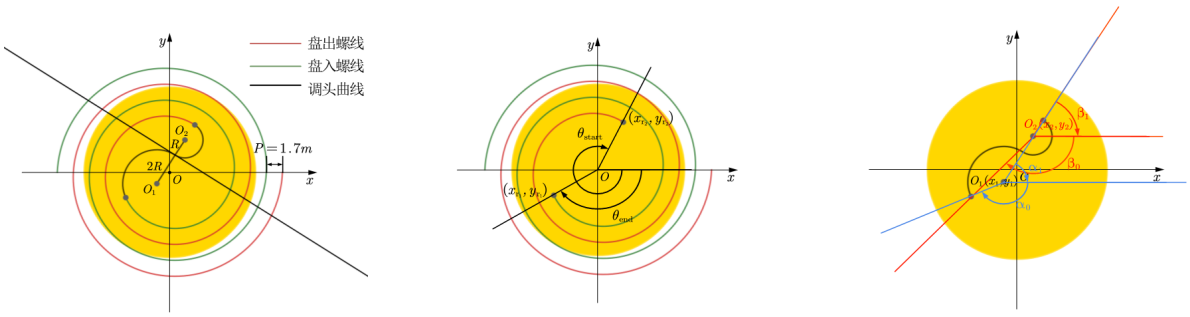


图 8　调头空间内部运动示意图

盘入螺线和盘出螺线　已知螺距 $P=1.7m$, 则 $b=1.7/2\pi$，所以当龙头把手在盘入螺线上运动时，对应的笛卡尔坐标为：

$$x_{in}=b\theta\cos\theta,\quad y_{in}=b\theta\sin\theta \tag{24}$$

同理，可得龙头把手在盘出螺线上运动时，对应的笛卡尔坐标：

$$x_{\text{out}}=b\theta\cos\theta,\quad y_{\text{out}}=b\theta\sin\theta \tag{25}$$

盘入时，极角 θ 逐渐减小；盘出时，极角 θ 逐渐增大。

螺线一点处的切线斜率: $f(\theta)=\dfrac{dy}{dx}=\dfrac{\frac{dy}{d\theta}}{\frac{dx}{d\theta}}=\dfrac{\sin\theta+\theta\cos\theta}{\cos\theta-\theta\sin\theta}$.

调头曲线 不妨设圆弧一和圆弧二对应的圆心分别为 $O_1(x_1,y_1)$ 和 $O_2(x_2,y_2)$，设圆弧二对应的圆的半径为 R，则圆弧一对应的圆的的半径为 $2R$.

对于在圆弧一上的点，其坐标方程满足:

$$x_{r_1}=x_1+2R\cos(\alpha),\quad y_{r_1}=y_1+2R\sin(\alpha) \tag{26}$$

其中 α 代表 $O_1\,(x_1,y_1)$ 与该点的连线和极轴（或 x 轴）的夹角（如图8）。

同理，在圆弧二上的点的坐标方程满足:

$$x_{r_2}=x_2+R\cos(\beta),\quad y_{r_2}=y_2+R\sin(\beta) \tag{27}$$

其中 β 代表 $O_2\,(x_2,y_2)$ 与该点的连线和极轴（或 x 轴）的夹角。

所以，圆弧上一点与圆心连线斜率即为 $\tan\gamma$, 可推导出切线斜率为: $g(\gamma)=-\dfrac{1}{\tan\gamma}$

曲线相切 设盘入螺线结束点对应的极角为 θ_{end}，该点与圆弧一圆心连线与 x 轴夹角为 α_0，盘出螺线开始点对应的极角为 θ_{start}, 该点与圆弧二

由于 S 型的调头曲线与两螺线相切，有以下约束条件:

1) 盘入螺线终点/盘出螺线起点使用螺线方程和圆弧方程算出的坐标相等:

$$\begin{aligned} x_1+2R\cos(\alpha_0)&=b\theta_{end}\cos\theta_{end},\quad y_1+2R\sin(\alpha_0)=b\theta_{end}\sin\theta_{end}\\ x_2+R\cos(\beta_1)&=b\theta_{start}\cos\theta_{start},\quad y_2+R\sin(\beta_1)=b\theta_{start}\sin\theta_{start}\end{aligned} \tag{28}$$

2) 螺线和圆弧交点处的切线斜率相等:

$$\frac{\sin\theta_{end}+\theta_{end}\cos\theta_{end}}{\cos\theta_{end}}=-\frac{1}{tan\alpha_0}\quad \frac{\sin\theta_{start}+\theta_{start}\cos\theta_{start}}{\cos\theta_{start}}=-\frac{1}{tan\beta_1} \tag{29}$$

3) 两圆弧交点坐标相等

$$x_1+2R\cos(\alpha_1)=x_2+R\cos(\beta_0),\quad y_1+R\sin(\alpha_1)=y_2+R\cos(\beta_0) \tag{30}$$

4) 两圆弧交点斜率相等

$$-\frac{1}{\tan(\alpha_1)}=-\frac{1}{\tan(\beta_0)} \tag{31}$$

空间限制 因为调头曲线位于问题三设定的调头空间内部，所以两圆弧上任意坐标点都必须满足:

$$x^2+y^2\le 4.5^2 \tag{32}$$

避免碰撞　本文在建立优化模型时，假设舞龙队在进入调头空间后，并非立即进行调头，而是在继续运行一段时间后，才进行螺线运动向圆周运动的转变，因此，在能够缩小 R，来极小化弧长的同时，需要避免碰撞的发生。

即选取的半径 R 需要满足在进行圆周运动时，龙头与龙身不会发生碰撞，若满足此条件，则显然有龙身和龙身之间也不会发生碰撞。

防止回退　倘若舞龙队进入调头空间后继续前进，会导致 R 缩小。由于龙头板凳具有刚体特性，可能出现龙头前把手沿着既定路线继续前进，龙头板凳后把手沿原路线后退的情况。在该优化模型中需通过约束 $R, \alpha_0, \alpha_1(\beta_0), \beta_1$ 来避免这种情况。

产生碰撞或回退的情况如下图所示：

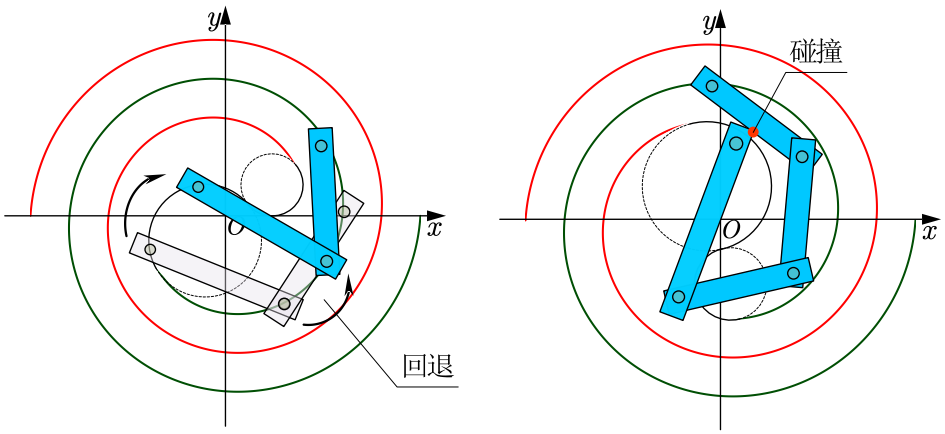


图 9　产生碰撞或回退的情况

根据上述的约束条件和目标函数，最终的优化模型可表示为：

目标函数

$$\min \quad 2R\left(\alpha_1-\alpha_0\right)+R\left(\beta_1-\beta_0\right) \tag{33}$$

约束条件

$$\text{s.t.}\quad \begin{cases} b\theta_{\text{start}} \le 4.5, \quad b\theta_{\text{end}} \le 4.5, \quad x^2+y^2 \le 4.5^2 \\ \alpha_1=\beta_0 \\ c \le R \le d \\ x_1+2R\cos\left(\alpha_0\right)=b\theta_{\text{end}}\cos\left(\theta_{\text{end}}\right), \quad y_1+2R\sin\left(\alpha_0\right)=b\theta_{\text{end}}\sin\left(\theta_{\text{end}}\right) \\ x_2+R\cos\left(\beta_1\right)=b\theta_{\text{start}}\cos\left(\theta_{\text{start}}\right), \quad y_2+R\sin\left(\beta_1\right)=b\theta_{\text{start}}\sin\left(\theta_{\text{start}}\right) \\ x_1+2R\cos\left(\alpha_1\right)=x_2+R\cos\left(\beta_0\right), \quad y_1+R\sin\left(\alpha_1\right)=y_2+R\cos\left(\beta_0\right) \\ f(\theta)=\dfrac{\sin\theta+\theta\cos\theta}{\cos\theta-\theta\sin\theta}, \quad g(\gamma)=-\dfrac{1}{\tan\gamma} \\ f\left(\theta_{\text{end}}\right)=g\left(\alpha_0\right), \quad f\left(\theta_{\text{start}}\right)=g\left(\beta_1\right) \end{cases} \tag{34}$$

决策变量包括：$(x_1, y_1), (x_2, y_2), R, \alpha_0, \alpha_1, \beta_0, \beta_1, \theta_{\text{start}}, \theta_{\text{end}}$

其中经过避免碰撞和防止回退分析，R 的取值在 c, d 之间

因为在圆弧一和圆弧二上运动时，半径不变，所以角速度 $\omega_{r_1} = v/2R, \omega_{r_2} = v/R$。

因此有，龙头把手在圆弧一上运行的时间 $t_1 = (\alpha_1 - \alpha_0)/\omega$，在圆弧二上运行的时间 $t_2 = (\beta_1 - \beta_0)/\omega$。

5.4.2 问题四模型求解

遗传算法 由于本问决策变量很多，传统的遍历方法时间成本较高，因此考虑用遗传算法对模型进行求解 [1]，具体的算法步骤如下所示 [2]：

Algorithm 1 遗传算法求解调头轨迹优化问题

Input: 种群大小 N，最大迭代次数 $MaxIter$，初始可行域 D，优化函数 Opt

Output: 最优解 $(\theta_{\text{start}}, \theta_{\text{end}}, R, \alpha_0, \alpha_1, \beta_0, \beta_1, x_1, y_1, x_2, y_2) \rightarrow ans$

$D^1 \leftarrow D$

$Population^0 \leftarrow$ random values

$k \leftarrow 1$

while $k \leq MaxIter$ **do**

$f_k(x^k) \leftarrow GeneticAlgorithm(Opt(ans^k), D_k, Population^{k-1})$

$Population^k \leftarrow$ get final population of $GeneticAlgorithm(Opt(ans), D_k, Population^{k-1})$

$D^{k+1} \leftarrow \{ans \in D^k : Opt(ans) \leq Opt(ans^k) + \lambda_s\}$

$k \leftarrow k + 1$

end while

return $ans^* =$ Best individual in $Population^s$

end function

***Step1:* 初始化种群** 初始化一个包含 N 个个体的种群，每个个体表示决策变量的一个组合。决策变量包括 θ_{start}、θ_{end}、R、α_0、α_1、β_0、β_1、x_1、y_1、x_2、y_2。所有决策变量的初始值在可行域内随机生成。

***Step 2:* 适应度函数计算** 对每个个体，基于目标函数 $2R(\alpha_1 - \alpha_0) + R(\beta_1 - \beta_0)$，以及约束条件计算其适应度。适应度函数不仅要考虑目标函数值，还要判断约束条件是否被满足，若不满足约束条件，则该个体的适应度被设为一个较大的惩罚值。

***Step 3:* 选择操作** 选择适应度较高的个体进入下一代。

***Step 4:* 交叉与变异** 对选择后的个体进行交叉操作，以生成新的个体。对交叉生成的后代个体进行变异操作，随机选择部分变量进行微小调整。例如，R 的变异可能是在原值基础上加上一个小幅变化。

***Step 5:* 终止条件** 重复 *Step 2* 到 *Step 5* 的过程，直到达到预定的最大迭代次数或种群适应度值收敛不再显著变化为止。最终保留适应度值最优的个体，即为问题的最优解。

得到最优个体的决策变量值 θ_{start}、θ_{end}、R 等后，计算相应的圆弧路径和时间，并根据此路径得到具体的调头运动轨迹，进而计算各时刻舞龙队的位置和速度。

5.4.3 问题四结果展示

由于该优化问题极为复杂，计算该优化问题时假设舞龙队一进入调头空间就开始调头，同时结束调头的点也在调头空间边界上。

经过编程计算，得到 $R = 1.507m, \alpha_0 = 3.9723, \alpha_1 = \beta_0 = 1.0109, \beta_1 = 1.6572, \theta_{end} = 16.6320, \theta_{start} = 13.4904$，最短的调头曲线为 $13.6148m$.

下表给出了 $t = -100s, -50s, 0s, 50s, 100s$ 时关键节点的位置和速度。

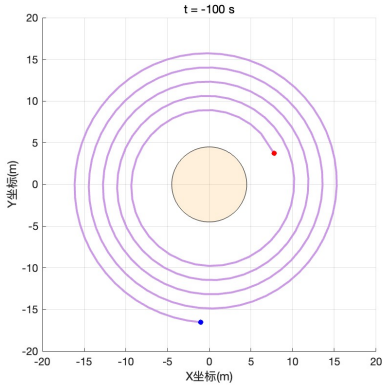
	-100 s	-50 s	0 s	50 s	100 s
龙头 x (m)	7.856437	6.653099	-2.859506	1.215096	-3.144702
龙头 y (m)	3.533177	1.703953	-3.456226	6.204820	-7.318135
第 1 节龙身 x (m)	6.343327	5.486646	-0.261686	3.770072	-0.380782
第 1 节龙身 y (m)	5.960130	4.315272	-4.652444	4.919637	-8.053219
第 51 节龙身 x (m)	-10.550009	-3.812061	2.268369	-1.782407	5.580551
第 51 节龙身 y (m)	3.022748	-8.881800	-7.829317	-6.039847	8.862289
第 101 节龙身 x (m)	-11.992044	10.018995	3.196336	-7.522610	-4.438682
第 101 节龙身 y (m)	-4.614958	-6.140931	10.045388	5.269582	11.603884
第 151 节龙身 x (m)	-14.373282	12.913316	-6.835547	-4.710481	-4.222891
第 151 节龙身 y (m)	-1.782449	-4.000265	10.443692	-10.336599	13.460377
第 201 节龙身 x (m)	-11.817189	10.376134	-6.697197	0.220229	6.435196
第 201 节龙身 y (m)	10.713553	-10.943102	12.474187	-13.177681	14.221862
龙尾 x (m)	-2.844121	2.031892	-3.740280	7.393896	-15.927596
龙尾 y (m)	-16.281667	15.557738	-14.325677	11.738315	-2.956442
龙尾（后）x (m)	-1.210125	0.389258	-2.129310	5.963754	-15.571131
龙尾（后）y (m)	-16.510918	15.713478	-14.682436	12.561233	-4.567477

表 4　问题四调头前后 100 秒范围内关键节点位置

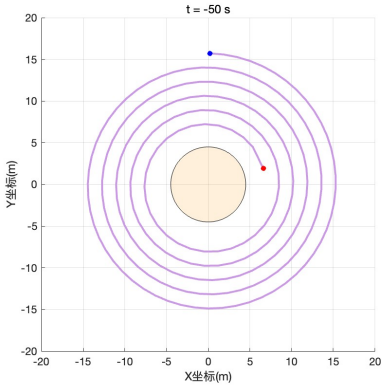
	-100 s	-50 s	0 s	50 s	100 s
龙头 (m/s)	0.999994	0.999991	0.999954	0.999990	0.999156
第 1 节龙身 (m/s)	0.999898	0.999752	0.989954	1.000351	0.999147
第 51 节龙身 (m/s)	0.999341	0.998631	0.986416	1.002371	0.998848
第 101 节龙身 (m/s)	0.999086	0.998237	0.985736	1.000839	0.998683
第 151 节龙身 (m/s)	0.998940	0.998037	0.985447	1.000371	0.998579
第 201 节龙身 (m/s)	0.998845	0.997915	0.985287	1.000145	0.926793
龙尾 (m/s)	0.998814	0.997877	0.985239	1.000081	0.926713
龙尾（后）(m/s)	0.998813	0.997876	0.985237	1.000078	0.926710

表 5　问题四调头前后 100 秒范围内关键节点速度

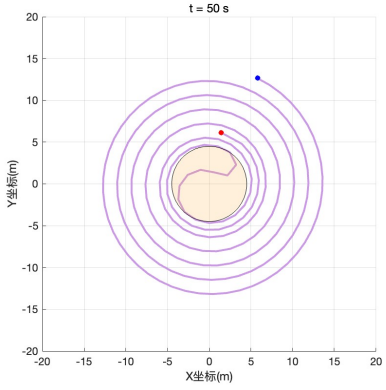
分别取 $t=-100s,-50s,50s,100s$，绘出舞龙队在每一时刻的位置。



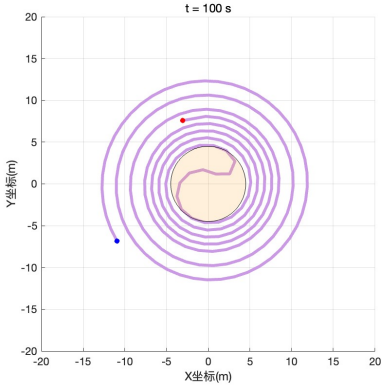
(a) 第-100 秒时的舞龙队位置



(b) 第-50 秒时的舞龙队位置



(c) 第 50 秒时的舞龙队位置



(d) 第 100 秒时的舞龙队位置

上图中，舞龙队转向经过的调头曲线分别与调头空间边界相切于两个点，在盘入阶段，随着时间推移，舞龙队的运动轨迹逐渐向中心靠拢。调头阶段里第一段圆弧半径较大，为第二段圆弧半径两倍，确保了运动的平滑性，同时也最大限度地利用了有限的调

头空间。完成调头后，舞龙队开始沿盘出螺线运动。这个阶段的运动特征与盘入阶段相似，但方向相反，板凳间距离变小，更有可能发生碰撞。可以观察到板凳龙逐渐从中心向外扩展，最终完成整个运动过程。

5.5 问题五模型的建立与求解

5.5.1 问题五模型建立

问题五在第四问的基础上，探究龙头的最大行进速度，使得舞龙队不仅满足第四问的运动学方程，还满足各把手的运动速度均小于 $2m/s$.

所以该问需要探寻龙身各把手速度和龙头速度的转换关系。而舞龙队的运动同样可以分成两个阶段：螺线运动和圆周运动。

螺线运动 与问题一相同，当舞龙队在螺线上运动时，已知龙头的运动速度，可以根据各把手之间的位置关系，以及时间和角度关系，唯一确定后续把手的位置和速度。

即同样满足位置关系：

$$(b\theta_i \cos\theta_i - x_{i-1})^2 + (b\theta_i \sin\theta_i - y_{i-1})^2 = l_i^2 \tag{35}$$

时间与角度关系：

$$-v \cdot t = \int_{\alpha}^{\beta} \sqrt{r(\theta)^2 + \left(\frac{dr}{d\theta}\right)^2}\, d\theta \tag{36}$$

进而对该阶段的运动仿照第一问进行类推，从初始时间节点，推算到各个不同的时间节点，及不同把手的速度值。

圆周运动 将舞龙队视为一个由 n 个刚性连接的节点组成的链状结构，其中 n 为板凳数.

当舞龙队的节点在圆弧上运动时，由第四问模型可知，其运动方程先后满足：

$$\begin{aligned} x_{r_1} &= x_1 + 2R\cos(\alpha), \quad y_{r_1} = y_1 + 2R\sin(\alpha) \\ x_{r_2} &= x_2 + R\cos(\beta), \quad y_{r_2} = y_2 + R\sin(\beta) \end{aligned} \tag{37}$$

龙头把手中心与原点 O 的夹角 θ_0 满足：

$$\theta_0 = \arctan\left(\frac{y_{r_1}}{x_{r_1}}\right) \tag{38}$$

对于后续的龙把手节点，考虑将位置关系直接使用角度关系进行刻画：

$$\begin{aligned} x_i &= x_{i-1} - l_i \cdot \cos(\theta_{i-1}) \\ y_i &= y_{i-1} - l_i \cdot \sin(\theta_{i-1}) \end{aligned} \tag{39}$$

其中 l_i 是第 i 个板凳两孔中心距离.

后续把手的速度 v 可通过将式 (39) 两端，对时间 t 的求导推得：

$$\begin{aligned} v_{x,i} &= v_{x,i-1} - l_i \cdot \frac{d\theta_{i-1}}{dt} \cdot \sin(\theta_{i-1}) \\ v_{y,i} &= v_{y,i-1} + l_i \cdot \frac{d\theta_{i-1}}{dt} \cdot \cos(\theta_{i-1}) \end{aligned} \tag{40}$$

其中 $\dfrac{d\theta_{i-1}}{dt}$ 是把手角速度, 考虑数值计算方法进行求解。

对于速度的约束条件, 可表示为：

$$\sqrt{v_{x,i}^2 + v_{y,i}^2} \leq 2, \quad \forall i = 0, 1, 2, ..., n \tag{41}$$

5.5.2 问题五模型求解

对于该单目标优化模型，在确定龙头把手速度 v_0 的迭代区间后，考虑采用黄金分割迭代优化算法，来快速寻找较优解，降低计算复杂度.

首先规定一个损失函数 $Loss(v)$，若该速度对应的运动模型，满足所有把手速度始终小于 $2m/s$，则为 0，否则为 1；

黄金分割迭代算法 给定龙头把手速度 v_0 的取值区间为 $[a, b]$, 给出迭代步长 Δv，进行黄金分割迭代 [3]。引入 λ_1，λ_2 满足：

$$\lambda_1 = a + (1 - \varphi)(b - a), \quad \lambda_2 = b - (1 - \varphi)(b - a) \tag{42}$$

其中 $\varphi = \dfrac{\sqrt{5} - 1}{2} \approx 0.618$.

分别计算决策变量 p 在这两点对应的 $Loss$ 函数值，分以下两种情况进行处理：

1) 若前者更小，则进行以下更改：

$$b = p_2, \quad p_2 = p_1, \quad p_1 = a + (b - a) \cdot \varphi \tag{43}$$

2) 若后者更小，则进行以下更改：

$$a = \lambda_1, \quad \lambda_1 = \lambda_2, \quad \lambda_2 = b - (b - a) \cdot \varphi \tag{44}$$

按照黄金分割比例进行区间迭代，能够迅速成比例缩短区间长度，当区间长度小于迭代阈值 ϵ 时，则停止迭代，返回 $(a + b)/2$.

5.5.3 问题五结果

经过黄金分割迭代算法，可以得到满足条件的龙头最大行进速度 $1.246m/s$。在这种情况下当前所有板凳的最大可能达到的速度为 $1.598m/s$。

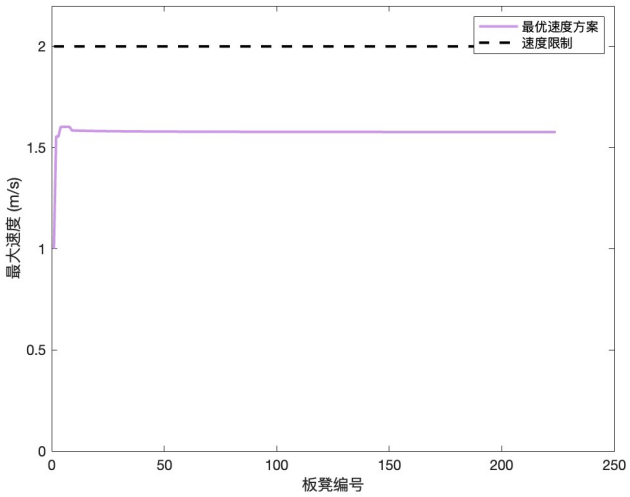
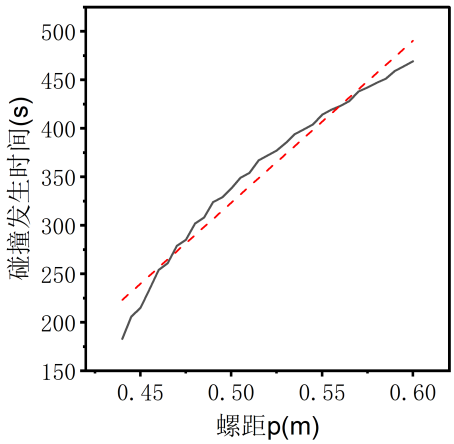


图 11　最优方案中每个板凳的最大速度

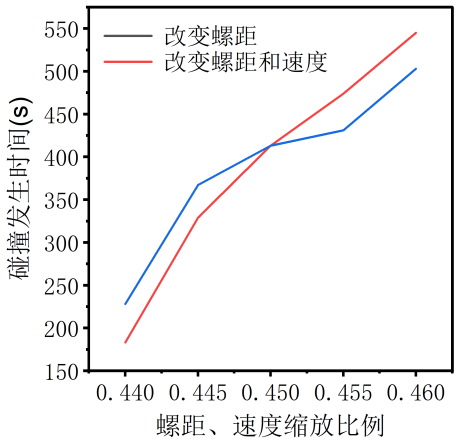
图 (11) 表示在龙头行进速度为 $1.246m/s$ 时所有板凳的最大可能达到的速度。由图可知，值得注意的是最大速度位于 $1.60m/s$ 附近, 而不是 $2m/s$ 附近，这是由黄金分割迭代算法所限制的。

六、 模型评价

6.1 灵敏度分析



(a) 碰撞时间随螺距变化图



(b) 碰撞时间随 pv 变化图

在第二问模型的基础上可以探究螺距和龙头速度的微小变化对碰撞时间产生的影响。

1) 由图（a）可知，螺距 p 在 0.55 周围约 0.1 的范围内浮动。碰撞时间和螺距大致呈线性变化关系。经过分析，当螺距 p 发生变化时，碰撞发生的地点相对于螺旋线位置基本不变，所以龙头走过的总路程 s，会随着 p 线性增加，在速度确定的情况下，从而碰撞时间随螺距变化呈线性关系。
2) 螺距 p 和龙头把手速度 v 同时呈相同比例缩放，缩放系数取值范围为 $[0.8, 1, 2]$. 理论上分析，当 v 增大时，碰撞所需时间减小，反之同理。即其与螺距 p 造成的影响相反。图（b）显示了当螺距 p 和速度 v 等比例缩放时，碰撞时间变化情况，表明螺距 p 对碰撞时间所造成的影响要大于速度 v 的改变所带来的影响。即可以得到碰撞时间对螺距 p 更加敏感。

6.2 模型优点

1. 模型对实际运动进行适当简化处理，能够较为精确地求解运动过程中各节点的位置和速度；
2. 模型提出利用点与矩形位置关系判断碰撞的方法，并成功地将运动模拟与碰撞检测相结合，用于各路径设计时的可行性判断；
3. 模型使用多种优化算法，确保在复杂的路径和约束条件下能够有效地求解最优运动轨迹和策略。

6.3 模型缺点

1. 模型主要考虑了板凳龙的几何特性而非动力学特性，可能导致在实际运动中的适用性有所限制；
2. 在长时间或复杂运动过程中，由于数值积分和逼近算法的使用，误差可能逐步累积；
3. 模型中的时间离散化处理可能受到步长精度的限制。

参考文献

[1] ANON. Genetic algorithms in search, optimization, and machine learning[C] // Choice Reviews Online : Vol 27. 1989 : 27 – 0936.

[2] 关锋. 遗传算法求解单目标柔性流水车间调度问题研究 [J]. 自动化应用, 2022(4) : 40 – 42+46.

[3] 张德宣, 吴钢伟, 邓辉云. 黄金比例分割法确定对称逐次超松弛迭代法的最佳松弛因子 [J]. 科技创新导报, 2010(25) : 217 – 219.

附录 A　问题一- MATLAB 代码

```
clear,clc
% 给定值
l1 = 286;
l2 = 165;
b = 55 / (2*pi);
dt = 0.1;
n = 300/dt+1;
v = 100; % 1m/s=100cm/s

% 初始化
theta = zeros(224,n);
theta(1,1) = 32*pi;
x = zeros(224,n);
y = zeros(224,n);
x_ans = zeros(224,301);
y_ans = zeros(224,301);

% 计算坐标
for i=1:n % 时间循环，i表示时间节点
    for j=2:224 % 后续节点循环，j表示节点
        x(j-1,i) = (b * theta(j-1,i)) * cos(theta(j-1,i)); % i个时间节点，第j-1个节点x轴坐标
        y(j-1,i) = (b * theta(j-1,i)) * sin(theta(j-1,i)); % i个时间节点，第j-1个节点y轴坐标

        if j == 2
            distance_func = @(t) ( ( (b * t) * cos(t) - x(j-1,i) )^2 + ( ( b * t) * sin(t) -
                y(j-1,i) )^2 ) - l1^2;
        else
            distance_func = @(t) ( ( (b * t) * cos(t) - x(j-1,i) )^2 + ( ( b * t) * sin(t) -
                y(j-1,i) )^2 ) - l2^2;
        end

        % 设置初始值
        initial_guess = theta(j-1,i) + 0.01; % 初始值一定要大于上一个节点的值
        lower_bound = theta(j-1,i) + 0.01; % 解的下界
        upper_bound = theta(j-1,i) + 2*pi; % 解的上界

        max_attempts = 500; % 最多尝试次数
        attempt = 0;
        while distance_func(lower_bound) * distance_func(upper_bound) > 0 && attempt <
            max_attempts
           upper_bound = upper_bound - 0.01; % 逐渐降低上界
            if upper_bound <= lower_bound
              % 若有需要，扩展下界
              lower_bound = lower_bound - 0.01;
```

```
            end
            attempt = attempt + 1;
        end

        if attempt == max_attempts
           error('Unable to find an interval with differing signs for fzero at index %d', i);
        end
        theta(j,i) = fzero(distance_func, [lower_bound, upper_bound]);
    end

    % 计算龙头节点的delta_theta
    delta_theta = 0;
    distance_func = @(t) ( 2 * v * dt ) / ( b * ( sqrt ( 1 + theta(1,i) ^ 2 ) + sqrt ( 1 + (
        theta(1,i) + t ) ^ 2 ) ) ) - t;

    % 设置初始值
    initial_guess = 0.01; % 初始值一定要大于上一个节点的值
    lower_bound = 0.01; % 解的下界
    upper_bound = 2*pi; % 解的上界

    max_attempts = 500; % 最多尝试次数
    attempt = 0;
    while distance_func(lower_bound) * distance_func(upper_bound) > 0 && attempt < max_attempts
       upper_bound = upper_bound - 0.01; % 逐渐降低上界
        if upper_bound <= lower_bound
          % 若有需要，扩展下界
          lower_bound = lower_bound - 0.01;
        end
        attempt = attempt + 1;
    end

    if attempt == max_attempts
       error('Unable to find an interval with differing signs for fzero at index %d', i);
    end

    delta_theta = fzero(distance_func, [lower_bound, upper_bound]);
    theta(1,i+1) = theta(1,i) - delta_theta;
end

for i = 1:n
    x(224,i) = (b * theta(224,i)) * cos(theta(224,i));
    y(224,i) = (b * theta(224,i)) * sin(theta(224,i));
end

% x，y坐标答案
for i = 2:301
    for j =1:224
```

```
        x_ans(j,i) = x(j,(i-1)*(1/dt));
        y_ans(j,i) = y(j,(i-1)*(1/dt));
    end
end

for i = 1:224
    x_ans(i,1) = x(i,1);
    y_ans(i,1) = y(i,1);
end

% 计算速度矩阵
speed = zeros(224,n); % 224个节点，n个时间点
speed_ans = zeros(224,301);
for i = 1:n-1
    for j = 1:224
        if i == 1
            speed(j,i) = 0;
        else
            % 其余时间点，用前后2dt秒的距离除以2dt
            distance = sqrt((x(j,i+1) - x(j,i-1))^2 + (y(j,i+1) - y(j,i-1))^2);
            speed(j,i) = distance / (2 * dt);
        end
    end
end

% 处理最后一个时间点的速度
for j = 1:224
    distance = sqrt((x(j,n) - x(j,n-1))^2 + (y(j,n) - y(j,n-1))^2);
    speed(j,n) = distance / dt;
end

% 速度答案
for i = 2:301
    for j =1:224
        speed_ans(j,i) = speed(j,(i-1)*(1/dt));
    end
end
for i = 1:224
    speed_ans(i,1) = 0;
end

% 将x_ans和y_ans交替写入Excel文件
filename = '.\result1.xlsx';
sheetname = '位置';

% 将数据乘以0.01
x_ans = x_ans * 0.01;
```

```
y_ans = y_ans * 0.01;

% 合并 x_ans 和 y_ans 并交替写入
[row_count, col_count] = size(x_ans);
combined_data = zeros(2 * row_count, col_count);

for i = 1:row_count
    combined_data(2*i-1, :) = x_ans(i, :); % 写入 x_ans 的一行
    combined_data(2*i, :) = y_ans(i, :); % 写入 y_ans 的一行
end

% 写入到 Excel，从第二行第二列开始
writematrix(combined_data, filename, 'Sheet', sheetname, 'Range', 'B2');
```

附录 B　问题二- MATLAB 代码

```
clear,clc
% 第一题给定值
l1 = 286;
l2 = 165;
b = 55 / (2*pi);
dt = 1;
v = 100; % 1m/s=100cm/s
flag = 0;
% 木板的尺寸（长度和宽度）
length1 = 341; % 龙头的长度
width1 = 30; % 龙头的宽度
length2 = 220; % 其余板的长度
width2 = 30; % 其余板的宽度

% 定义孔的距离
edge_dist1 = 27.5; % 孔到木板边缘的距离

theta = zeros(30,500);
theta(1,1) = 57.144840319137955; %假设从第300秒开始
x = zeros(30,500);
y = zeros(30,500);
for i=1:200 %时间循环，i表示秒
    if flag == 1
        break
    end
    for j=2:30 %后续节点循环，j表示节点
        x(j-1,i) = (b * theta(j-1,i)) * cos(theta(j-1,i)); %i秒，第j-1个节点x轴坐标
        y(j-1,i) = (b * theta(j-1,i)) * sin(theta(j-1,i)); %i秒，第j-1个节点y轴坐标
```

```
    if j == 2
    distance_func = @(t) ( ( (b * t) * cos(t) - x(j-1,i) )^2 + ( ( b * t) * sin(t) -
        y(j-1,i) )^2 ) - l1^2;
    else
    distance_func = @(t) ( ( (b * t) * cos(t) - x(j-1,i) )^2 + ( ( b * t) * sin(t) -
        y(j-1,i) )^2 ) - l2^2;
    end

    % 设置初始值
    initial_guess = theta(j-1,i) + 0.01; % 初始值一定要大于上一个节点的值
    lower_bound = theta(j-1,i) + 0.01; % 解的下界
    upper_bound = theta(j-1,i) + 2*pi; % 解的上界

    max_attempts = 500;%最多尝试次数
    attempt = 0;
    while distance_func(lower_bound) * distance_func(upper_bound) > 0 && attempt <
        max_attempts
      upper_bound = upper_bound - 0.01; % 逐渐降低上界
       if upper_bound <= lower_bound
        % 若有需要，扩展下界
        lower_bound = lower_bound - 0.01;
       end
       attempt = attempt + 1;
    end

    if attempt == max_attempts
      error('Unable to find an interval with differing signs for fzero at index %d', i);
    end
    theta(j,i) = fzero(distance_func, [lower_bound, upper_bound]);
    x(j,i) = (b * theta(j,i)) * cos(theta(j,i)); %i秒，第j个节点x轴坐标
    y(j,i) = (b * theta(j,i)) * sin(theta(j,i)); %i秒，第j个节点y轴坐标
end


head_front = zeros(1,2); %第i秒龙头前的x，y坐标
head_front(1,1) = x(1,i); %x坐标
head_front(1,2) = y(1,i); %y坐标
head_last = zeros(1,2); %第i秒龙头后的x，y坐标
head_last(1,1) = x(2,i); %x坐标
head_last(1,2) = y(2,i); %y坐标
% 计算木板的方向角（相对于X轴），通过两孔确定方向
theta1 = atan2(head_last(1,2) - head_front(1,2), head_last(1,1) - head_front(1,1)); %
    龙头的角度
% 定义旋转矩阵
R1 = [cos(theta1), -sin(theta1); sin(theta1), cos(theta1)]; % 龙头的旋转矩阵
% 相对于孔的位置定义未旋转的顶点
unrotated_corners1 = [
```

```
    -edge_dist1, -width1/2;
    length1 - edge_dist1, -width1/2;
    length1 - edge_dist1, width1/2;
    -edge_dist1, width1/2
];
head_board = (R1 * unrotated_corners1')' + head_front; % 第一块木板顶点坐标
%disp('第一块木板的顶点坐标: ');
%disp(head_board);
%disp(size(head_board));

%定义一个cell数组，用于存储符合条件的 (1, 2) 矩阵
matrix_array = {};
for k = 2:30
    if theta(1,i)-theta(k,i)<2*pi %只考虑与第一个节点差为2pi的节点
        matrix_array{end + 1} = [x(k,i),y(k,i)];
    end
end
matrix_array(1) = []; %去除第一块木板，防止干扰
board_x_y_array = zeros(2,4,length(matrix_array)-1); %存储有可能相交的矩阵的顶点x，y坐标

for k = 1:length(matrix_array)-1
    hole3 = matrix_array{k};
    hole4 = matrix_array{k+1};
    %disp(size(hole3));
    theta2 = atan2(hole4(2) - hole3(2), hole4(1) - hole3(1)); % 木板的角度
    % 相对于孔的位置定义未旋转的顶点
    unrotated_corners2 = [
        -edge_dist1, -width2/2;
        length2 - edge_dist1, -width2/2;
        length2 - edge_dist1, width2/2;
        -edge_dist1, width2/2
    ];
    R2 = [cos(theta2), -sin(theta2); sin(theta2), cos(theta2)]; % 木板的旋转矩阵
    corners_board2 = (R2 * unrotated_corners2')' + hole3; % 木板顶点坐标
    for q=1:4
        board_x_y_array(1,q,k) = corners_board2(q,1);
        board_x_y_array(2,q,k) = corners_board2(q,2);
    end
end
%分别判断龙头的四个点是否与其他板有交点
for k = 1:4
    temp_x_y = [head_board(k,1),head_board(k,2)]; %用于判断的龙头顶点
    for q = 1:length(matrix_array)-1
    temp_xv =
        [board_x_y_array(1,1,q),board_x_y_array(1,2,q),board_x_y_array(1,3,q),board_x_y_array(1,4,q)];
    temp_yv =
        [board_x_y_array(2,1,q),board_x_y_array(2,2,q),board_x_y_array(2,3,q),board_x_y_array(2,4,q)];
```

```
        [in, on] = inpolygon(temp_x_y(1,1),temp_x_y(1,2),temp_xv,temp_yv);
        if any(in == 1) || any(on == 1)
            disp(i);
            flag = 1;
            break; % 假设此代码在一个循环中，遇到 1 就会终止
        end

        end
    end

    %计算龙头节点的delta_theta
    delta_theta = 0;
    distance_func = @(t) ( 2 * v * dt ) / ( b * ( sqrt ( 1 + theta(1,i) ^ 2 ) + sqrt ( 1 + (
        theta(1,i) + t ) ^ 2 ) ) ) - t;
    % 设置初始值
    initial_guess = 0.01; % 初始值一定要大于上一个节点的值
    lower_bound = 0.01; % 解的下界
    upper_bound = 2*pi; % 解的上界

    max_attempts = 500;%最多尝试次数
        attempt = 0;
        while distance_func(lower_bound) * distance_func(upper_bound) > 0 && attempt <
            max_attempts
           upper_bound = upper_bound - 0.01; % 逐渐降低上界
            if upper_bound <= lower_bound
              % 若有需要，扩展下界
              lower_bound = lower_bound - 0.01;
            end
            attempt = attempt + 1;
        end

        if attempt == max_attempts
          error('Unable to find an interval with differing signs for fzero at index %d', i);
        end

        delta_theta = fzero(distance_func, [lower_bound, upper_bound]);

        theta(1,i+1) = theta(1,i) - delta_theta;
end
```

附录 C　问题三- MATLAB 代码

```
clear, clc
a = 30; % 下界
c = 70; % 上界
```

```
m = 1; % 收敛阈值
step_size = 0.5; % 步长
best_d = 0; % 答案
min_reward = 1000; % 最小的reward值

% 第一题给定值
l1 = 286;
l2 = 165;
dt = 0.5;
v = 100; % 1m/s=100cm/s
% 木板的尺寸（长度和宽度）
length1 = 341; % 龙头的长度
width1 = 30; % 龙头的宽度
length2 = 220; % 其余板的长度
width2 = 30; % 其余板的宽度
% 定义孔的距离
edge_dist1 = 27.5; % 孔到木板边缘的距离
flag = 0;

% 遍历以步长0.5遍历d的值
for d = a:step_size:c
    b = d / (2*pi);
    flag_1 = 0; % 用于标记是否有碰撞
    flag_2 = 0; % 用于标记是否到达掉头范围
    temp_theta = zeros(224, 1000);
    temp_x = zeros(224, 1000);
    temp_y = zeros(224, 1000);
    temp_theta(1, 1) = 32 * pi;
    temp_time = 0;

    if flag == 1
        break
    end
    % 开始运动模拟
    for i = 1:1000
        if flag_1 == 1 || flag_2 == 1
            break
        end

        for j = 2:224 % 后续节点循环，j表示节点
            temp_x(j-1, i) = (b * temp_theta(j-1, i)) * cos(temp_theta(j-1, i)); %
                i时间点，第j-1个节点x轴坐标
            temp_y(j-1, i) = (b * temp_theta(j-1, i)) * sin(temp_theta(j-1, i)); %
                i时间点，第j-1个节点y轴坐标

            if j == 2
                distance_func = @(t) ((b * t) * cos(t) - temp_x(j-1, i))^2 + ((b * t) * sin(t) -
```

```
            temp_y(j-1, i))^2 - l1^2;
    else
        distance_func = @(t) ((b * t) * cos(t) - temp_x(j-1, i))^2 + ((b * t) * sin(t) -
            temp_y(j-1, i))^2 - l2^2;
    end

    % 设置初始值
    initial_guess = temp_theta(j-1, i) + 0.01; % 初始值一定要大于上一个节点的值
    lower_bound = temp_theta(j-1, i) + 0.01; % 解的下界
    upper_bound = temp_theta(j-1, i) + 2*pi; % 解的上界

    max_attempts = 500; % 最多尝试次数
    attempt = 0;
    while distance_func(lower_bound) * distance_func(upper_bound) > 0 && attempt <
        max_attempts
        upper_bound = upper_bound - 0.01; % 逐渐降低上界
        if upper_bound <= lower_bound
            lower_bound = lower_bound - 0.01; % 若有需要，扩展下界
        end
        attempt = attempt + 1;
    end

    if attempt == max_attempts
        error('Unable to find an interval with differing signs for fzero at index %d', i);
    end

    temp_theta(j, i) = fzero(distance_func, [lower_bound, upper_bound]);
    temp_x(j, i) = (b * temp_theta(j, i)) * cos(temp_theta(j, i)); %
        i时间点，第j个节点x轴坐标
    temp_y(j, i) = (b * temp_theta(j, i)) * sin(temp_theta(j, i)); %
        i时间点，第j个节点y轴坐标
end

head_front = [temp_x(1, i), temp_y(1, i)]; % 第i时间点龙头前的x，y坐标

if norm(head_front) < 450
    flag_2 = 1;
    temp_time = i - 1;
elseif norm(head_front) == 450
    flag_2 = 1;
    temp_time = i;
end

% 检查碰撞
head_last = zeros(1,2); %第i秒龙头后的x，y坐标
head_last(1,1) = temp_x(2,i); %x坐标
head_last(1,2) = temp_y(2,i); %y坐标
```

```
% 计算木板的方向角（相对于X轴），通过两孔确定方向
theta1 = atan2(head_last(1,2) - head_front(1,2), head_last(1,1) - head_front(1,1)); %
    龙头的角度
% 定义旋转矩阵
R1 = [cos(theta1), -sin(theta1); sin(theta1), cos(theta1)]; % 龙头的旋转矩阵
% 相对于孔的位置定义未旋转的顶点
unrotated_corners1 = [
    -edge_dist1, -width1/2;
    length1 - edge_dist1, -width1/2;
    length1 - edge_dist1, width1/2;
    -edge_dist1, width1/2
];
head_board = (R1 * unrotated_corners1')' + head_front; % 第一块木板顶点坐标

%定义一个cell数组，用于存储符合条件的 (1, 2) 矩阵
matrix_array = {};
for k = 2:224
    if temp_theta(1,i)-temp_theta(k,i)<2*pi %只考虑与第一个节点差为2pi的节点
        matrix_array{end + 1} = [temp_x(k,i),temp_y(k,i)];
    end
end
matrix_array(1) = []; %去除第一块木板，防止干扰
board_x_y_array = zeros(2,4,length(matrix_array)-1); %存储有可能相交的矩阵的顶点x，y坐标

for k = 1:length(matrix_array)-1
    hole3 = matrix_array{k};
    hole4 = matrix_array{k+1};
    theta2 = atan2(hole4(2) - hole3(2), hole4(1) - hole3(1)); % 木板的角度
    unrotated_corners2 = [
        -edge_dist1, -width2/2;
        length2 - edge_dist1, -width2/2;
        length2 - edge_dist1, width2/2;
        -edge_dist1, width2/2
    ];
    R2 = [cos(theta2), -sin(theta2); sin(theta2), cos(theta2)]; % 木板的旋转矩阵
    corners_board2 = (R2 * unrotated_corners2')' + hole3; % 木板顶点坐标
    for q=1:4
        board_x_y_array(1,q,k) = corners_board2(q,1);
        board_x_y_array(2,q,k) = corners_board2(q,2);
    end
end

%分别判断龙头的四个点是否与其他板有交点
for k = 1:4
    temp_x_y = [head_board(k,1),head_board(k,2)]; %用于判断的龙头顶点
    for q = 1:length(matrix_array)-1
        temp_xv =
```

```
                [board_x_y_array(1,1,q),board_x_y_array(1,2,q),board_x_y_array(1,3,q),board_x_y_array(1,4,q)]
            temp_yv =
                [board_x_y_array(2,1,q),board_x_y_array(2,2,q),board_x_y_array(2,3,q),board_x_y_array(2,4,q)]
            [in, on] = inpolygon(temp_x_y(1,1),temp_x_y(1,2),temp_xv,temp_yv);

            if any(in == 1) || any(on == 1)
                disp(i);
                flag_1 = 1;
                break; % 假设此代码在一个循环中，遇到 1 就会终止
            end

        end
    end

    % 计算龙头节点的delta_theta
    distance_func = @(t) (2 * v * dt) / (b * (sqrt(1 + temp_theta(1, i)^2) + sqrt(1 +
        (temp_theta(1, i) + t)^2))) - t;
    initial_guess = 0.01;
    lower_bound = 0.01;
    upper_bound = 2 * pi;

    attempt = 0;
    while distance_func(lower_bound) * distance_func(upper_bound) > 0 && attempt <
        max_attempts
        upper_bound = upper_bound - 0.01;
        if upper_bound <= lower_bound
            lower_bound = lower_bound - 0.01;
        end
        attempt = attempt + 1;
    end

    if attempt == max_attempts
        error('Unable to find an interval with differing signs for fzero at index %d', i);
    end

    delta_theta = fzero(distance_func, [lower_bound, upper_bound]);
    temp_theta(1, i+1) = temp_theta(1, i) - delta_theta;
end
%d = d - 30/2;
% 计算reward函数值
if flag_1 == 1
    f_x = 1000;
elseif flag_2 == 1
    f_x = (norm([temp_x(1, temp_time), temp_y(1, temp_time)]) - 450);
else
    f_x = 1000;
end
```

```
    % 更新最优值
    if f_x < 1000
        min_reward = f_x;
        best_d = d;
        flag = 1;
    end
end

fprintf('最优的d值为: %.2f, 最小reward为: %.2f\n, 时间为: %.1f\n', best_d, min_reward,temp_time);
```

附录 D　问题四- MATLAB 代码

```
function optimize_model
    % 初始变量上下界
    lb = [-inf, -inf, -inf, -inf, 0, -pi, -pi, -pi, -pi, 0, 0]; % 下界
    ub = [inf, inf, inf, inf, inf, pi, pi, pi, pi, inf, inf]; % 上界

    % 使用遗传算法进行优化
    options = optimoptions('ga', 'Display', 'iter', 'PopulationSize', 100, 'MaxGenerations',
        200);
    [x, fval] = ga(@objective_function, 11, [], [], [], [], lb, ub, @nonlinear_constraints,
        options);

    % 显示结果
    disp('优化结果:');
    disp(x);
    disp(['目标函数值: ', num2str(fval)]);
end

% 目标函数
function f = objective_function(x)
    % 提取变量
    R = x(5);
    alpha0 = x(6);
    alpha1 = x(7);
    beta0 = x(8);
    beta1 = x(9);

    % 计算目标函数
    f = 2 * R * (abs(alpha1 - alpha0)) + R * (abs(beta1 - beta0));
end

% 非线性约束
function [c, ceq] = nonlinear_constraints(x)
```

```
    % 提取变量
    x1 = x(1); y1 = x(2); x2 = x(3); y2 = x(4);
    R = x(5);
    alpha0 = x(6); alpha1 = x(7); beta0 = x(8); beta1 = x(9);
    theta_start = x(10); theta_end = x(11);
    b = 170 / (2 * pi);

    % 非线性等式约束
    ceq = zeros(8, 1);
    ceq(1) = x1 + 2 * R * cos(alpha0) - b * theta_end * cos(theta_end);
    ceq(2) = y1 + 2 * R * sin(alpha0) - b * theta_end * sin(theta_end);
    ceq(3) = x2 + R * cos(beta1) - b * theta_start * cos(theta_start);
    ceq(4) = y2 + R * sin(beta1) - b * theta_start * sin(theta_start);
    ceq(5) = x1 + 2 * R * cos(alpha1) - x2 - R * cos(beta0);
    ceq(6) = y1 + 2 * R * sin(alpha1) - y2 - R * cos(beta0);
    ceq(7) = alpha1 - beta0; % alpha1 = beta0
    ceq(8) = f(theta_end) - g(alpha0); % 使用自定义函数

    % 非线性不等式约束
    c = zeros(10, 1);
    c(1) = b * theta_start - 450; % b * theta_start <= 450
    c(2) = b * theta_end - 450; % b * theta_end <= 450
    c(3) = -R;
    c(4) = -theta_start;
    c(5) = -theta_end;
    c(6) = ((x1 + 2 * R * cos(alpha1))^2 + (y1 + 2 * R *sin(alpha1))^2) - 450^2;
    c(7) = R - 300;
    c(8) = 170 - R;
    c(9) = (alpha1 - alpha0) - pi;
    c(10) = (beta1 - beta0) - pi;
    %c(11) = pi/2 - (alpha1 - alpha0);
    %c(12) = pi/2 - (beta1 - beta0);
end

% 自定义函数 f 和 g
function result = f(theta)
    result = (sin(theta) + theta * cos(theta)) / (cos(theta) - theta * sin(theta));
end

function result = g(gamma)
    result = -1 / tan(gamma);
end
```

附录 E　问题五- MATLAB 代码

```
    figure('Name', '最优方案中');

% 原始板凳编号
bench_indices = 1:length(max_velocities);

% 对前20个板凳设置较大的间距，剩余的板凳按正常顺序排列
x = [linspace(1, 20, 20), 21:length(max_velocities)];

% 使用自定义颜色 [0.8 0.6 0.9]
plot(x, max_velocities, 'Color', [0.8 0.6 0.9], 'LineWidth', 2);
hold on;

% 绘制速度限制线
plot([1, length(max_velocities)], [2, 2], 'k--', 'LineWidth', 2);
% 设置纵坐标上限为2.5
ylim([0 2.2]);
%title('最优方案每个板凳的最大速度');
xlabel('板凳编号');
ylabel('最大速度 (m/s)');
legend('最优速度方案', '速度限制');
```