基于几何模型的板凳龙运动路径问题

摘要

“板凳龙”是我国浙闽地区的一项元宵节习俗，村民们将板凳首尾相连组成长龙，龙头在前领头，龙身龙尾相随盘旋。在保证舞龙队能够自如地盘入和盘出的情况下，盘龙所需要的面积越小、行进速度越快，则观赏性越好。本文将盘龙过程抽象为数学模型，在各板凳数据已知的条件下研究不同螺距、不同龙头行进速度时，各时刻下整个舞龙队的位置和速度，为深入研究如何优化盘龙所需的面积和行进速度提供了有效的帮助。

针对问题一，舞龙队沿一螺距已知的等距螺线顺时针盘入，各把手中心严格位于螺线上。本文将盘入螺线转化为**极坐标方程**，采用积分的方法建立起龙头前把手所处位置与速度和时间的关系，得到其每个时刻的龙头前把手坐标。后面本文采用建立**位置迭代公式**的方法，由龙头前把手位置，得到舞龙队各前把手位置。为解决各时刻舞龙队各前把手速度问题，本文根据得到的位置坐标，建立相邻两把手**速度迭代公式**。结合已知的龙头前把手恒定速度，本文求解出各个时刻整个舞龙队的位置和速度。

针对问题二，在舞龙队沿着螺线盘入的过程中，随着龙头前把手逐渐接近圆心，由于螺距的限制，板凳之间可能会发生碰撞。本文首先分析整个舞龙队盘入的过程，发现只要龙头与第一节龙身外部四个角点在某一位置不与其他部分发生碰撞，则后续龙身在此处也不会发生碰撞，由此确定了最早发生碰撞的四个角点，由此确定了可能发生碰撞的**四个角点**。在问题一中本文确定了各个时刻整个舞龙队的位置，由于角点与其所在板凳的把手中心的相对位置固定，本文得以确定角点的位置。进而采用几何方法，计算角点到其他板凳前后把手中心所在直线的**距离**，以此判定是否发生碰撞。从而得到了舞龙队不能再继续盘入的位置，进而计算得不能再继续盘入的时间。

针对问题三，舞龙队顺时针盘入等距螺线后，将会逆时针盘出，因此需要一定的调头空间。在螺距已知的情况下，结合问题一、问题二的模型，本文能够得出在不同的螺距情况下舞龙队盘入的终止位置.因此，本文首先通过调整螺距，首先得到使得终止时刻龙头前把手中心坐标位于调头空间内和调头空间外的两个螺距，再使用十分法，精确求得使终止时刻龙头前把手中心坐标恰好位于调头空间边界的螺距。

针对问题四，题目给定盘入螺线的螺距、调头空间，且盘出螺线与盘入螺线关于螺线中心呈中心对称。本文首先通过几何方法证明了无法通过调整两段圆弧各自半径的比例使得调头路径（由两段圆弧相切连接而成的S形曲线，与盘入、盘出螺线相切）变短。随后，本文设定一符合题意的调头路径，将该路径按顺序分为四段弧线与三个关键节点。与问题一类似，确定调头路径的方程后，可以根据龙头前把手行进的速度得到其各个时刻的坐标。在构建位置、速度迭代公式前，本文首先构建了一个**判断函数**，用以根据板凳前把手中心的坐标判断后把手中心位于哪段弧线。判断函数构建完成后，本文分七种情况讨论了板凳前后把手中心可能所处哪一部分弧线，并利用几何方法与关联速度的思想，建立了这七种情况各自的**位置、速度迭代公式**，从而将问题一的位置、速度迭代公式推广到了整个盘龙路径，即盘入、调头、盘出。由此求得了从调头前一段时间到调头后一段时间内各个时刻整个舞龙队的位置和速度。

针对问题五，在舞龙队调头过程中，龙头前把手行进速度始终保持不变，但其余各把手的速度会随着位置的变化而发生变化。本文通过分析问题四所得结果，精确测试对象，找到了速度会达到整个舞龙队最大值的目标把手，并分析出在龙头进入盘出螺线的过程中，目标把手的速度将会达到最大值，进而通过python程序求解出在该过程中目标把手达到最大速度时龙头前把手精确位置。最后利用问题四的速度迭代公式，建立起目标把手与龙头前把手的速度关系，通过限制目标把手中心速度不超过上限，求得龙头的最大行进速度。

关键词：“板凳龙”，等距螺线，极坐标方程，关联速度，位置迭代，速度迭代，判断函数。

一　问题背景与重述

1.1　问题背景

我国浙闽地区有一项盛大的元宵节习俗——“板凳龙”：村民们把一节节的板凳钻孔连接，组成几十至上百节的板凳长龙。演出时，龙头在前领头，龙身和龙尾相随着将长龙盘旋成圆盘状。在能够自如盘入盘出的条件下，若是能够减少盘龙面积、加快行进速度，观赏性将会进一步提升。

1.2　问题要求

假设有一由223节板凳组成的板凳龙，其中第1节为龙头，接着221节龙身，最后1节为龙尾。龙头的板长为341cm，龙身与龙尾的板长均为220cm，板凳的板宽均为30cm，相邻两条板凳钻孔并通过把手连接，孔径为5.5cm，钻孔中心距最近的板头27.5cm。各部分的详细尺寸数据与连接方式如图1.1、1.2、1.3所示：

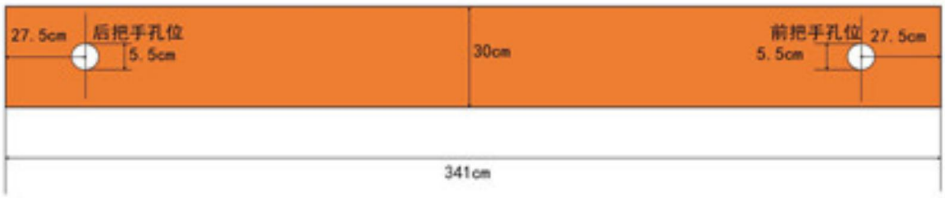

图 1.1: 龙头俯视数据图

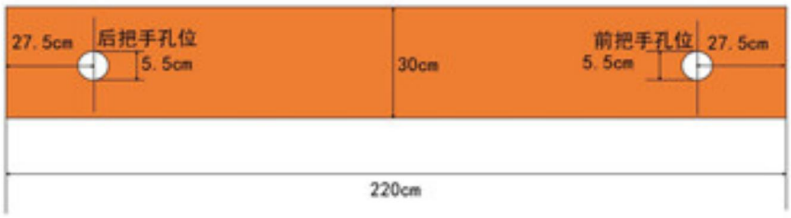

图 1.2: 龙身、龙尾俯视数据图

图 1.3: 板凳连接方式正视图

我们需要解决以下问题：

1. 有一舞龙队使用上述板凳龙，沿着螺距为55cm的等距螺线顺时针盘入，各把手中心均位于螺线上，龙头前把手的前进速度始终为1m/s。初始时刻，即$t = 0$s时，龙头位于螺线第16圈A点处。通过建立合适的数学模型，求解出从$t = 0$s至$t = 300$s为止，每秒各节板凳的前把手中心以及龙尾后把手中心的位置和速度。盘入螺线的起点以及方向如图1.4所示：

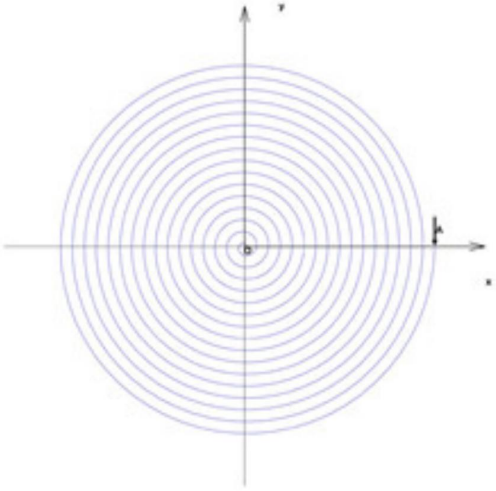

图 1.4: 盘入螺线示意图

2. 舞龙队继续向内盘入问题一中的螺线，求解在何时刻舞龙队不能再继续盘入，即板凳之间即将发生碰撞的时刻，并求出该时刻下各节板凳的前把手中心以及龙尾后把手中心的位置和速度。

3. 考虑一个螺距为d(m)的等距螺线，下称盘入螺线，舞龙队顺时针沿盘入螺线盘入后，将会切换为逆时针沿着盘出螺线（盘出螺线与盘入螺线关于螺线中心呈中心对称）盘出。因此，舞龙队需要一定的调头空间。在设定调头空间是以螺线中心为圆心、直径为9m的圆形区域的情况下，求最小的螺距d(m)，使得龙头前把手能够沿着盘入螺线盘入到调头空间的边界。调头空间如图1.5所示：

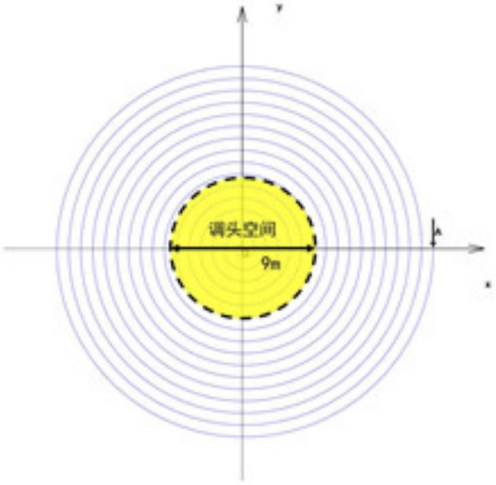

图 1.5: 调头空间示意图

4. 对于给定螺距$d = 1.7$m的盘入螺线，沿用问题三中对调头空间的设定，舞龙队需要在该调头空间内完成调头。调头路径是由两段圆弧相切连接成的S形曲线，前后

两段圆弧半径之比为2，该路径与盘入、盘出螺线均相切。尝试通过调整圆弧的方式，在保持相切关系的情况下给出尽可能短的调头路径。若龙头前把手的前进速度始终为1m/s，求出从调头前至调头后这段时间内每秒各节板凳的前把手中心以及龙尾后把手中心的位置和速度。

5. 舞龙队沿问题四设定的最短调头路径前进，假定龙头的前进速度保持为v(m/s)，求最大的v(m/s)使得该过程中舞龙队各把手的速度均不超过2m/s。

二　问题分析

2.1　问题一的分析

对于问题一，已知盘入螺线的螺距和起点，本文能够建立该螺线对应的极坐标方程。由于龙头前把手沿盘入螺线的行进速度恒定，可以结合已经建立起的极坐标方程，采用积分的方式去确定其某段时间内走过的路径长度与走过的角度的关系，进而得到每一时刻的位置坐标。由此，本文能够利用极坐标方程与每节板凳的数据，建立起由前把手中心位置得到后把手中心位置的迭代公式。

在舞龙队盘入的过程中，虽然每个把手中心的前进速度各不相同，但不难看出，同一板凳上前后把手中心沿板凳中心所在直线的速度是一样的。利用这一特性，本文通过几何方法得到前后把手中心速度方向与板凳中心所在直线的夹角，进而得到同一板凳上前后把手中心速度的关系式，建立起由前把手中心速度得到后把手中心速度的迭代公式。

2.2　问题二的分析

对于问题二，在舞龙队沿着螺线盘入的过程中，随着龙头前把手逐渐接近圆心，由于每块板凳有自己固定的形状与体积，因此在接近于螺线中心的过程中，由于螺距的限制，可能会有一个时刻，舞龙队的板凳之间发生了碰撞。

经过分析，由于龙身与龙尾所有板凳形状完全相同，并且在盘入过程中走过的路径也相同。因此这些板凳是否发生碰撞只需要考虑第一节龙身是否与其他板凳发生碰撞，因为若在走过的路径中，第一节龙身未与其他板凳发生碰撞，后面的龙身自然也不会与其他板凳发生碰撞。故龙靠前部分只有龙头与第一节龙身可能会与其他板凳发生碰撞，并且显然只有这两块板外部的四个角点A_1、A_2、A_3、A_4可能会与其他板凳相撞，如下图所示：

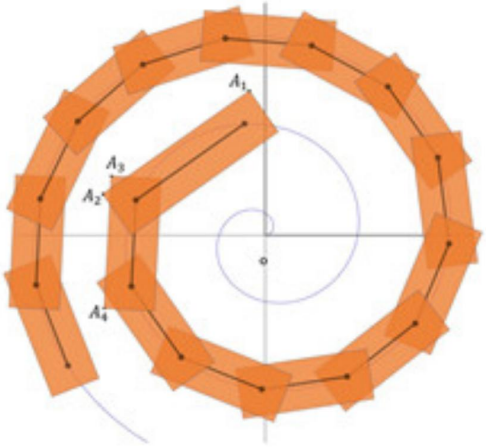

图 2.6: 板凳龙盘入示意图（橙色矩形为板凳）

由于问题一中，我们已经确定了各个时刻各把手中心位置，并且，上面提到的两块板凳把手中心与板凳的角的相对位置是固定的，本文由此计算出每个时刻各个角点的位置坐标。同时，在龙头外一层螺线上分布着一圈板凳，板凳的两个把手中心连接形成的线段是螺线的弦，并且可以利用问题一中所得到的坐标求出每条弦所在直线的解析式。因此本文利用角点到弦所在直线的距离来刻画碰撞与否，当距离小于板凳的半宽时说明已经发生了碰撞。

2.3 问题三的分析

结合问题一问题二，本文能够求解出当螺距已知时舞龙队盘入的终止时刻，以及此时舞龙队各把手的位置。对于问题三的问题要求，只需要通过调整螺距，使得在该螺距对应的终止时刻时，龙头前把手中心的位置恰好位于调头空间的边界上。

由此，可以先确定出两个螺距，使得在其各自的终止时刻时，龙头前把手中心的位置分别位于题设要求的调头空间内和调头空间外。再采用十分法，精确求解满足要求的最小螺距。

2.4 问题四的分析

对于问题四的第一问，即能否通过调整圆弧使得调头曲线变短，由于调头路径是由两段相切的圆弧构成的S型曲线，并且两段圆弧分别与盘入盘出螺线相切，通过分析其几何特性，本文通过计算判断是否能通过调整两段圆弧的半径比例使调头路径变短。

对于问题四的第二问，首先有掉头路径如图2.7所示：

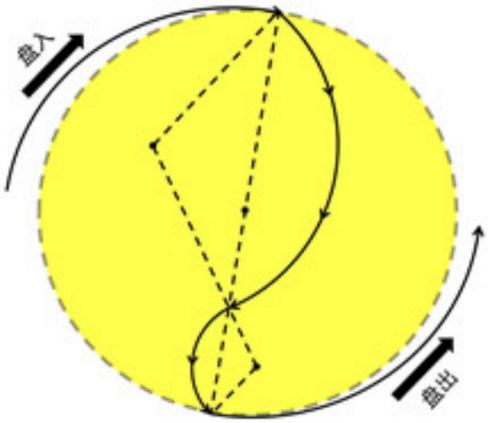

图 2.7: 调头空间（黄色）内掉头路径示意图

此路径有四段弧线，本文分别称作盘入螺线、第一段圆弧、第二段圆弧、盘出螺线。同时，此路径有三个关键节点，分别是盘入螺线与第一段圆弧的交点、两段圆弧的交点以及第二段圆弧与盘出螺线的交点。

首先，与问题一中的思路类似，当确定了调头路径的方程以后，根据龙头前把手的速度，可以得到各个时刻龙头前把手的位置坐标。本文先构建了一个判断函数：根据一块板凳的前把手的位置，判断该板凳后把手所位于的弧线

当一块板凳的前后两个把手均位于螺线上时，位置坐标的推导公式与问题一中相似；当一块板凳的前后两个把手均位于同一段圆弧上时，或者当一块板凳的前后两个把手分别位于两段不同的弧线时，根据前把手的位置，再利用几何知识，推导出后一把手位置。这样就可以通过前一个把手位置，通过判断函数判断后一个把手所在圆弧，即这两个把手的位置情况，再利用上面的位置推导公式，将问题一中的位置迭代公式推广到

了在整个路径中运动时位置的迭代公式。进而，与问题一的思路类似，通过位置迭代公式，本文得到各个时刻，每块板凳前把手中心位置的坐标。

当一块板凳的前后两个把手均位于螺线上时，速度的推导公式与问题一中相似；当一块板凳的前后两个把手均位于同一段圆弧上时，两把手速度相同；当一块板凳的前后两个把手分别位于两段不同的弧线时，可以根据前把手与后把手的位置坐标以及其分别所处的曲线，推导出前后把手中心速度的方向，以及连接前后把手中心位置线段所在直线的方向。利用直线夹角公式分别计算出前后把手中心速度方向与其所处板凳所在直线的夹角。最后根据问题一中的思路，采用关联速度的方法，根据前把手的速度得到后把手的速度。结合上述位置迭代公式，我们就将问题一中的速度迭代公式推广到了在整个路径中运动时速度的迭代公式。进而，与问题一的思路类似，通过速度的迭代公式，本文得到各个时刻，每个前把手中心的速度。

2.5 问题五的分析

在舞龙队调头过程中，龙头前把手行进速度始终保持不变，但其余各把手的速度会随着位置的变化而发生变化。经过分析问题四的结果发现，在龙头板凳由第二段调头圆弧进入盘出螺线的过程中，第三块板凳的前把手中心会在运动过程中速度达到最大值。

本文通过分析第四问结果确定了第三块板凳的前把手中心速度达到最大值的大致时间区间，进而计算出此时龙头前把手大致位置区间。再采用十分法，利用编程求解第三块板凳的前把手中心速度达到最大值的精确时间，以及此时龙头前把手精确位置。进而利用问题四中速度的迭代公式，可以得到在这个位置时龙头前把手中心速度与第三块板凳的前把手中心速度的关系，再通过限制第三块板凳的前把手中心速度不超过2，进而求得龙头最大行进速度。

三 模型准备

3.1 模型假设

1. 在舞龙队盘入前，已经以相同的螺距排列为等距螺线列队在盘入螺线外。
2. 忽略板凳厚度带来的影响。
3. 各把手中心严格位于螺线上。
4. 忽略摩擦力带来的影响。

3.2 符号说明

所使用的符号及说明如表3.1所示。

表 3.1: 符号说明

符号	说明	单位
d	螺距	m
ρ	极径	m
θ	极角	rad
v	龙头前把手行进速度	m/s
d_1	把手中心到最近板头的距离	m
d_2	半板宽	m
l_i	第i块板凳前后把手中心所在直线	
l	板凳前后把手中心距离	m
A_1	龙头前外部角点	
A_2	龙头后外部角点	
A_3	第一节龙身前外部角点	
A_4	第一节龙身后外部角点	
dij	角点A_j与直线l_i的距离	m
O_1	第一段圆弧圆心	
O_2	第二段圆弧圆心	
ϕ	第一段圆弧所对圆心角	rad
D	调头空间直径	m

注意：其他符号已在文章的相应部分给出说明

四　模型的建立与求解

4.1　问题一模型的建立与求解

4.1.1　模型的建立

STEP1　龙头前把手中心位置的确定

由已知的螺距$d = 0.55$m和起点A(8.8,0)，本文能够建立盘入螺线所对应的极坐标方程：

$$\rho(\theta) = \frac{d}{2\pi}\theta$$

由于龙头前把手沿盘入螺线的行进速度恒定为$v = 1m/s$，从初始时刻$t = 0$s至$t = t_0 \in [0, 300]$时刻，龙头前把手走过的路径长度为vt_0，极角的角度从$\theta = 32\pi$变为$\theta = \theta_0$。根据螺线长度积分公式，能够得到以下关系式：

$$vt_0 = \int_{\theta_0}^{32\pi} \sqrt{(\rho'(\theta))^2 + \rho^2(\theta)}\mathrm{d}\theta$$

通过已知的t_0，可以解得对应的θ_0，进而通过极坐标与直角坐标的转化公式：

$$\begin{cases} x_0 = \rho(\theta_0)\cos\theta_0 \\ y_0 = \rho(\theta_0)\sin\theta_0 \end{cases}$$

得到$t = t_0$时刻龙头前把手中心坐标(x_0, y_0)。

STEP2　建立位置迭代公式

在得到了龙头前把手中心每个时刻的坐标后，本文建立了一个位置迭代公式以计算各个把手在每个时刻的坐标，建立过程如下：

假设在某一时刻下，某一板凳，其前后把手中心距离为l，其前把手中心位置坐标为(x_1, y_1)，极角为$\theta = \theta_1$，极径为$\rho = \rho_1$。假设该板凳后把手中心位置此时的坐标为(x_2, y_2)，极角为$\theta = \theta_2$，根据螺线方程，其极径满足方程：

$$\rho_2 = \rho_1 + \frac{d}{2\pi}(\theta_2 - \theta_1) \qquad (1)$$

如图4.8所示的三角形中，通过余弦定理可得到如下公式：

$$\rho_1^2 + \rho_2^2 - 2\rho_1\rho_2\cos(\theta_2 - \theta_1) = l^2 \qquad (2)$$

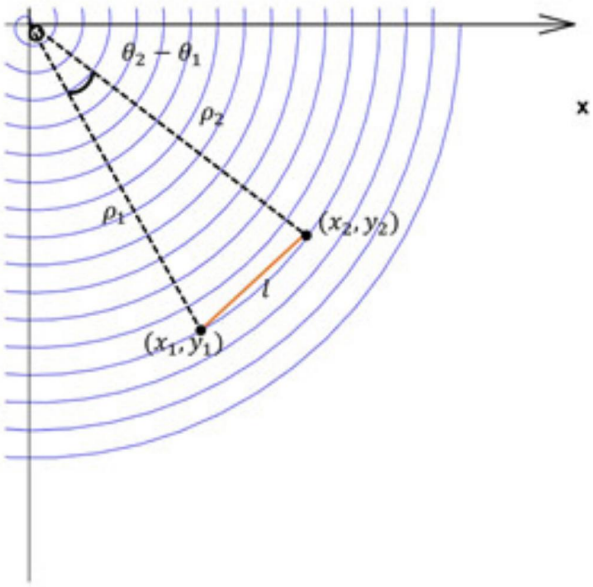

图 4.8: 前后把手相对位置示意图

联立式(1)、式(2)，采用二分求零点的方法可解得ρ_2与θ_2，进而通过坐标转化公式：

$$\begin{cases} x_2 = \rho_2\cos\theta_2 \\ y_2 = \rho_2\sin\theta_2 \end{cases}$$

得到该板凳后把手中心位置的坐标(x_2, y_2)。

通过STEP1中各个时刻龙头前把手中心坐标，利用该迭代公式，本文能够求解出各个时刻整个舞龙队各把手的位置。

STEP3 建立速度迭代公式

不难发现，在舞龙队盘入等距螺线的过程中，同一板凳上前后把手中心沿板凳前后把手中心所在直线的速度是一样的。本文借助这一特性，利用前后把手关于板凳的关联速度建立速度迭代公式，建立过程如下：

假设在某一时刻下，如图4.9所示：

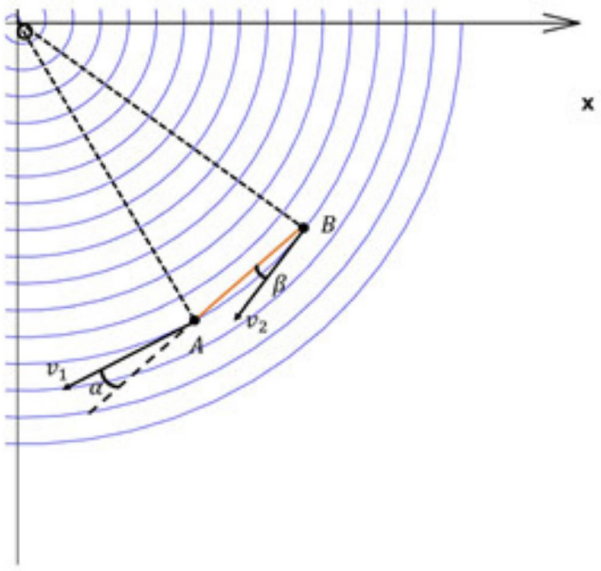

图 4.9: 前后把手速度示意图

某一板凳，已知其前把手中心位置坐标为$A(x_1, y_1)$，行进速度为v_1，极角为θ_1，螺线在该点的切线斜率为k_1。由螺线极坐标方程，可以得到k_1为：

$$k_1 = \frac{\sin\theta_1 + \theta_1\cos\theta_1}{\cos\theta_1 - \theta_1\sin\theta_1}$$

由上述位置迭代公式可以得到后把手中心B的坐标，记为(x_2, y_2)，设其行进速度为v_2。由螺线极坐标方程，可以得到其极角为θ_2，设螺线在该点的切线斜率为k_2。

则k_2为：

$$k_2 = \frac{\sin\theta_2 + \theta_2\cos\theta_2}{\cos\theta_2 - \theta_2\sin\theta_2}$$

接下来，分别设螺线在点A、点B处的切线与A、B所在直线的夹角分别为α、β。

由点A、点B的坐标可以得到A、B所在直线的斜率k为：

$$k = \frac{y_1 - y_2}{x_1 - x_2}$$

从而可以得到α、β为：

$$\alpha = \arctan\left|\frac{k_1 - k}{1 + kk_1}\right|$$

$$\beta = \arctan\left|\frac{k_2 - k}{1 + kk_2}\right|$$

最后，利用前后把手关于板凳的关联速度建立速度迭代公式如下：

$$v_1\cos\alpha = v_2\cos\beta$$

通过已经求得的各个时刻整个舞龙队各把手的位置，以及龙头前把手的恒定速度，可以得到各个时刻整个舞龙队各把手的速度。

4.1.2 模型计算结果

将数据代入位置、速度迭代公式，通过程序得到结果如下：

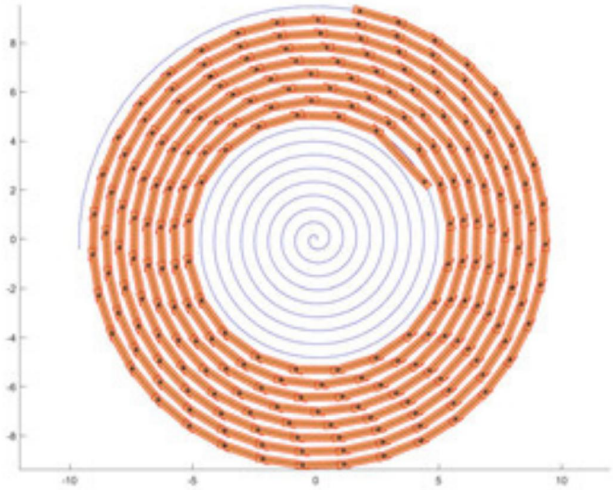

图 4.10: $t = 300$s时舞龙队位置示意图

表 4.2: 位置结果

	0s	60s	120s	180s	240s	300s
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	-5.543149	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	1.898795	-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	6.682312	2.388757	1.005154	2.965378	7.040740
第151节龙身y (m)	1.828753	8.134544	9.727411	9.424751	8.399721	4.393013
第201节龙身x (m)	4.555102	-6.619664	-10.627210	-9.287720	-7.457151	-7.458662
第201节龙身y (m)	10.725118	9.025570	1.359848	-4.246673	-6.180726	-5.263384
龙尾（后）x (m)	-5.305444	7.364557	10.974348	7.383895	3.241051	1.785033
龙尾（后）y (m)	-10.676584	-8.797992	0.843473	7.492371	9.469336	9.301164

表 4.3: 速度结果

	0s	60s	120s	180s	240s	300s
龙头 (m/s)	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
第1节龙身 (m/s)	0.999971	0.999961	0.999945	0.999917	0.999859	0.999709
第51节龙身 (m/s)	0.999742	0.999662	0.999538	0.999331	0.998941	0.998065
第101节龙身 (m/s)	0.999575	0.999453	0.999269	0.998971	0.998435	0.997302
第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

4.2 问题二模型的建立与求解

4.2.1 模型的建立

STEP1 龙头与第一节龙身外部四个角点位置的确定

对任一块板凳，记把手中心距最近的板头距离为d_1，半板宽为d_2，前后把手中心连线所在直线为l_1，板外侧边所在直线为a_3，前、后把手中心与最近的外部角点连线所在直线分别为a_2、a_4，由对称性，l_1与a_2、a_4的夹角均为γ，如图4.11所示：

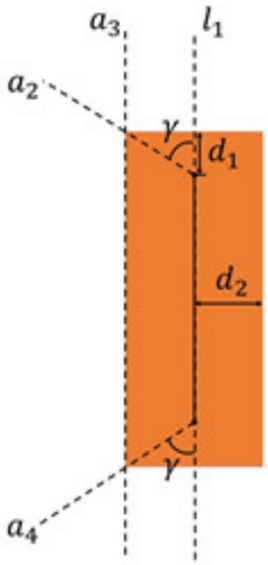

图 4.11: 板凳示意图

龙头前外部角点A_1位置的确定如下：

假设t时刻下龙头前把手中心位置坐标为(x_1, y_1)，极角为$\theta = \theta_1$。根据问题一的位置迭代公式，求龙头后把手中心位置坐标为(x_2, y_2)，进而求得龙头两把手中心所在直线l_1的解析式:

$$l_1 : y - y_1 = k_1(x - x_1) \qquad \text{其中} k_1 = \frac{y_1 - y_2}{x_1 - x_2}$$

由于a_2是由l_1绕前把手中心逆时针旋转γ得到的，根据直线旋转角公式，得到a_2的解析式：

$$a_2: y - y_1 = k_2(x - x_1) \qquad \text{其中} k_2 = \frac{\tan\gamma + k_1}{k_1 \tan\gamma - 1}, \tan\gamma = \frac{d_2}{d_1}$$

由于a_3是由l_1向远离中心原点方向平移d_2得到的，由此得到a_3的解析式：

$$a_3: y = k_1 x + b \qquad \text{其中} b \text{满足条件：} \frac{|b - y_1 + k_1 x_1|}{\sqrt{1 + k_1^2}} = d_2 \text{与} |b| > |y_1 - k_1 x_2|$$

联立a_2与a_3，即可解得A_1的坐标：

$$(x_{A_1}, y_{A_1}) = (\frac{y_1 - k_2 x_1 - b}{k_1 - k_2}, \frac{k_1 y_1 - k_1 k_2 x_1 - k_2 b}{k_1 - k_2})$$

类似的，能够确定t时刻下龙头后外部角点A_2、第一节龙身前外部角点A_3、第一节龙身后外部角点A_4的坐标。

STEP2　四个角点碰撞情况的判断

在STEP1中，本文利用t时刻下确定的龙头前把手坐标(x_1, y_1)，解得四个角点A_1、A_2、A_3、A_4的坐标。

同时，可以通过问题一所得的结果，得到t时刻下极角$\theta \in \left[\theta_1 + \frac{3\pi}{2}, \theta_1 + \frac{5\pi}{2}\right]$的各前把手中心坐标$(x_i, y_i)$（$i$即为该前把手属于第$i$块板凳，龙头为第1块板凳），记满足前把手坐标$(x_i, y_i)$落在要求的极角范围内的$i$的集合为$I$。通过两点间直线公式，能够求解出第$i$块板凳前后把手中心连线所在直线$l_i$的解析式:

$$l_i: y - y_i = \frac{y_i - y_{i+1}}{x_i - x_{i+1}}(x - x_i)$$

得到了角点坐标与直线l_i的解析式后，记角点A_j与直线l_i的距离为d_{ij}，即：

$$d_{ij} = \frac{\left|\frac{y_i - y_{i+1}}{x_i - x_{i+1}}(x_{A_1} - x_i) - y_{A_1} + y_i\right|}{\sqrt{(\frac{y_i - y_{i+1}}{x_i - x_{i+1}})^2 + 1}}$$

对于是否发生碰撞，本文给出如下判断准则：

$$\begin{cases} \forall i, j (i \in I, j = 1,2,3,4), d_{ij} > d_2 & \text{则} t \text{ 时刻未发生碰撞} \\ \exists i, j (i \in I, j = 1,2,3,4) \quad \text{使得} \quad d_{ij} \le d_2 & \text{则} t \text{ 时刻发生碰撞} \end{cases}$$

由此能够确定舞龙队盘入的终止时刻。

4.2.2　模型计算结果

代入数据后，本文通过程序得到终止时刻412.473894s，发生碰撞的点为龙头左前角点，位置、速度结果如下：

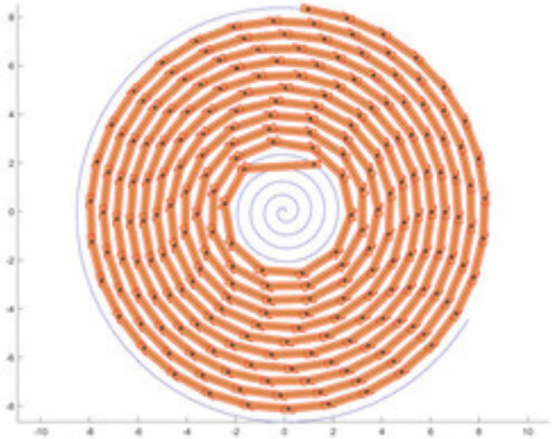

图 4.12: 盘入终止时刻舞龙队位置示意图

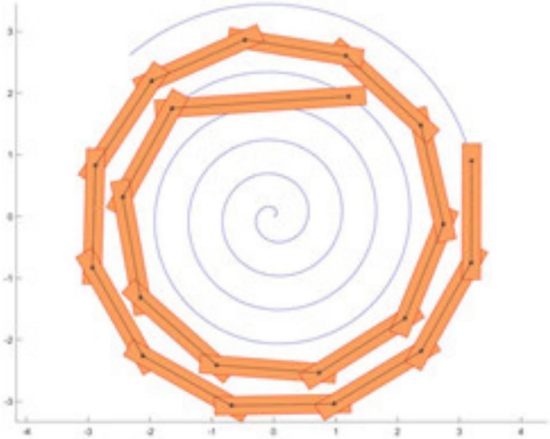

图 4.13: 盘入终止时刻舞龙队位置示意图（放大）

表 4.4: 位置结果

	终止时刻412.473894s
龙头x (m)	1.209931
龙头y (m)	1.942784
第1节龙身x (m)	-1.643792
第1节龙身y (m)	1.753399
第51节龙身x (m)	1.281201
第51节龙身y (m)	4.326588
第101节龙身x (m)	-0.536245
第101节龙身y (m)	-5.880138
第151节龙身x (m)	0.968841
第151节龙身y (m)	-6.957479
第201节龙身x (m)	-7.893161
第201节龙身y (m)	-1.230764
龙尾（后）x (m)	0.956216
龙尾（后）y (m)	8.322736

表 4.5: 速度结果

	终止时刻412.473894s
龙头 (m/s)	1.000000
第1节龙身 (m/s)	0.991551
第51节龙身 (m/s)	0.976858
第101节龙身 (m/s)	0.974550
第151节龙身 (m/s)	0.973608
第201节龙身 (m/s)	0.973096
龙尾（后）(m/s)	0.972938

4.3 问题三模型的建立与求解

4.3.1 模型的建立

通过问题一、问题二中建立的模型，不难发现，在其他条件不变的情况下，每个时刻下舞龙队各把手所处的位置和舞龙队盘入的终止时刻都由螺距大小决定。

因此，本文在问题三中将沿用问题一、问题二的模型和公式，对于任一螺距d，本文建立如下判定流程：

STEP1　对于该螺距d，利用问题二的模型，求解出该螺距下舞龙队的盘入终止时刻。

STEP2　将STEP1中求得的终止时刻与螺距d代入问题一的模型，求解出此时龙头前把手中心的坐标。

STEP3　检查STEP2中得到的龙头前把手中心坐标落在题设要求调头空间的边界上/内部/外部。

本文首先通过调整螺距d得到两个分别使龙头前把手中心落在调头空间内部和外部的螺距，再通过十分法，精确求解使龙头前把手中心恰好位于边界上的螺距。

4.3.2　模型计算结果

代入数据，本文通过程序求解得到能够满足要求的最小螺距为：0.450338（m）

在该最小螺距下，碰撞是由龙头板凳的左后角点造成的。

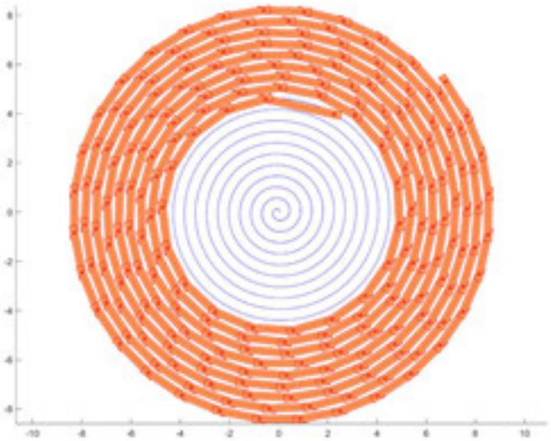

图 4.14: 最小螺距下最接近碰撞时刻示意图

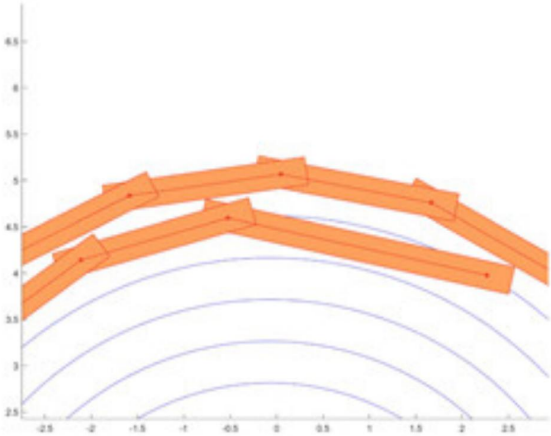

图 4.15: 最小螺距下最接近碰撞时刻示意图（放大）

4.4　问题四模型的建立与求解

4.4.1　模型的建立

首先证明不可以通过调整圆弧使得调头路径更短：

假设两圆弧的半径比为k时，调头路径如图4.16所示：

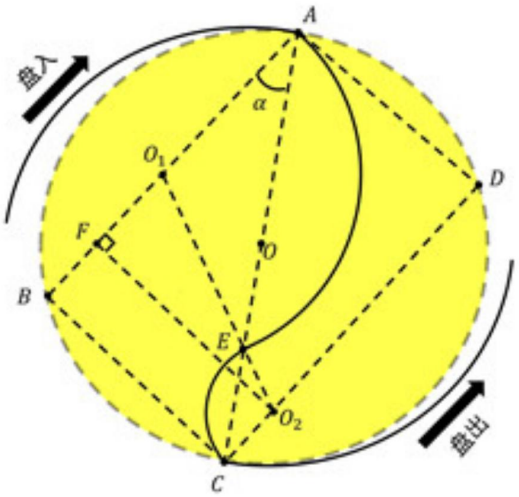

图 4.16: 调头路径示意图

其中四边形$ABCD$为矩形，O_1、O_2分别为两个圆弧对应的圆心，过O_2作O_2F垂直于AB，垂足为F。

因角α为盘入螺线在切入点处切线的垂线与调头空间直径AC的夹角，因此，α与两端圆弧的半径比无关。

设第二段圆弧半径为r，则第一段圆弧半径为kr。两段圆弧所对应的圆心角的角度为$\pi-2\alpha$，则调头路径的长度，即两段圆弧的长度之和为:

$$s=(k+1)r(\pi-2\alpha) \qquad (1)$$

在直角三角形ABC中，AC为调头空间的直径，其长度记为D，则有：

$$AB=D\cos\alpha$$

$$BC=D\sin\alpha$$

从而，在直角三角形O_2FO_1中，有：

$$O_2F=BC=D\sin\alpha$$

$$O_1F=AB-O_1A-BF=D\cos\alpha-(k+1)r$$

$$O_1O_2=(k+1)r$$

利用勾股定理，有：

$$D^2\sin^2\alpha+(D\cos\alpha-(k+1)r)^2=(k+1)^2r^2 \qquad (2)$$

联立公式(1)、(2)，即可解得：

$$s=\frac{D\alpha}{2\cos\alpha}$$

由此发现调头路径的长度s与两段圆弧半径比k无关，因此不可以通过调整圆弧使得调头路径更短。

本文在后续求解过程中设定两段圆弧的半径比为$2:1$。

计算得该比例下部分数据如下：

切入点A坐标为：

$$(x_A,y_A)=(-2.711856,-3.591078)$$

大圆弧圆心0_1坐标为：

$$(x_{O_1},y_{O_1})=(-0.760009,-1.305726)$$

小圆弧圆心O_2坐标为：

$$(x_{O_2},y_{O_2})=(1.735932,2.448402)$$

小圆弧的半径为：

$$r=1.502709(m)$$

两段圆弧交点E坐标为：

$$(x_E, y_E) = (0.903952, 1.197026)$$

切出点C坐标为：

$$(x_C, y_C) = (2.711856, 3.591078)$$

两端圆弧所对应的圆心角为：

$$\phi = 3.021487(rad)$$

随后，在求解调头过程中各个时刻整个舞龙队的位置和速度，有如下步骤：

STEP1　确定龙头前把手中心位置

和问题一的思路类似，由于龙头前把手的行进速度恒定，本文利用四段弧线的方程和积分求解出掉头过程中各个时刻龙头前把手中心的位置。

STEP2　建立根据前把手中心位置判断后把手中心所处弧线的判断函数

假设某一时刻，某一板凳，其前后把手中心距离为l，其前把手中心位置坐标为(x_1, y_1)已知，从而能够得知该点位于哪一段弧线上。有如下情况：

1. 当该点位于盘入螺线时，该板凳后把手中心一定位于盘入螺线上。
2. 当该点位于第一段圆弧时，引入判断角φ，如图4.17所示：

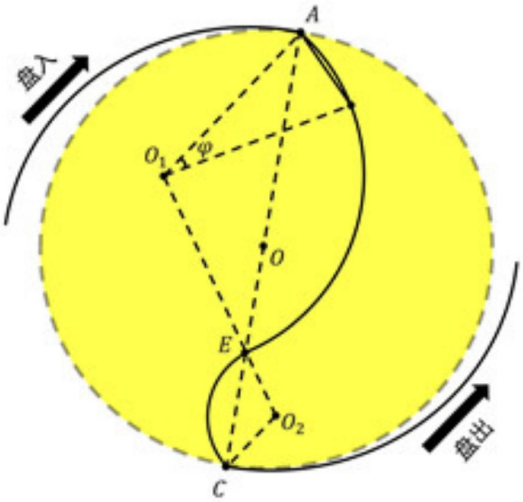

图 4.17: 情况二判断角示意图

其中，φ满足：

$$(2r)^2 + (2r)^2 - 2(2r)^2 \cos\varphi = l^2$$

解得：

$$\varphi = \arccos(\frac{8r^2 - l^2}{8r^2})$$

接下来，设前把手中心与O_1连线的线段与O_1A的夹角为φ'，有：

$$\varphi' = \arctan\left|\frac{\frac{y_A-y_{O_1}}{x_A-x_{O_1}} - \frac{y_1-y_{O_1}}{x_1-x_{O_1}}}{1+\frac{y_A-y_{O_1}}{x_A-x_{O_1}}\frac{y_1-y_{O_1}}{x_1-x_{O_1}}}\right|$$

当$\varphi' > \varphi$时，后把手中心在第一段圆弧上；

当$\varphi' \leq \varphi$时，后把手中心在盘入螺线上。

3. 当该点位于第二段圆弧时，此时的判断角φ如图4.18所示：

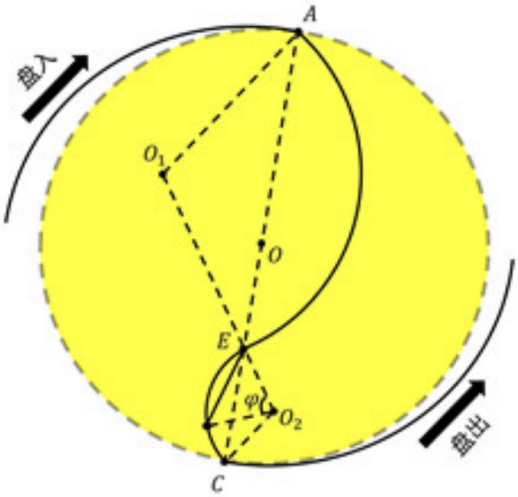

图 4.18: 情况三判断角示意图

其中，φ满足：

$$r^2 + r^2 - 2r^2\cos\varphi = l^2$$

解得：

$$\varphi = \arccos(\frac{2r^2 - l^2}{2r^2})$$

接下来，设前把手中心与O_2连线的线段与O_2E的夹角为φ'，有：

$$\varphi' = \arctan\left|\frac{\frac{y_E-y_{O_2}}{x_E-x_{O_2}} - \frac{y_1-y_{O_2}}{x_1-x_{O_2}}}{1+\frac{y_E-y_{O_2}}{x_E-x_{O_2}}\frac{y_1-y_{O_2}}{x_1-x_{O_2}}}\right|$$

当$\varphi' > \varphi$时，后把手中心在第二段圆弧上；

当$\varphi' \leq \varphi$时，后把手中心在第一段圆弧上。

4. 当该点位于盘出螺线时，此时的判断角φ如图4.19所示：

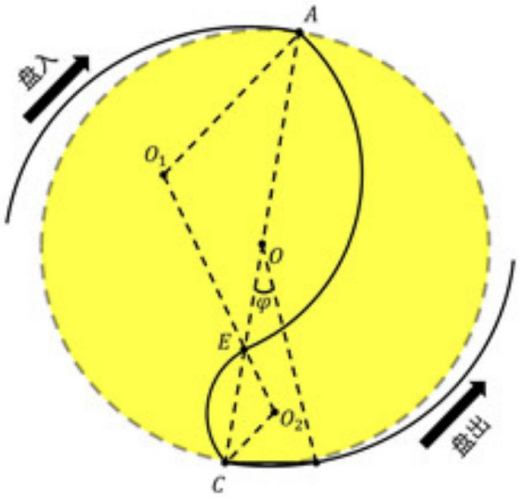

图 4.19: 情况四判断角示意图

其中，φ满足：

$$(\frac{D}{2}+\frac{d}{2\pi}\varphi)^2+(\frac{D}{2})^2-D(\frac{D}{2}+\frac{d}{2\pi})\cos\varphi=l^2$$

本文通过程序解得φ的值。

接下来，设前把手中心与O连线的线段与OC的夹角为φ'，有：

$$\varphi'=\frac{2\pi(\sqrt{x_1^2+y_1^2}-\frac{D}{2})}{d}$$

当$\varphi'>\varphi$时，后把手中心在盘出螺线上；

当$\varphi'\leq\varphi$时，后把手中心在第二段圆弧上。

STEP3　建立位置迭代公式

假设某一时刻，某一板凳，其前后把手中心距离为l，其前把手中心位置坐标为(x_1,y_1)已知，从而能够得知该点位于哪一段弧线上。

有如下情况：

1. 当板凳的前把手中心位于盘入螺线上时，位置迭代公式与问题一相同。
2. 当板凳的前把手中心位于第一段圆弧，后把手中心位于盘入螺线上，如图4.20所示：

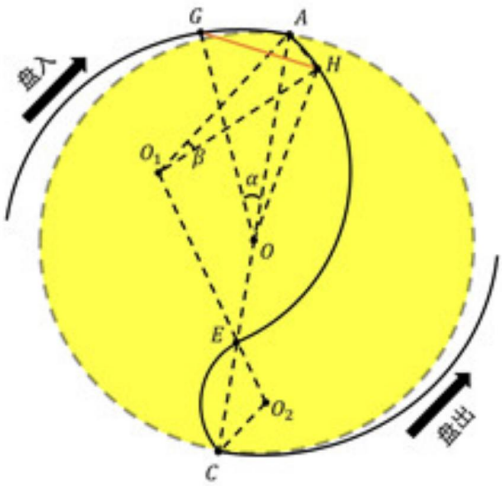

图 4.20: 情况二示意图

结合螺线方程，可知：

$$OG = \frac{D}{2} + \frac{d\alpha}{2\pi}$$

$$OH = \sqrt{x_1^2 + y_1^2}$$

$$\angle GOH = \alpha + \angle AOH$$

在三角形OGH中，有余弦定理：

$$OG^2 + OH^2 - 2OG \cdot OH \cos(\angle HOG) = l^2$$

在三角形AO_1H中，有余弦定理：

$$(2r)^2 + (2r)^2 - 2(2r)^2 cos\beta = AH^2$$

在三角形AOH中，有余弦定理：

$$(\frac{D}{2})^2 + OH^2 - D \cdot OH \cos(\angle AOH) = AH^2$$

联立上述公式，用程序解得α和OG的值，进而由螺线方程得到后把手中心G的坐标：

$$(x_2, y_2) = ((\frac{D}{2} + \frac{d\alpha}{2\pi}) \cos(\frac{D\pi}{d} + \alpha), (\frac{D}{2} + \frac{d\alpha}{2\pi}) \sin(\frac{D\pi}{d} + \alpha))$$

3. 当板凳的前后把手中心均位于第一段圆弧，如图4.21所示：

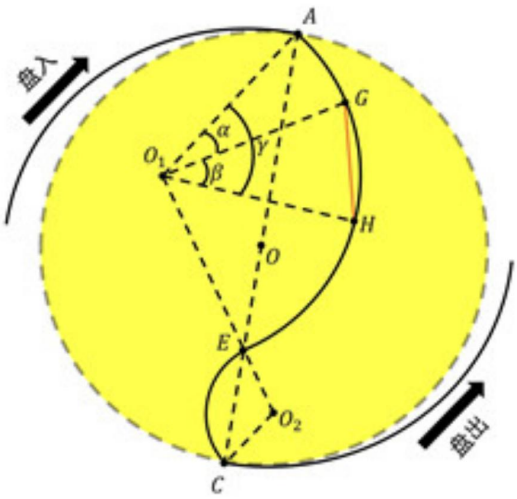

图 4.21: 情况三示意图

其中，β满足：

$$(2r)^2+(2r)^2-2(2r)^2\cos\beta=l^2$$

解得：

$$\beta=\arccos(\frac{8r^2-l^2}{8r^2})$$

又有：

$$\gamma=\arctan(\frac{\frac{y_A-y_{O_1}}{x_A-x_{O_1}}-\frac{y_1-y_{O_1}}{x_1-x_{O_1}}}{1+\frac{y_A-y_{O_1}}{x_A-x_{O_1}}\frac{y_1-y_{O_1}}{x_1-x_{O_1}}})$$

$$\alpha=\gamma-\beta$$

从而能够通过程序解得后把手中心G的坐标：

$$(x_2,y_2)=(x_{O_1}+2r\cos(\arctan(\frac{y_{O_1}-y_A}{x_{O_1}-x_A})+\pi-\alpha),y_{O_1}+2r\sin(\arctan(\frac{y_{O_1}-y_A}{x_{O_1}-x_A})+\pi-\alpha))$$

4. 当板凳的前后把手中心位于第二段圆弧，后把手中心位于第一段圆弧，如图4.22所示：

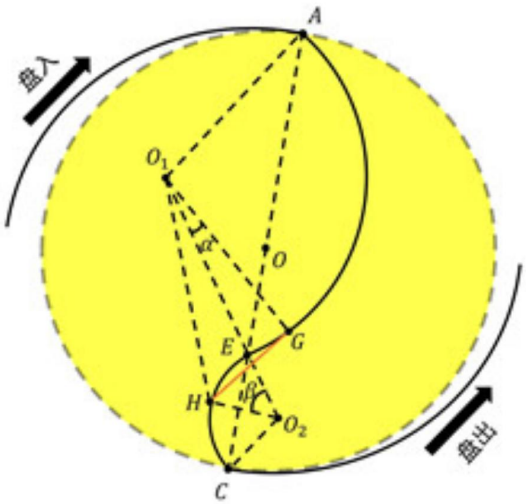

图 4.22: 情况四示意图

其中，角β已知，在三角形O_1HO_2中，由余弦定理、正弦定理有：

$$r^2 + (3r)^2 - 6r\cos\beta = O_1H^2$$

$$\frac{O_1H}{\sin\beta} = \frac{O_2H}{\sin\angle HO_1O_2}$$

在三角形O_1HG中，由余弦定理：

$$O_1H^2 + (2r)^2 - 4r \cdot O_1H\cos(\angle HO_1G) = l^2$$

又：

$$\alpha = \angle HO_1G - \angle HO_1O_2$$

联立上述公式解得：

$$\alpha = \arccos(\frac{14r^2 - 6r\cos\beta - l^2}{4r\sqrt{10r^2 - 6r\cos\beta}}) - \arcsin(\frac{r\sin\beta}{\sqrt{10r^2 - 6r\cos\beta}})$$

从而解得后把手中心G的坐标：

$$(x_2, y_2) = (x_{O_1} + 2r\cos(\arctan(\frac{y_{O_1} - y_A}{x_{O_1} - x_A}) + \pi + \alpha - \phi), y_{O_1} + 2r\sin(\arctan(\frac{y_{O_1} - y_A}{x_{O_1} - x_A}) + \pi + \alpha - \phi))$$

5. 当板凳的前后把手中心均位于第二段圆弧，如图4.23所示：

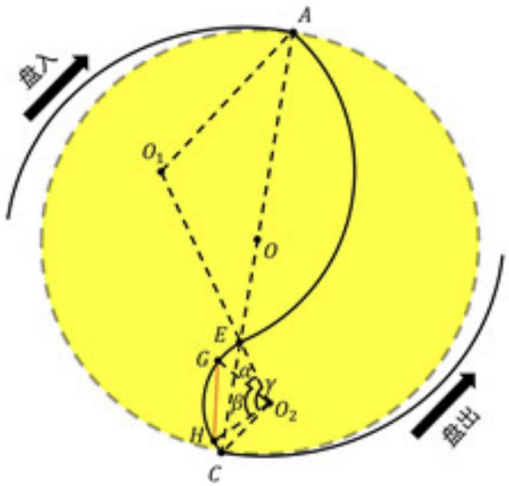

图 4.23: 情况五示意图

其中，角γ已知，在三角形GO_2H中，由余弦定理：

$$r^2 + r^2 - 2r^2 \cos\beta = l^2$$

解得：

$$\beta = \arccos(\frac{2r^2 - l^2}{2r^2})$$

又：

$$\alpha = \gamma - \beta$$

从而能够通过程序解得后把手中心G的坐标：

$$(x_2, y_2) = (x_2, y_2) = (x_{O_2} + r\cos(\arctan(\frac{y_{O_2} + y_A}{x_{O_2} + x_A}) + \alpha - \phi), y_{O_2} + r\sin(\arctan(\frac{y_{O_2} + y_A}{x_{O_2} + x_A}) + \alpha - \phi))$$

6. 当板凳的前后把手中心位于盘出螺线，后把手中心位于第二段圆弧，如图4.24所示：

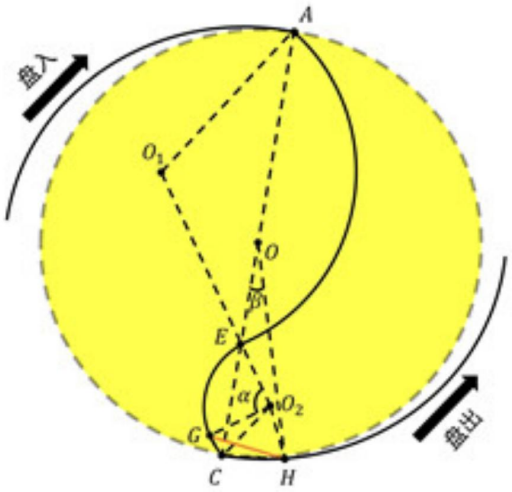

图 4.24: 情况六示意图

由螺线方程，有：

$$OH = \frac{D}{2} + \frac{d\beta}{2\pi}$$

在三角形OCH中，由余弦、正弦定理：

$$\frac{D^2}{4} + (\frac{D}{2} + \frac{d\beta}{2\pi})^2 - D(\frac{D}{2} + \frac{d\beta}{2\pi})\cos\beta = CH^2$$

$$\frac{\sin\beta}{CH} = \frac{\sin(\angle OCH)}{OH}$$

从而得到：

$$\angle O_2CH = \angle OCH - \frac{(\pi - \phi)}{2}$$

在三角形O_2CH中，由余弦、正弦定理：

$$r^2 + CH^2 - 2r \cdot CH\cos(\angle O_2CH) = O_2H^2$$

$$\frac{\sin(\angle O_2HC)}{r} = \frac{\sin(\angle O_2CH)}{O_2H} = \frac{\sin(\angle CO_2H)}{CH}$$

在三角形O_2GH中，由余弦定理：

$$r^2 + O_2H^2 - 2r \cdot O_2H\cos(\angle GO_2H) = l^2$$

从而：

$$\angle GO_2C = \angle GO_2H - \angle CO_2H$$

$$\alpha = \phi - \angle GO_2C$$

利用程序解得α的值，即可解得后把手中心G的坐标：

$$(x_2,y_2)=(x_2,y_2)=(x_{O_2}+r\cos(\arctan(\frac{y_{O_2}+y_A}{x_{O_2}+x_A})+\alpha-\phi),y_{O_2}+r\sin(\arctan(\frac{y_{O_2}+y_A}{x_{O_2}+x_A})+\alpha-\phi))$$

7. 当板凳的前后把手中心均位于盘出螺线上时，由于盘入螺线与盘出螺线关于螺线中心呈中心对称，故可采用类似于情况一的方法求解后把手中心的坐标。

STEP4　建立速度迭代公式

根据上述位置迭代公式，对于一块板凳，可以解得任一时刻其前后把手中心的坐标，从而有前后把手中心所在直线的斜率。因此，要将前后把手的速度关联起来，只需要确定前后把手分别的速度方向，即前后把手中心的切线方向。

同STEP3一样，有7种情况，由于已经通过程序求解出两段圆弧的圆心O_1、O_2的坐标，并且已知盘入、盘出螺线的方程，故不论前后把手中心位于哪一段弧线上，都能求解出其速度方向，即在该弧线上的切线方向。从而结合前后把手中心所在直线的斜率、前后把手沿中心所在直线的速度相同，建立起前后把手速度的关系式，也就是速度迭代公式。

取定一点$I(x,y)$，本文分四种情况讨论该点处切线的斜率：

1. 当I位于盘入螺线时，可由极坐标方程计算其极角θ，从而有切线斜率：

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

2. 当I位于第一段圆弧时，可计算切线斜率为：

$$k=-\frac{x-x_{O_1}}{y-y_{O_1}}$$

3. 当I位于第二段圆弧时，可计算切线斜率为：

$$k=-\frac{x-x_{O_2}}{y-y_{O_2}}$$

4. 当I位于盘出螺线时，可由极坐标方程计算其极角θ，从而有切线斜率：

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

本文以前把手中心在第一段圆弧上，后把手中心在盘入螺线上的情况为例：

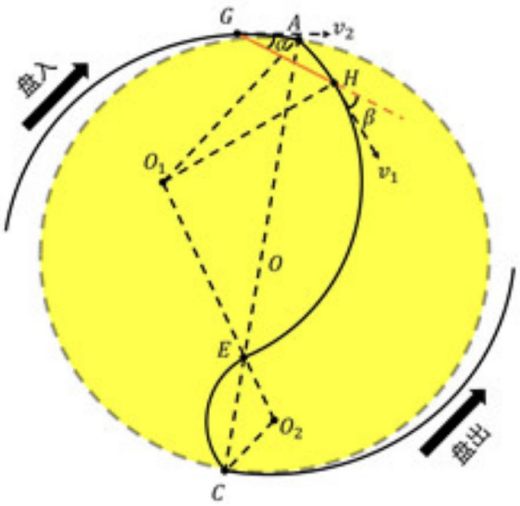

图 4.25: 速度迭代示意图

已知前把手中心H坐标为(x_1, y_1)，根据上述位置迭代公式可以得到后把手中心G的坐标，记为(x_2, y_2)，由盘入螺线方程可得极角为θ_2。

从而有前后把手中心所在直线的斜率：

$$k = \frac{y_1 - y_2}{x_1 - x_2}$$

前把手中心的切线斜率：

$$k_1 = -\frac{x_1 - x_{O_1}}{y_1 - y_{O_1}}$$

后把手中心的切线斜率：

$$k_2 = \frac{\sin\theta_2 + \theta_2\cos\theta_2}{\cos\theta_2 - \theta_2\sin\theta_2}$$

可以得到α、β为：

$$\alpha = \arctan\left|\frac{k_2 - k}{1 + kk_2}\right|$$

$$\beta = \arctan\left|\frac{k_1 - k}{1 + kk_1}\right|$$

从而建立起前后把手速度v_1、v_2的关系式：

$$v_1 \cos\beta = v_2 \cos\alpha$$

对于其他情况，本文使用相应的方法得到了相应的速度迭代公式。

4.4.2 模型计算结果

代入数据，通过程序得到结果如下：

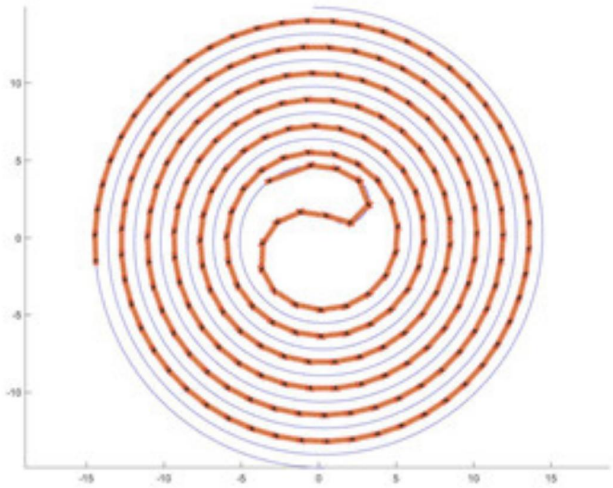

图 4.26: $t = 20$s时舞龙队位置示意图

表 4.6: 位置结果

	-100s	-50s	0s	50s	100s
龙头x (m)	7.778034	6.608301	-2.711855	1.332696	-3.157228
龙头y (m)	3.717164	1.898865	-3.591077	6.175324	7.548511
第1节龙身x (m)	6.209273	5.366911	-0.063533	3.862265	-0.346889
第1节龙身y (m)	6.108521	4.475403	-4.670887	4.840828	8.079166
第51节龙身x (m)	-10.608037	-3.629945	2.459962	-1.665659	2.095033
第51节龙身y (m)	2.831492	-8.963799	-7.778145	-6.078552	4.033787
第101节龙身x (m)	-11.922761	10.125787	3.008493	-7.595340	-7.288774
第101节龙身y (m)	-4.802377	-5.972246	10.108539	5.170626	2.063875
第151节龙身x (m)	-14.351032	12.974784	-7.002788	-4.599737	9.462514
第151节龙身y (m)	-1.980992	-3.810357	10.337482	-10.389549	-3.540356
第201节龙身x (m)	-11.952942	10.522508	-6.872841	0.342952	8.524374
第201节龙身y (m)	10.566998	-10.807425	12.382609	-13.177577	8.606933
龙尾（后）x (m)	-1.011058	0.189810	-1.933627	5.853703	-10.980157
龙尾（后）y (m)	-16.527572	15.720588	-14.713128	12.615526	-6.770006

表 4.7: 速度结果

	-100s	-50s	0s	50s	240s
龙头 (m/s)	1.000000	1.000000	1.000000	1.000000	1.000000
第1节龙身 (m/s)	0.999904	0.999762	0.998686	1.000363	1.000124
第51节龙身 (m/s)	0.999346	0.998641	0.995134	0.949698	1.003966
第101节龙身 (m/s)	0.999091	0.998248	0.994448	0.948246	1.096263
第151节龙身 (m/s)	0.998944	0.998047	0.994156	0.947802	1.095306
第201节龙身 (m/s)	0.998849	0.997925	0.993994	0.947587	1.094934
龙尾（后）(m/s)	0.998817	0.997885	0.993944	0.947524	1.094833

4.5 问题五模型的建立与求解

4.5.1 模型的建立

在舞龙队调头过程中，龙头前把手行进速度始终保持不变，但其余各把手的速度会随着位置的变化而发生变化。本文先将龙头前把手速度设为恒定1m/s。通过问题四的模型，我们得到如下图：

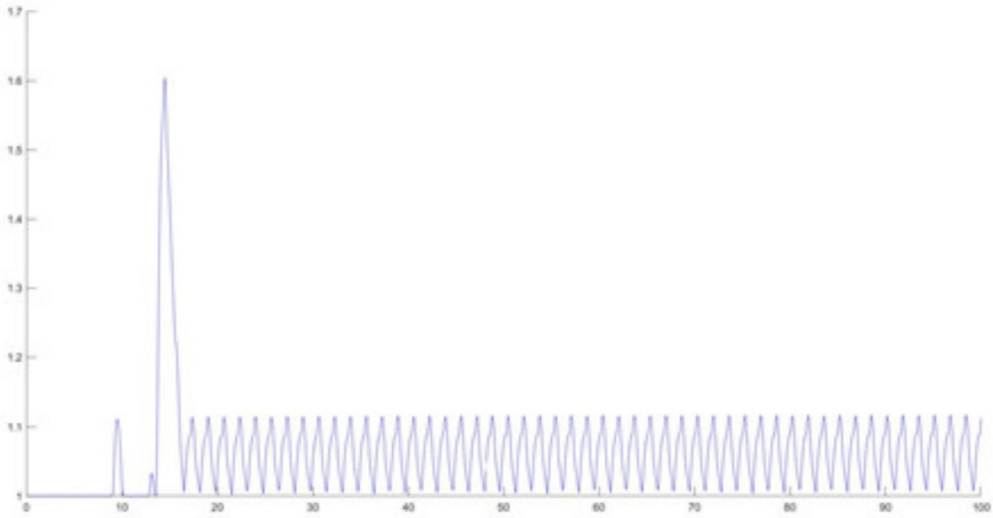

图 4.27: 0 − 100s内舞龙队所有把手最大速度随时间变化示意图

经过分析上图发现，在龙头前把手进入盘出螺线到龙头后把手进入盘出螺线的过程中，第一至八节龙身各把手的速度会明显增大，但在第九节及以后速度递减。

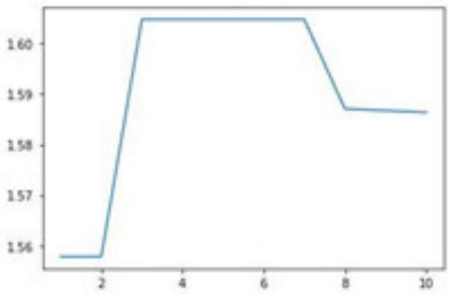

图 4.28: 龙头进入盘出螺线的过程中，各节龙身前把手的最大速度

分析图4.28发现，第三至第七节龙身前把手的速度在此过程中达到最大，因此本文选定第三节龙身的前把手为目标把手，分析在龙头由第二段圆弧至盘出螺线的过程中，该把手的速度变化。

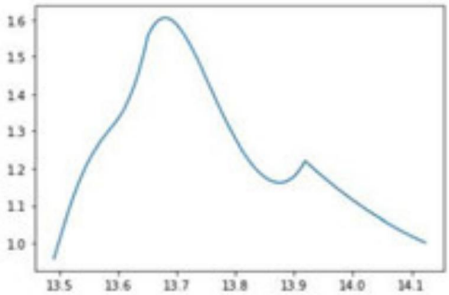

图 4.29: 龙头进入盘出螺线的过程中，第三节龙身前把手的速度

程序求解得到，这段过程中第三节龙身前把手的最大速度为1.604793m/s。

由问题四中建立的速度迭代公式可知，在第三节龙身前把手达到最大速度的位置，其速度与龙头前把手的速度成正比。因此可通过限制第三节龙身前把手的最大速度为2m/s，来得到龙头的最大行进速度。

4.5.2 模型计算结果

代入数据，求解得龙头的最大行进速度为1.246266m/s。

五 模型的评价

5.1 模型的优点

- 优点1：模型没有过多的简省，较为精确地给出问题所需要的位置和速度数据，较为准确地解决了问题。
- 优点2：模型的主体思路是构造位置迭代公式与速度迭代公式，也适用于不同运动路径的情况，可以进行推广。

5.2 模型的缺点

- 缺点1：计算量大，计算复杂度较高。

参考文献

[1] 姜启源，《数学模型》，高等教育出版社，2007年.

附录

支撑材料的文件列表：

1. result1.xlsx
2. result2.xlsx
3. result4.xlsx
4. function.py
5. zeropoint.py
6. number.py
7. 问题一代码.py
8. crushjudge.py
9. 问题二代码.py
10. 问题三代码.py
11. dataproduce.py
12. positioniteration.py
13. velocityiteration.py
14. 问题四代码.py
15. velctorytheta.py
16. 问题五代码.py

代码：

1. function

```
import numpy as np
def f1(theta):
    result=theta*np.sqrt(theta**2+1)+np.log(theta+np.sqrt(theta**2+
                                                1))
    return result
#用于计算龙头位置随时间的变化
def f2(theta,theta0,v0,t,d):
    result=f1(theta0)-f1(theta)-4*v0*t*np.pi/d
    return result
#用于计算龙头位置随时间的变化
def f3(theta,d,d0,theta_last):
    t=theta
    t_l=theta_last
```

```
    result=t**2+t_l**2-2*t*t_l*np.cos(t-t_l)-4*np.pi**2*d0**2/d**2
    return result
#用于计算盘入螺线上的位置迭代
def f4(theta):
    t=theta
    result=(np.sin(t)+t*np.cos(t))/(np.cos(t)-t*np.sin(t))
    return result
#用于计算螺线上速度的斜率
def f5(theta,d,d0,theta_last):
    t=theta+np.pi
    t_l=theta_last+np.pi
    result=t**2+t_l**2-2*t*t_l*np.cos(t-t_l)-4*np.pi**2*d0**2/d**2
    return result
#用于计算盘出螺线上的位置迭代
def f6(theta,d,d0,theta0,l,gamma):
    t=theta
    t0=theta0
    result=l**2+d**2*t**2/(4*np.pi**2)-d*l*t*np.cos(t-t0+gamma)/np.
                                                pi-d0**2
    return result
#用于计算前把手在第一段圆弧而后把手在盘入螺线的情形

import numpy as np
def f1(theta):
    result=theta*np.sqrt(theta**2+1)+np.log(theta+np.sqrt(theta**2+
                                                1))
    return result
#用于计算龙头位置随时间的变化
def f2(theta,theta0,v0,t,d):
    result=f1(theta0)-f1(theta)-4*v0*t*np.pi/d
    return result
#用于计算龙头位置随时间的变化
def f3(theta,d,d0,theta_last):
    t=theta
    t_l=theta_last
    result=t**2+t_l**2-2*t*t_l*np.cos(t-t_l)-4*np.pi**2*d0**2/d**2
    return result
#用于计算盘入螺线上的位置迭代
def f4(theta):
    t=theta
    result=(np.sin(t)+t*np.cos(t))/(np.cos(t)-t*np.sin(t))
    return result
#用于计算螺线上速度的斜率
def f5(theta,d,d0,theta_last):
    t=theta+np.pi
    t_l=theta_last+np.pi
    result=t**2+t_l**2-2*t*t_l*np.cos(t-t_l)-4*np.pi**2*d0**2/d**2
    return result
#用于计算盘出螺线上的位置迭代
def f6(theta,d,d0,theta0,l,gamma):
    t=theta
    t0=theta0
    result=l**2+d**2*t**2/(4*np.pi**2)-d*l*t*np.cos(t-t0+gamma)/np.
                                                pi-d0**2
    return result
#用于计算前把手在第一段圆弧而后把手在盘入螺线的情形
```

2. zeropoint

```
    def zero1(f,a,b,e,theta0,v0,t,d):
    while b-a>=e:
        c=(a+b)/2 #取中点
        if f(a,theta0,v0,t,d)*f(c,theta0,v0,t,d)<0:
            b=c
        else:
            a=c
        #判断零点所在区间
    return (a+b)/2
#用于计算函数f2的零点
def zero2(f,a,b,e,d,d0,theta_last):
    while b-a>=e:
        c=(a+b)/2 #取中点
        if f(a,d,d0,theta_last)*f(c,d,d0,theta_last)<0:
            b=c
        else:
            a=c
        #判断零点所在区间
    return (a+b)/2
#用于计算函数f3和f5的零点
def zero3(f,a,b,e,d,d0,theta0,l,gamma):
    while b-a>=e:
        c=(a+b)/2 #取中点
        if f(a,d,d0,theta0,l,gamma)*f(c,d,d0,theta0,l,gamma)<0:
            b=c
        else:
            a=c
        #判断零点所在区间
    return (a+b)/2
#用于计算函数f6的零点
```

3. number

```
        import numpy as np
def number(A,n):
    for i in np.arange(A.shape[0]):
        for j in np.arange(A.shape[1]):
            a=A[i,j]
            b=int(a*10**n)*10**(-n)
            #获得a的前n位小数
            if a-b>=5*10**(-n-1):
                b=b+10**(-n)
            #四舍五入
            A[i,j]=b
    return A
#用于保留6位小数
```

4. 问题一代码

```
        import numpy as np
import pandas as pd
from function import f2 #用于计算龙头位置随时间的变化
from function import f3 #用于计算盘入螺线上的位置迭代
from function import f4 #用于计算螺线上速度的斜率
from zeropoint import zero1 #用于计算函数f2的零点
from zeropoint import zero2 #用于计算函数f3的零点
from number import number #用于保留6位小数
```

```
d=0.55 #螺距
v0=1 #龙头速度
theta0=32*np.pi #龙头初始极角
lst_chair_theta=[]
for t in np.arange(301):
    if t==0:
        theta_chair0=theta0
    else:
        theta_chair0=zero1(f2,0,theta0,10**(-8),theta0,v0,t,d)
    #给出龙头的t极角theta
    lst_theta=[theta_chair0]
    for i in np.arange(223):
        if i==0:
            d0=3.41-0.275*2
        else:
            d0=2.2-0.275*2
        #确定板长
        theta_last=lst_theta[-1] #获得上一个把手的极角theta
        theta=zero2(f3,theta_last,theta_last+np.pi/2,10**(-8),d,d0,
                                                    theta_last)
        lst_theta.append(theta)
        #计算当前把手的极角theta
    lst_chair_theta.append(lst_theta)
lst_chair_theta=np.array(lst_chair_theta)
lst_chair_xy=[]
for t in np.arange(301):
    lst_xy=[]
    for i in np.arange(224):
        theta=lst_chair_theta[t,i]
        lst_xy.append(d*theta*np.cos(theta)/(2*np.pi))
        lst_xy.append(d*theta*np.sin(theta)/(2*np.pi))
        #根据theta角度确定把手坐标 (x, y)
    lst_chair_xy.append(lst_xy)
lst_chair_xy=np.array(lst_chair_xy).T
lst_chair_xy=number(lst_chair_xy,6) #保留6位小数
df=pd.DataFrame(lst_chair_xy)
df.to_excel("result1_1.xlsx",index=False)
#保存数据到Excel中
lst_chair_v=[]
for t in np.arange(0,301):
    lst_v=[v0]
    for i in np.arange(223):
        v_last=lst_v[-1] #获得上一个把手的速度
        theta_last=lst_chair_theta[t,i]
        theta=lst_chair_theta[t,i+1]
        x_last=lst_chair_xy[i*2,t]
        y_last=lst_chair_xy[i*2+1,t]
        x=lst_chair_xy[i*2+2,t]
        y=lst_chair_xy[i*2+3,t]
        #获得上一个把手和当前把手的坐标 (x, y) 和极角
        k_chair=(y_last-y)/(x_last-x)
        k_v_last=f4(theta_last)
        k_v=f4(theta)
        #计算板凳和两个速度的斜率
        aleph1=np.arctan(np.abs((k_v_last-k_chair)/(1+k_v_last*
                                                    k_chair)))
        aleph2=np.arctan(np.abs((k_v-k_chair)/(1+k_v*k_chair)))
        #计算两个速度与板凳的夹角
```

```
        v=v_last*np.cos(aleph1)/np.cos(aleph2) #计算当前把手的速度
        lst_v.append(v)
    lst_chair_v.append(lst_v)
lst_chair_v=np.array(lst_chair_v).T
lst_chair_v=number(lst_chair_v,6) #保留6位小数
df=pd.DataFrame(lst_chair_v)
df.to_excel("result1_2.xlsx",index=False)
#保存数据到Excel中
```

5. crushjudge

```
        import numpy as np
from function import f3 #用于计算盘入螺线上的位置迭代
from zeropoint import zero2 #用于计算函数f3的零点
def judge(theta,d,v0):
    d1=0.275
    d2=0.15
    lst_theta=[theta]
    for i in np.arange(223):
        if i==0:
            d0=3.41-0.275*2
        else:
            d0=2.2-0.275*2
        #确定板长
        theta_last=lst_theta[-1] #获得上一个把手的极角theta
        a=theta_last
        b=theta_last+np.pi/2
        theta_new=zero2(f3,a,b,10**(-8),d,d0,theta_last)
        lst_theta.append(theta_new)
        #计算当前把手的极角theta
        if theta_new-theta>=3*np.pi:
            break
        #当当前把手位置离龙头过远时结束
    lst_x=[]
    lst_y=[]
    for i in np.arange(len(lst_theta)):
        p=lst_theta[i]*d/(2*np.pi)
        x=p*np.cos(lst_theta[i])
        y=p*np.sin(lst_theta[i])
        lst_x.append(x)
        lst_y.append(y)
        #根据theta角度确定把手坐标 (x, y)
    lst_k=[]
    for i in np.arange(len(lst_theta)-1):
        k=(lst_y[i]-lst_y[i+1])/(lst_x[i]-lst_x[i+1])
        lst_k.append(k)
        #计算板凳的斜率
    k1=lst_k[0]
    x1=lst_x[0]
    y1=lst_y[0]
    #获得龙头的坐标和斜率
    k2=(d2/d1+k1)/(1-d2*k1/d1) #计算龙头前把手和外前点直线的斜率
    b=d2*np.sqrt(k1**2+1)+y1-k1*x1
    if np.abs(b)<=np.abs(y1-k1*x1):
        b=-d2*np.sqrt(k1**2+1)+y1-k1*x1
    #计算龙头外侧边的截距
    x=(y1-k2*x1-b)/(k1-k2)
    y=(k1*y1-k1*k2*x1-k2*b)/(k1-k2)
```

```
#计算龙头外前点的坐标
flag=0
for i in np.arange(len(lst_k)):
    if lst_theta[i+1]-theta>=np.pi:
        ki=lst_k[i]
        xi=lst_x[i]
        yi=lst_y[i]
        d_chair=np.abs(ki*(x-xi)+yi-y)/np.sqrt(ki**2+1)
        #计算龙头外前点到当前板凳中心线的距离
        if d_chair<d2:
            flag=1
        #判断是否相撞
x2=lst_x[1]
y2=lst_y[1]
#获得第一节龙身前把手的坐标
k2=(k1-d2/d1)/(1+d2*k1/d1) #计算第一节龙身前把手和龙头外后点直线的斜率
x=(y2-k2*x2-b)/(k1-k2)
y=(k1*y2-k1*k2*x2-b*k2)/(k1-k2)
#计算龙头外后点的坐标
for i in np.arange(len(lst_k)):
    if lst_theta[i+1]-theta>=np.pi:
        ki=lst_k[i]
        xi=lst_x[i]
        yi=lst_y[i]
        d_chair=np.abs(ki*(x-xi)+yi-y)/np.sqrt(ki**2+1)
        #计算龙头外后点到当前板凳中心线的距离
        if d_chair<d2:
            flag=1
        #判断是否相撞
k1=lst_k[1]
x1=lst_x[1]
y1=lst_y[1]
#获得第一节龙身的前把手坐标和斜率
k2=(d2/d1+k1)/(1-d2*k1/d1) #计算第一节龙身前把手和外前点直线的斜率
b=d2*np.sqrt(k1**2+1)+y1-k1*x1
if np.abs(b)<=np.abs(y1-k1*x1):
    b=-d2*np.sqrt(k1**2+1)+y1-k1*x1
#计算第一节龙身外侧边的截距
x=(y1-k2*x1-b)/(k1-k2)
y=(k1*y1-k1*k2*x1-k2*b)/(k1-k2)
#计算第一节龙身外前点的坐标
for i in np.arange(len(lst_k)):
    if lst_theta[i+1]-theta>=np.pi:
        ki=lst_k[i]
        xi=lst_x[i]
        yi=lst_y[i]
        d_chair=np.abs(ki*(x-xi)+yi-y)/np.sqrt(ki**2+1)
        #计算第一节龙身外前点到当前板凳中心线的距离
        if d_chair<d2:
            flag=1
        #判断是否相撞
x2=lst_x[2]
y2=lst_y[2]
#获得第二节龙身前把手的坐标
k2=(k1-d2/d1)/(1+d2*k1/d1) #计算第二节龙身前把手和第一节龙身外后点直线的
                                                    斜率
x=(y2-k2*x2-b)/(k1-k2)
y=(k1*y2-k1*k2*x2-b*k2)/(k1-k2)
```

```
    #计算第一节龙身外后点的坐标
    for i in np.arange(len(lst_k)):
        if lst_theta[i+1]-theta>=np.pi:
            ki=lst_k[i]
            xi=lst_x[i]
            yi=lst_y[i]
            d_chair=np.abs(ki*(x-xi)+yi-y)/np.sqrt(ki**2+1)
            #计算第一节龙身外后点到当前板凳中心线的距离
            if d_chair<d2:
                flag=1
            #判断是否相撞
    return flag
#用于判断是否发生碰撞
```

6. 问题二代码

```
        import numpy as np
import pandas as pd
from function import f1 #用于计算龙头位置随时间的变化
from function import f3 #用于计算盘入螺线上的位置迭代
from function import f4 #用于计算螺线上速度的斜率
from zeropoint import zero2 #用于计算函数f3的零点
from number import number #用于保留6位小数
from crashjudge import judge #用于判断是否发生碰撞
d=0.55 #螺距
v0=1 #龙头速度
theta0=32*np.pi #龙头初始极角
for theta in np.arange(60,0,-0.01):
    flag=judge(theta,d,v0)
    if flag:
        break
    #判断龙头处于当前位置时是否有板凳碰撞
for theta in np.arange(theta+0.01,theta-0.01,-0.0001):
    flag=judge(theta,d,v0)
    if flag:
        break
for theta in np.arange(theta+0.0001,theta-0.0001,-0.000001):
    flag=judge(theta,d,v0)
    if flag:
        break
#细化碰撞时龙头的极角theta
theta_chair0=theta+0.000001
t=d*(f1(theta0)-f1(theta))/(4*np.pi*v0) #计算碰撞的时刻
lst_chair_theta=[theta_chair0]
for i in np.arange(223):
    if i==0:
        d0=3.41-0.275*2
    else:
        d0=2.2-0.275*2
    #确定板长
    theta_last=lst_chair_theta[-1] #获得上一个把手的极角theta
    theta=zero2(f3,theta_last,theta_last+np.pi/2,10**(-8),d,d0,
                                        theta_last)
    lst_chair_theta.append(theta)
    #计算当前把手的极角theta
lst_chair_xyv=[]
for i in np.arange(224):
    lst_xyv=[]
```

```
    theta=lst_chair_theta[i]
    lst_xyv.append(d*theta*np.cos(theta)/(2*np.pi))
    lst_xyv.append(d*theta*np.sin(theta)/(2*np.pi))
    #根据theta角度确定把手坐标（x, y）
    lst_xyv.append(v0)
    lst_chair_xyv.append(lst_xyv)
lst_chair_xyv=np.array(lst_chair_xyv)
for i in np.arange(223):
    v_last=lst_chair_xyv[i,2] #获得上一个把手的速度
    theta_last=lst_chair_theta[i]
    theta=lst_chair_theta[i+1]
    x_last=lst_chair_xyv[i,0]
    y_last=lst_chair_xyv[i,1]
    x=lst_chair_xyv[i+1,0]
    y=lst_chair_xyv[i+1,1]
    #获得上一个把手和当前把手的坐标（x, y）和极角
    k_chair=(y_last-y)/(x_last-x)
    k_v_last=f4(theta_last)
    k_v=f4(theta)
    #计算板凳和两个速度的斜率
    aleph1=np.arctan(np.abs((k_v_last-k_chair)/(1+k_v_last*k_chair)
                                                    ))
    aleph2=np.arctan(np.abs((k_v-k_chair)/(1+k_v*k_chair)))
    #计算两个速度与板凳的夹角
    v=v_last*np.cos(aleph1)/np.cos(aleph2) #计算当前把手的速度
    lst_chair_xyv[i+1,2]=v
lst_chair_xyv=number(lst_chair_xyv,6) #保留6位小数
df=pd.DataFrame(lst_chair_xyv)
df.to_excel("result2_.xlsx",index=False)
#保存数据到Excel中
```

7. 问题三代码

```
        import numpy as np
from crashjudge import judge #用于判断是否发生碰撞
v0=1  #龙头速度
D=9 #调头空间的直径
for d in np.arange(0.55,0.4,-0.01):
    theta_min=D*np.pi/d #确定进入调头空间时的极角theta
    for theta in np.arange(theta_min+6,theta_min,-0.1):
        flag=judge(theta,d,v0)
        if flag:
            break
    if flag:
        break
    #判断当前螺距是否在调头空间外有板凳碰撞
for d in np.arange(d+0.01,d-0.01,-0.0001):
    theta_min=D*np.pi/d
    for theta in np.arange(theta_min+6,theta_min,-0.1):
        flag=judge(theta,d,v0)
        if flag:
            break
    if flag:
        break
for d in np.arange(d+0.0001,d-0.0001,-0.00001):
    theta_min=D*np.pi/d
    for theta in np.arange(theta_min+6,theta_min,-0.1):
        flag=judge(theta,d,v0)
```

```
        if flag:
            break
    if flag:
        break
for d in np.arange(d+0.00001,d-0.00001,-0.000001):
    theta_min=D*np.pi/d
    for theta in np.arange(theta_min+6,theta_min,-0.1):
        flag=judge(theta,d,v0)
        if flag:
            break
    if flag:
        break
for d in np.arange(d+0.000001,d-0.000001,-0.0000001):
    theta_min=D*np.pi/d
    for theta in np.arange(theta_min+6,theta_min,-0.1):
        flag=judge(theta,d,v0)
        if flag:
            break
    if flag:
        break
#细化最小的螺距
print(d) #输出在调头空间外不会发生碰撞的最小的螺距
```

8. dataproduce

```
        import numpy as np
from function import f4 #用于计算螺线上速度的斜率
from function import f5 #用于计算盘出螺线上的位置迭代
from zeropoint import zero2 #用于计算函数f5的零点
d=1.7 #螺距
v0=1 #龙头速度
D=9 #调头空间的直径
d0_1=3.41-0.275*2 #龙头板长
d0_2=2.2-0.275*2 #龙身和龙尾板长
theta0=D*np.pi/d #计算龙头0时刻时的极角
x0=D*np.cos(theta0)/2
y0=D*np.sin(theta0)/2
#计算龙头0时刻的坐标（x, y）
k=f4(theta0)
l1=2*np.abs(y0-k*x0)/np.sqrt(k**2+1)
l2=np.sqrt(D**2-l1**2)
r=D**2/(6*l1) #计算第二段圆弧的半径
x1=x0+2*r/np.sqrt(1+1/k**2)
y1=-(x1-x0)/k+y0
#计算第一段圆弧的圆心坐标
x2=-x0-r/np.sqrt(1+1/k**2)
y2=-(x2+x0)/k-y0
#计算第二段圆弧的圆心坐标
aleph=np.arccos(l2/(r*3))+np.pi/2 #计算两段圆弧的圆心角
theta_chair0_1=np.arccos((8*r**2-d0_1**2)/(8*r**2))
#计算第一节龙身前把手到达第一段圆弧时龙头的位置参数theta
theta_chair0_2=np.arccos((2*r**2-d0_1**2)/(2*r**2))
#计算第一节龙身前把手到达第二段圆弧时龙头的位置参数theta
a=theta0-np.pi
b=theta0-np.pi/2
theta_chair0_3=zero2(f5,a,b,10**(-8),d,d0_1,theta0-np.pi)
#计算第一节龙身前把手到达盘出螺线时龙头的位置参数theta
theta_chair_1=np.arccos((8*r**2-d0_2**2)/(8*r**2))
```

```
#计算龙身后把手到达第一段圆弧时前把手的位置参数theta
theta_chair_2=np.arccos((2*r**2-d0_2**2)/(2*r**2))
#计算龙身后把手到达第二段圆弧时前把手的位置参数theta
theta_chair_3=zero2(f5,a,b,10**(-8),d,d0_2,theta0-np.pi)
#计算龙身后把手到达盘出螺线时前把手的位置参数theta
t1=2*r*aleph/v0 #计算龙头到达第二段圆弧的时刻
t2=t1+r*aleph/v0 #计算龙头到达盘出螺线的时刻
theta1=np.arctan((y1-y0)/(x1-x0))+np.pi #计算第一段圆弧的进入点相对于圆心
                                          的极角
theta2=np.arctan((y2+y0)/(x2+x0)) #计算第二段圆弧的离开点相对于圆心的极角
x_=-x0/3
y_=-y0/3
#计算两段圆弧交界点的坐标
print(theta0,x0,y0,r,x1,y1,x2,y2,aleph,theta_chair0_1,
                                        theta_chair0_2,
      theta_chair0_3,theta_chair_1,theta_chair_2,theta_chair_3,t1,
                                            t2,theta1,
      theta2,x_,y_)
#输出各项重要参数
```

9. positioniteration

```
        import numpy as np
from function import f3 #用于计算盘入螺线上的位置迭代
from function import f5 #用于计算盘出螺线上的位置迭代
from function import f6 #用于计算前把手在第一段圆弧而后把手在盘入螺线的情形
from zeropoint import zero2 #用于计算函数f3和f5的零点
from zeropoint import zero3 #用于计算函数f6的零点
def iteration1(theta_last,flag_last,flag_chair):
    d=1.7 #螺距
    D=9 #调头空间的直径
    theta0=16.6319611 #龙头0时刻时的极角
    r=1.5027088 #第二段圆弧的半径
    aleph=3.0214868 #两段圆弧的圆心角
    if flag_chair==0:
        d0=3.41-0.275*2
        theta_1=0.9917636
        theta_2=2.5168977
        theta_3=14.1235657
    else:
        d0=2.2-0.275*2
        theta_1=0.5561483
        theta_2=1.1623551
        theta_3=13.8544471
    #确定板长和三个重要位置参数
    if flag_last==1:
        theta=zero2(f3,theta_last,theta_last+np.pi/2,10**(-8),d,d0,
                                            theta_last)
        #计算后把手的位置参数theta
        flag=1 #返回后把手所在曲线的类型
    #计算前把手和后把手都在盘入螺线的情形
    elif flag_last==2:
        if theta_last<theta_1:
            b=np.sqrt(2-2*np.cos(theta_last))*r*2
            beta=(aleph-theta_last)/2
            l=np.sqrt(b**2+D**2/4-b*D*np.cos(beta))
            gamma=np.arcsin(b*np.sin(beta)/l)
            theta=zero3(f6,theta0,theta0+np.pi/2,10**(-8),d,d0,
```

```
                                                    theta0,l,gamma)
            #计算后把手的位置参数theta
            flag=1 #返回后把手所在曲线的类型
            #计算前把手在第一段圆弧而后把手在盘入螺线的情形
        else:
            theta=theta_last-theta_1
            #计算后把手的位置参数theta
            flag=2 #返回后把手所在曲线的类型
            #计算前把手和后把手都在第一段圆弧的情形
    elif flag_last==3:
        if theta_last<theta_2:
            a=np.sqrt(10-6*np.cos(theta_last))*r
            phi=np.arccos((4*r**2+a**2-d0**2)/(4*a*r))
            beta=np.arcsin(r*np.sin(theta_last)/a)
            theta=aleph-phi+beta
            #计算后把手的位置参数theta
            flag=2 #返回后把手所在曲线的类型
            #计算前把手在第二段圆弧而后把手在第一段圆弧的情形
        else:
            theta=theta_last-theta_2
            #计算后把手的位置参数theta
            flag=3 #返回后把手所在曲线的类型
            #计算前把手和后把手都在第二段圆弧的情形
    else:
        if theta_last<theta_3:
            p=d*(theta_last+np.pi)/(2*np.pi)
            a=np.sqrt(p**2+D**2/4-p*D*np.cos(theta_last-theta0+np.
                                                    pi))
            beta=np.arcsin(p*np.sin(theta_last-theta0+np.pi)/a)
            gamma=beta-(np.pi-aleph)/2
            b=np.sqrt(a**2+r**2-2*a*r*np.cos(gamma))
            sigma=np.arcsin(a*np.sin(gamma)/b)
            phi=np.arccos((r**2+b**2-d0**2)/(2*r*b))
            theta=aleph-phi+sigma
            #计算后把手的位置参数theta
            flag=3 #返回后把手所在曲线的类型
            #计算前把手在盘出螺线而后把手在第二段圆弧的情形
        else:
            a=theta_last-np.pi/2
            b=theta_last
            theta=zero2(f5,a,b,10**(-8),d,d0,theta_last)
            #计算后把手的位置参数theta
            flag=4 #返回后把手所在曲线的类型
            #计算前把手和后把手都在盘出螺线的情形
    return [theta,flag]
#用于计算位置迭代
```

10. velocityiteration

```
        import numpy as np
from function import f4 #用于计算螺线上速度的斜率
def iteration2(v_last,flag_last,flag,theta_last,theta,x_last,y_last
                                        ,x,y):
    x1=-0.7600091
    y1=-1.3057264
    #计算第一段圆弧的圆心坐标
    x2=1.7359325
    y2=2.4484020
```

```
    #计算第二段圆弧的圆心坐标
    k_chair=(y_last-y)/(x_last-x) #计算板凳的斜率
    v=-1
    if flag_last==1 and flag==1:
        k_v_last=f4(theta_last)
        k_v=f4(theta)
        #计算前把手和后把手都在盘入螺线时两个速度的斜率
    elif flag_last==2 and flag==1:
        k_v_last=-(x_last-x1)/(y_last-y1)
        k_v=f4(theta)
        #计算前把手在第一段圆弧而后把手在盘入螺线时两个速度的斜率
    elif flag_last==2 and flag==2:
        v=v_last
        #计算前把手和后把手都在第一段圆弧的情形
    elif flag_last==3 and flag==2:
        k_v_last=-(x_last-x2)/(y_last-y2)
        k_v=-(x-x1)/(y-y1)
        #计算前把手在第二段圆弧而后把手在第一段圆弧时两个速度的斜率
    elif flag_last==3 and flag==3:
        v=v_last
        #计算前把手和后把手都在第二段圆弧的情形
    elif flag_last==4 and flag==3:
        theta_last=theta_last+np.pi
        k_v_last=f4(theta_last)
        k_v=k_v=-(x-x2)/(y-y2)
        #计算前把手在盘出螺线而后把手在第二段圆弧时两个速度的斜率
    else:
        theta_last=theta_last+np.pi
        theta=theta+np.pi
        k_v_last=f4(theta_last)
        k_v=f4(theta)
        #计算前把手和后把手都在盘出螺线时两个速度的斜率
    if v==-1:
        aleph1=np.arctan(np.abs((k_v_last-k_chair)/(1+k_v_last*
                                                k_chair)))
        aleph2=np.arctan(np.abs((k_v-k_chair)/(1+k_v*k_chair)))
        #计算两个速度与板凳的夹角
        v=v_last*np.cos(aleph1)/np.cos(aleph2) #计算当前把手的速度
    return v
#用于计算速度迭代
import numpy as np
from function import f4 #用于计算螺线上速度的斜率
def iteration2(v_last,flag_last,flag,theta_last,theta,x_last,y_last
                                        ,x,y):
    x1=-0.7600091
    y1=-1.3057264
    #计算第一段圆弧的圆心坐标
    x2=1.7359325
    y2=2.4484020
    #计算第二段圆弧的圆心坐标
    k_chair=(y_last-y)/(x_last-x) #计算板凳的斜率
    v=-1
    if flag_last==1 and flag==1:
        k_v_last=f4(theta_last)
        k_v=f4(theta)
        #计算前把手和后把手都在盘入螺线时两个速度的斜率
    elif flag_last==2 and flag==1:
        k_v_last=-(x_last-x1)/(y_last-y1)
```

```
        k_v=f4(theta)
        #计算前把手在第一段圆弧而后把手在盘入螺线时两个速度的斜率
    elif flag_last==2 and flag==2:
        v=v_last
        #计算前把手和后把手都在第一段圆弧的情形
    elif flag_last==3 and flag==2:
        k_v_last=-(x_last-x2)/(y_last-y2)
        k_v=-(x-x1)/(y-y1)
        #计算前把手在第二段圆弧而后把手在第一段圆弧时两个速度的斜率
    elif flag_last==3 and flag==3:
        v=v_last
        #计算前把手和后把手都在第二段圆弧的情形
    elif flag_last==4 and flag==3:
        theta_last=theta_last+np.pi
        k_v_last=f4(theta_last)
        k_v=k_v=-(x-x2)/(y-y2)
        #计算前把手在盘出螺线而后把手在第二段圆弧时两个速度的斜率
    else:
        theta_last=theta_last+np.pi
        theta=theta+np.pi
        k_v_last=f4(theta_last)
        k_v=f4(theta)
        #计算前把手和后把手都在盘出螺线时两个速度的斜率
    if v==-1:
        aleph1=np.arctan(np.abs((k_v_last-k_chair)/(1+k_v_last*
                                                k_chair)))
        aleph2=np.arctan(np.abs((k_v-k_chair)/(1+k_v*k_chair)))
        #计算两个速度与板凳的夹角
        v=v_last*np.cos(aleph1)/np.cos(aleph2) #计算当前把手的速度
    return v
#用于计算速度迭代
```

11. 第四问代码

```
        import numpy as np
import pandas as pd
from function import f2 #用于计算龙头位置随时间的变化
from zeropoint import zero1 #用于计算函数f2的零点
from number import number #用于保留6位小数
from positioniteration import iteration1 #用于计算位置迭代
from velocityiteration import iteration2 #用于计算速度迭代
d=1.7 #螺距
v0=1 #龙头速度
theta0=16.6319611 #龙头0时刻时的极角
r=1.5027088 #第二段圆弧的半径
aleph=3.0214868 #两段圆弧的圆心角
t1=9.0808299 #龙头到达第二段圆弧的时刻
t2=13.6212449 #龙头到达盘出螺线的时刻
x1=-0.7600091
y1=-1.3057264
#第一段圆弧的圆心坐标
x2=1.7359325
y2=2.4484020
#第二段圆弧的圆心坐标
theta1=4.0055376 #第一段圆弧的进入点相对于圆心的极角
theta2=0.8639449 #第二段圆弧的离开点相对于圆心的极角
lst_chair_theta=[]
lst_chair_flag=[]
```

```
for t in np.arange(-100,101):
    if t<0:
        theta_chair0=zero1(f2,theta0,100,10**(-8),theta0,v0,t,d)
        flag_chair0=1
    elif t==0:
        theta_chair0=theta0
        flag_chair0=1
    elif t<t1:
        theta_chair0=v0*t/(2*r)
        flag_chair0=2
    elif t<t2:
        theta_chair0=v0*(t-t1)/r
        flag_chair0=3
    else:
        theta_chair0=zero1(f2,theta0,100,10**(-8),theta0,v0,-t+t2,d
                                                    )-np.pi
        flag_chair0=4
    #给出龙头的位置参数theta和所在曲线的类型参数flag
    lst_theta=[theta_chair0]
    lst_flag=[flag_chair0]
    for i in np.arange(223):
        theta_last=lst_theta[-1] #获得上一个把手的位置参数theta
        flag_last=lst_flag[-1] #获得上一个把手所在曲线的类型参数flag
        [theta,flag]=iteration1(theta_last,flag_last,i)
        lst_theta.append(theta)
        lst_flag.append(flag)
    lst_chair_theta.append(lst_theta)
    lst_chair_flag.append(lst_flag)
    #计算当前把手的位置参数theta和所在曲线的类型参数flag
lst_chair_flag=np.array(lst_chair_flag)
lst_chair_theta=np.array(lst_chair_theta)
lst_chair_xy=[]
for i in np.arange(201):
    lst=[]
    for j in np.arange(224):
        flag=lst_chair_flag[i,j] #获得当前把手的位置参数theta
        theta=lst_chair_theta[i,j] #获得当前把手所在曲线的类型参数flag
        if flag==1:
            p=d*theta/(2*np.pi)
            x=p*np.cos(theta)
            y=p*np.sin(theta)
        elif flag==2:
            x=x1+2*r*np.cos(theta1-theta)
            y=y1+2*r*np.sin(theta1-theta)
        elif flag==3:
            x=x2+r*np.cos(theta2+theta-aleph)
            y=y2+r*np.sin(theta2+theta-aleph)
        else:
            p=d*(theta+np.pi)/(2*np.pi)
            x=p*np.cos(theta)
            y=p*np.sin(theta)
        #计算当前把手的坐标（x, y)
        lst.append(x)
        lst.append(y)
    lst_chair_xy.append(lst)
lst_chair_xy=np.array(lst_chair_xy).T
lst_chair_xy=number(lst_chair_xy,6) #保留6位小数
df=pd.DataFrame(lst_chair_xy)
```

```
df.to_excel("result4_1.xlsx",index=False)
#保存数据到Excel中
lst_chair_v=[]
for i in np.arange(201):
    lst_v=[v0]
    for j in np.arange(223):
        flag_last=lst_chair_flag[i,j]
        theta_last=lst_chair_theta[i,j]
        flag=lst_chair_flag[i,j+1]
        theta=lst_chair_theta[i,j+1]
        x_last=lst_chair_xy[j*2,i]
        y_last=lst_chair_xy[j*2+1,i]
        x=lst_chair_xy[j*2+2,i]
        y=lst_chair_xy[j*2+3,i]
        #获得上一个把手和当前把手的坐标，角度参数theta和所在曲线的位置参数flag
        v_last=lst_v[-1] #获得上一个把手的速度
        v=iteration2(v_last,flag_last,flag,theta_last,theta,x_last,
                                                y_last,x,y)
        lst_v.append(v)
        #计算当前把手的速度
    lst_chair_v.append(lst_v)
lst_chair_v=np.array(lst_chair_v).T
lst_chair_v=number(lst_chair_v,6) #保留6位小数
df=pd.DataFrame(lst_chair_v)
df.to_excel("result4_2.xlsx",index=False)
#保存数据到Excel中
```

12. velctorytheta

```
        import numpy as np
from positioniteration import iteration1 #用于计算位置迭代
from velocityiteration import iteration2 #用于计算速度迭代
def f(flag,theta):
    d=1.7 #螺距
    r=1.5027088 #第二段圆弧的半径
    aleph=3.0214868 #两段圆弧的圆心角
    x1=-0.7600091
    y1=-1.3057264
    #第一段圆弧的圆心坐标
    x2=1.7359325
    y2=2.4484020
    #第二段圆弧的圆心坐标
    theta1=4.0055376 #计算第一段圆弧的进入点相对于圆心的极角
    theta2=0.8639449 #计算第二段圆弧的离开点相对于圆心的极角
    if flag==1:
        p=d*theta/(2*np.pi)
        x=p*np.cos(theta)
        y=p*np.sin(theta)
        #计算位于盘入螺线时的坐标
    elif flag==2:
        x=x1+2*r*np.cos(theta1-theta)
        y=y1+2*r*np.sin(theta1-theta)
        #计算位于第一段圆弧时的坐标
    elif flag==3:
        x=x2+r*np.cos(theta2+theta-aleph)
        y=y2+r*np.sin(theta2+theta-aleph)
        #计算位于第二段圆弧时的坐标
    else:
```

```
        p=d*(theta+np.pi)/(2*np.pi)
        x=p*np.cos(theta)
        y=p*np.sin(theta)
        #计算位于盘出螺线时的坐标
    return [x,y]
#用于根据位置参数theta和所在曲线类型计算坐标
def v_theta(theta_last,flag_last,flag_chair,v_last):
    [theta,flag]=iteration1(theta_last,flag_last,flag_chair)
    #计算位置参数theta和所在曲线类型
    [x_last,y_last]=f(flag_last,theta_last)
    [x,y]=f(flag,theta)
    #计算前把手和后把手坐标
    v=iteration2(v_last,flag_last,flag,theta_last,theta,x_last,
                                        y_last,x,y)
    #计算速度
    return [theta,v,flag]
#用于计算速度和位置
```

13. 问题五代码

```
        import numpy as np
from velctorytheta import v_theta #用于计算速度和位置
v0=1 #龙头速度
v_max=2 #最大速度
theta0=16.6319611 #龙头0时刻时的极角
theta_chair0_3=14.1235657 #第一节龙身前把手到达盘出螺线时龙头的位置参数theta
lst_theta0=np.arange(theta0-np.pi,theta_chair0_3,0.001)
lst_v=[]
lst_flag=[]
lst_theta=[]
for theta0 in lst_theta0:
    [theta,v,flag]=v_theta(theta0,4,0,v0)
    lst_theta.append(theta)
    lst_v.append(v)
    lst_flag.append(flag)
for j in np.arange(2):
    for i in np.arange(len(lst_theta0)):
        theta_last=lst_theta[i]
        v_last=lst_v[i]
        flag_last=lst_flag[i]
        [theta,v,flag]=v_theta(theta_last,flag_last,1,v_last)
        lst_theta[i]=theta
        lst_v[i]=v
        lst_flag[i]=flag
#获得该过程中第三节龙身前把手的速度随龙头位置参数theta的变化
theta0=lst_theta0[lst_v.index(max(lst_v))] #获得速度最大时龙头的位置参
                                            数theta
lst_theta0=np.arange(theta0-0.001,theta0+0.001,0.000001)
lst_v=[]
lst_flag=[]
lst_theta=[]
for theta0 in lst_theta0:
    [theta,v,flag]=v_theta(theta0,4,0,v0)
    lst_theta.append(theta)
    lst_v.append(v)
    lst_flag.append(flag)
for j in np.arange(2):
    for i in np.arange(len(lst_theta0)):
```