基于改进欧拉法的“板凳龙”的运动学模型

摘　要

本文主要基于**改进欧拉法**和对**碰撞约束模型**研究“板凳龙”行进的**运动学模型**。基于运动的机理分析得到各状态量的控制方程和递推公式.

对于问题一，首先引入了极坐标系，建立了等距螺线的极坐标方程；其次，通过对时间进行离散化，取$\Delta t = 0.1s$，利用几何关系和杆的刚性约束得到关于**角速度**与极角的微分方程，通过分离变量法，得到T与极角θ的方程，由于θ与T无显式的函数关系式，先采取$fsolve$ 函数，每次迭代给定猜测值，得到一组离散的相关变量，同时也使用**改进欧拉法**，利用迭代逼近，得到各时刻下“把手”极角θ与T的离散变量。并且得到两组算法的误差平方和，数量级在 10^{-18}，进一步证明改进欧拉的准确性与高效。分析通过角速度递推式得到各时刻下“把手”的角速度，利用极坐标方程和杆的刚性约束得到各时刻下“板凳龙”的极坐标参量与速度；最后，将极坐标参量转化成笛卡尔坐标，求解出各时刻下“板凳龙”的位置。

对于问题二，首先引入**碰撞约束模型**，该模型通过计算龙头个点 在各时刻下与龙身板块组成的三角形面积求和，得到避免发生碰撞的约束条件；其次，通过几何关系求解出各时刻下各块“板凳龙”的顶点坐标；然后代入坐标，利用面积公式建立约束方程；最后求解出当$\boldsymbol{t = 412s}$时发生碰撞，“板凳龙”龙头和第 **28** 个龙身发生碰撞。

对于问题三，建立最小螺距优化模型，要求在给定调头区域时，搜寻最小螺距使得龙头可以顺利进入调头区域；约束条件为龙头进入调头区域之前不与其他“板凳龙”发生碰撞；先采用二分法区间搜索算法，得到螺距s位于$[0.4337, 0.4338]$，之后再对其进行 步长为0.00001的小步长搜索，求得**最小螺距为$\boldsymbol{0.4338m}$**。

对于问题四，首先仿照问题三建立最小调头路径优化模型，要求在给定螺距的情况下，搜寻最小调头路径，使得“板凳龙”能够顺利进入调头区域并盘出；约束条件为龙头进入调头区域之前不与其他“板凳龙”发生碰撞，且“板凳龙”要在调头区域内顺利调出，利用二分法求得**最小调头路径为$\boldsymbol{13.477m}$**；其次在此基础上，为求解各时刻“板块龙”的位置和速度，除了利用问题一的**运动学模型**，还建立了“板凳龙”在调头路径上的运动学模型，最终求解出给定时刻下“板凳龙”的位置和速度。

对于问题五，在问题四的基础上建立了对于最大龙头速度的优化模型；约束条件是“板凳龙”前进速度不超过$2m/s$；最终采用**遍历搜索法**，对于较小搜索间隔$\Delta v_1 = 0.001m/s$，求解得到龙头**最大前进速度为$\boldsymbol{1.1051m/s}$**。

关键词：等距螺线；改进欧拉算法；碰撞约束模型；二分区间搜索算法；遍历搜索

一、 问题重述

1.1 问题背景

本研究针对板凳龙这一文化现象，旨在通过建立舞龙队行进的运动学模型优化舞龙队的行进路径和速度控制。在保持板凳之间稳定连接的前提下，确定最佳的行进方式，以减少板凳之间的碰撞，提高舞龙的整体流畅度和观赏性。

1.2 问题提出

问题一：舞龙队沿着螺距 0.55m 等距螺线顺时针盘入，每个把手中心均位于螺线上。龙头前把手的行进速度始终保持 1m/s，初始时刻龙头位于螺线第 16 圈的 A 点。需要计算从初始时刻到 300s 之间，每秒整个舞龙队的位置和速度（包括龙头、龙身和龙尾的各个把手中心的位置和速度），

问题二：舞龙队继续按照问题 1 的设定沿螺线盘入。需要确定舞龙队盘入的终止时刻，使得各板凳之间不发生碰撞（即不能再继续盘入的时间）。并给出此时舞龙队的位置和速度。

问题三：舞龙队将从顺时针盘入调头为逆时针盘出，这需要一定的调头空间。假设调头空间是以螺线中心为圆心，直径为 9m 的圆形区域。需要确定最小螺距，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。

问题四：在问题 3 设定的调头空间内，舞龙队完成调头路径由两段圆弧相切连接而成的 S 形曲线，前一段圆弧的半径是后一段的 2 倍，并且与盘入和盘出螺线相切。需要研究能否通过调整圆弧，仍保持各部分相切，使得调头曲线变短。

问题五：舞龙队按照问题 4 设定的路径行进，龙头的行进速度保持不变。需要确定龙头的最大行进速度，使得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

针对问题一，引入极坐标系，建立等距螺线的极坐标方程，然后建立板凳龙在盘入螺线上的运动学模型，运用数值解法，求解出各时刻下板凳龙的各状态量。

2.2 问题二的分析

针对问题二，引入碰撞约束模型，将复杂抽象的碰撞检测转化为直观的通过面积检测的方法。通过遍历时间，求解出碰撞时刻以及该时刻下板凳龙的各状态量。

2.3 问题三的分析

针对问题三，建立最小螺距优化模型，要求在给定调头区域时，搜寻最小螺距使得龙头可以顺利进入调头区域；约束条件为龙头进入调头区域之前不与其他“板凳龙”发生碰撞，采用二分法区间搜索算法搜寻最小螺距。

2.4 问题四的分析

建立最小调头路径优化模型，要求在给定螺距的情况下，搜寻最小调头路径，使得“板凳龙”能够顺利进入调头区域并盘出；约束条件为龙头进入调头区域之前不与其他“板凳龙”发生碰撞，且“板凳龙”要在调头区域内顺利调出，其次在此基础上，为求解各时刻“板块龙”的位置和速度，建立了“板凳龙”在调头路径上的运动学模型，可以求解出给定时刻下“板凳龙”的位置和速度。

2.5 问题五的分析

在问题四的基础上建立对于最大龙头速度的优化模型；约束条件是“板凳龙”前进速度不超过$2m/s$；采用遍历搜索法进行搜索搜寻龙头的最大速度。

三、 模型假设

1. 板凳龙从主视角看，近似看成刚性轻杆；
2. 板凳龙从俯视角看，近似看成有一定宽度的刚性轻杆；
3. 假设时间是离散的，按秒分解整体的运动；
4. 板凳龙整体上看做刚性杆构成的链状结构；
5. 板凳龙只有进入螺线时才开始记录其位置和速度，未进入时假设其不存在。

四、 符号说明

符号	说明	单位
b	螺线的增量因子	m
r	极径	m
θ	极角	*度*/°
v_i	行进速度	m/s
v_r	径向速度	m/s
v_θ	横向速度	m/s
$v_{\|\|i}$	沿杆速度	m/s
α	极径方向与龙身方向夹角	*度*/°
ω	行进角速度	rad/s
T	时间	s
v_1	龙头行进速度	m/s
l_{AB}	孔到端点的距离	m
l	端点到极点的距离	m
β	l_{AB}与板中线的夹角	*度*/°
s	螺距	m
z	四个三角形和与底面四边形的面积差	m^2
S	三角形面积	m^2
R	调头区域半径	m

五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 模型的建立

1. 极坐标系的引入[1]

由题可知，板凳龙表演为等距螺线运动，以螺线中心为极点，以极点发出的水平正向射线为极轴，则其极坐标方程为

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

其中，r为极径，θ为极角，根据等距螺线的基本性质，其螺距受b控制，为$2\pi b$。

2. 板凳龙行进运动学模型的建立

板凳龙在沿螺线运动中可以视为刚性轻杆在沿螺线运动，板凳龙把手视为杆的连接点，连接点始终固定在螺线上，杆在连接点处是自由的，即连接点两端的杆夹角是可变的。

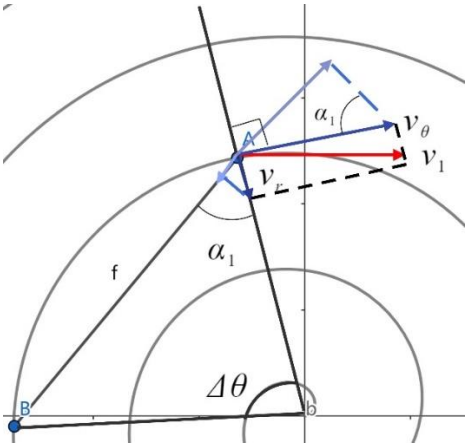

图 1. 速度分解示意图

板凳龙把手的速度视为连接点在螺线上运动的速度，方向是该点在曲线上切线的方向。根据曲线运动的性质，板凳龙速度可以按图示沿极径方向的径向速度v_r和垂直极径方向的横向速度v_θ分解。即

$$\boldsymbol{v} = \boldsymbol{v}_\theta + \boldsymbol{v}_r \tag{2}$$

根据几何关系，将径向速度和横向速度投影到杆方向，得到沿杆速度:

$$v_{//i} = v_{\theta i} sin\alpha_i + v_{ri} cos\alpha_i \tag{3}$$

其中α_i，α_{i+1}为极径方向与龙身方向的夹角。

对于刚性轻杆，杆两端沿杆速度相等，有速度的控制方程:

$$v_{//i+1} = v_{//i} \tag{4}$$

即：

$$v_\theta sin\alpha_{i+1} + v_r cos\alpha_{i+1} = v_\theta sin\alpha_i + v_r cos\alpha_i \tag{5}$$

根据余弦定理：

$$\cos\alpha_i = \frac{r_i^2 + L^2 - r_{i+1}^2}{2Lr_i}, \cos\alpha_{i+1} = \frac{r_{i+1}^2 + L^2 - r_i^2}{2Lr_{i+1}} sin\alpha_i = \sqrt{1 - cos^2\alpha_i} \tag{6}$$

根据曲线运动的运动规律：

$$v_{ri} = b\omega_i, v_{\theta_i} = b\theta_i\omega_i \tag{7}$$

其中ω_i为第i个板凳龙的角速度，θ_i是第i个板凳龙的极角；

联立（1）（3）（4）（5）解得：

$$\omega_{i+1} = \omega_i \cdot \frac{cos\alpha_i + \theta_i sin\alpha_i}{cos\alpha_{i+1} + \theta_{i+1} sin\alpha_{i+1}} \tag{8}$$

联立（5）（7）可求得：

$$v_i = b\omega_i \sqrt{\theta_i^2 + 1} \tag{9}$$

其中v_i为第 i 个板凳龙的线速度。

5.1.2 模型的求解

由（8），每一块板的角速度只与前面一块板的角速度相关，即与前一块板的状态量有关，与之后板的状态无关。考虑采取**隔离法**，只考虑每一块板与前一块板组成的系统，即通过（8），求解出每一块板的状态量。由于题目要求求解出每一时刻板凳龙

的状态量，考虑对时间进行离散化，取$\Delta t = 0.1s$。

1. 龙头状态量的求解

龙头的速度v_1保持不变，因此$\omega_1 = \frac{v_1}{b\sqrt{\theta^2+1}}$，而角速度可以看成极角关于时间的导数，则$\frac{d\theta_1}{dt} = \frac{v_1}{b\sqrt{\theta_1^2+1}}$。这是一个变量可分离方程，于是

$$v_1 dt = b\sqrt{\theta^2+1}\, d\theta \tag{10}$$

对方程两边进行积分可得：

$$\int_0^T v_1 dt = \int_\theta^{32\pi} b\sqrt{\theta^2+1}\, d\theta \tag{11}$$

代入龙头速度$v_1 = 1m/s$ ；

$$T = \frac{b}{2}(16\pi\sqrt{1024\pi^2+1} - \theta_1\sqrt{\theta_1^2+1} + \ln\left(32\pi + \sqrt{1024\pi^2+1}\right) - \ln\left(\theta_1 + \sqrt{\theta_1^2+1}\right) \tag{12}$$

我们想要得到θ与 t 的显示函数关系式，然而此方程并没有显示的解析解，因此可以利用数值求解的算法。每次迭代给定时间坐标以及猜测值，借助$fsolve$函数，可以得到方程的精确数值解。因此通过（12），可以求解出每一时刻下第一块板的极角，进而得到对应的角速度ω_1。

实际上，对于（10），也可以采取改进欧拉法进行求解，对于初值问题：

$$\begin{cases} \dot{\theta} = f(t,\theta) \\ \theta(0) = \theta_0 \end{cases} \tag{13}$$

改进欧拉法的过程如下：

$$\begin{cases} \theta_f = \theta_i + \Delta t f(t_i + \theta_i) \\ \theta_b = \theta_i + \Delta t f\left(t_{i+1} + \theta_f\right) \\ \theta_{i+1} = \frac{1}{2}\left(\theta_f + \theta_b\right) \end{cases} (i = 1,2,3\ldots) \tag{14}$$

2. 龙身状态量的求解

从第二块板凳龙开始，角速度通过（8）求解。参照龙头的处理思路，将角速度看成极角关于时间的导数：

$$\frac{\mathrm{d}\theta_{i+1}}{\mathrm{d}t} = \omega_i \cdot \frac{cos\alpha_i + \theta_i sin\alpha_i}{cos\alpha_{i+1} + \theta_{i+1} sin\alpha_{i+1}} (i = 1,2,3\ldots 223) \tag{16}$$

采用**改进欧拉法**求解出每一时刻下该板凳龙对应的极角，再代入（1）（8）（9），求解出板凳龙把手的极径和速度。最后，将其极坐标转化成笛卡尔坐标。

5. 1. 3 模型的结果分析

1. 龙头前把手、龙身前把手、龙尾后把手的位置分析

表 1 龙头前把手、龙身前把手、龙尾后把手的位置

	0 s	60 s	120 s	180 s	240 s	300 s
龙头 $x(m)$	8.800000	5.799209	-4.084887	-2.963608	2.594494	4.420274
龙头 $y(m)$	0.000000	-5.771092	-6.304479	6.094780	-5.356743	2.320429
第一节龙身$x(m)$)	-	7.387688	-1.805533	-4.824438	4.244141	3.408911
第一节龙身$y(m)$	-	-3.582457	-7.322009	4.801323	-4.215000	3.688052
第 51 节龙身$x(m)$)	-	-	-1.460849	7.406218	-6.796955	0.403661
第 51 节龙身$y(m)$	-	-	8.274835	-2.019796	0.948812	-5.904693
第 101 节龙身$x(m)$	-	-	-	-8.513686	7.383062	-4.112374
第 101 节龙身$y(m)$	-	-	-	-0.535267	2.276914	5.408542

第 151 节龙身$x(m)$	-	-	-	-	-6.236239	7.244468
第 151 节龙身$y(m)$	-	-	-	-	-5.909433	-2.521320
第 201 节龙身 $x(m)$	-	-	-	-	-	-8.504197
第 201 节龙身 $y(m)$	-	-	-	-	-	-0.689201
龙尾(后) $x(m)$	-	-	-	-	-	-
龙尾(后) $y(m)$	-	-	-	-	-	-

注解：其中“-”表示未进入螺线。

将表 1 数据分别用解析式法与欧拉法进行比较，得到误差分析图如下图所示：

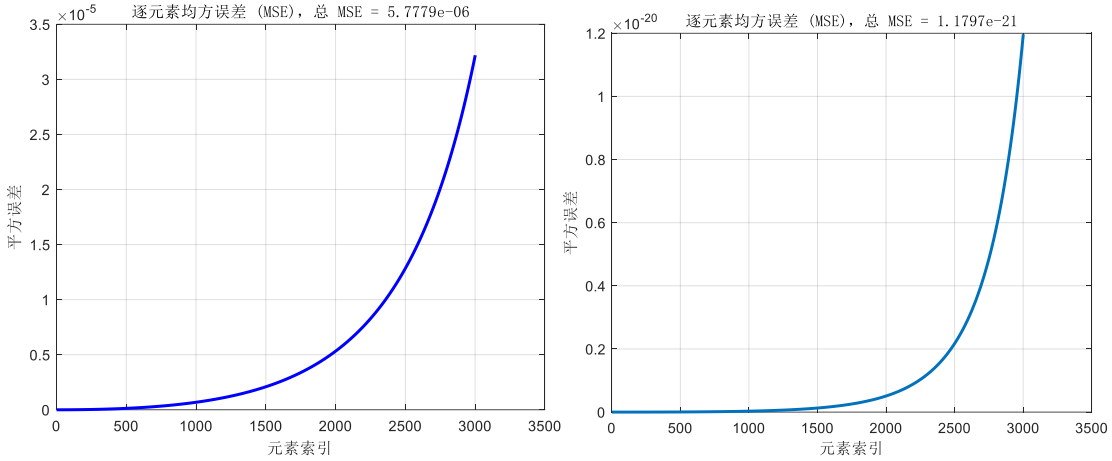

图 2. 解析式法与欧拉法/改进欧拉法的逐元素均方误差图

由图可以得出，随着时间的推进，误差的平方一直呈现指数级增长，时间步长取 0.1s，分别求得**解析式法与欧拉法**和**解析式法与改进欧拉法**的误差平方和分别为：

表 2 两种方法时间步长和误差平方和

时间步长 t=0.1s	误差平方和
解析式法与欧拉法	0.0173395706468081
解析式法与改进欧拉法	$3.54038845676244\times 10^{-18}$

由表 2 可以看出，解析法与改进欧拉法的误差平方和更小，运算效率高，准确度也高。因此采用解析式法与改进欧拉法。

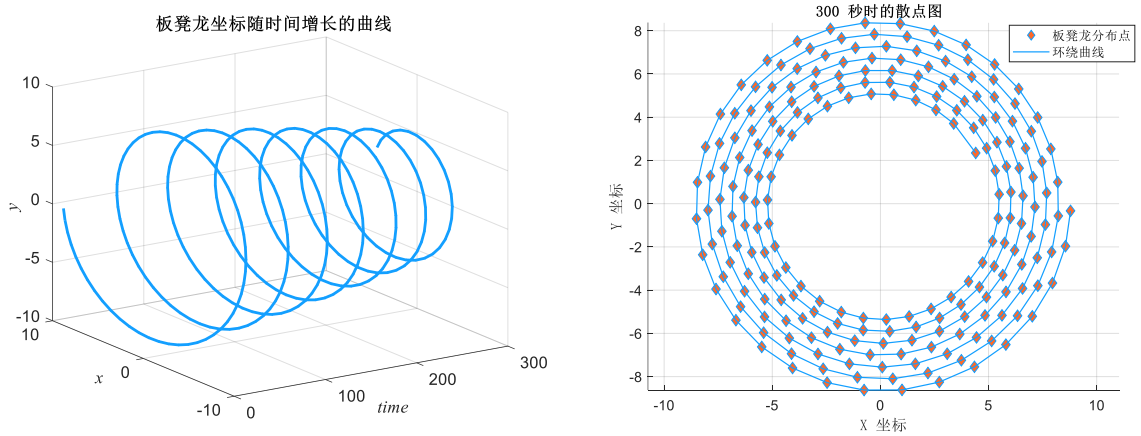

图 3. 板凳龙的时空坐标以及 300s 时板凳龙的空间坐标

由图 3 可以得到，随着时间的推移，板凳龙越来越靠近中心原点坐标，呈现坍缩的趋势，但 300s 的时候并没有到达中心原点坐标，也没有发生碰撞。300s 时的空间分布如图 3 所示。

2. 龙头前把手、龙身前把手、龙尾后把手的速度分析

表 3 龙头前把手、龙身前把手、龙尾后把手的速度

	0 s	60 s	120 s	180 s	240 s	300 s
龙头(m/s)	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
第一节龙身(m/s)	1.003332	1.003648	1.004070	1.004672	1.005632	1.003332
第 51 节龙身(m/s)	0.000000	1.105603	1.118172	1.136147	1.164866	0.000000
第 101 节龙身(m/s)	0.000000	0.000000	1.228781	1.263192	1.317988	0.000000
第 151 节龙身(m/s)	0.000000	0.000000	0.000000	1.388999	1.469063	0.000000
第 201 节龙身(m/s)	0.000000	0.000000	0.000000	0.000000	1.615781	0.000000
龙尾(后) (m/s)	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

由表 3 数据可得，每个把手速度随时间的变化如下图所示：

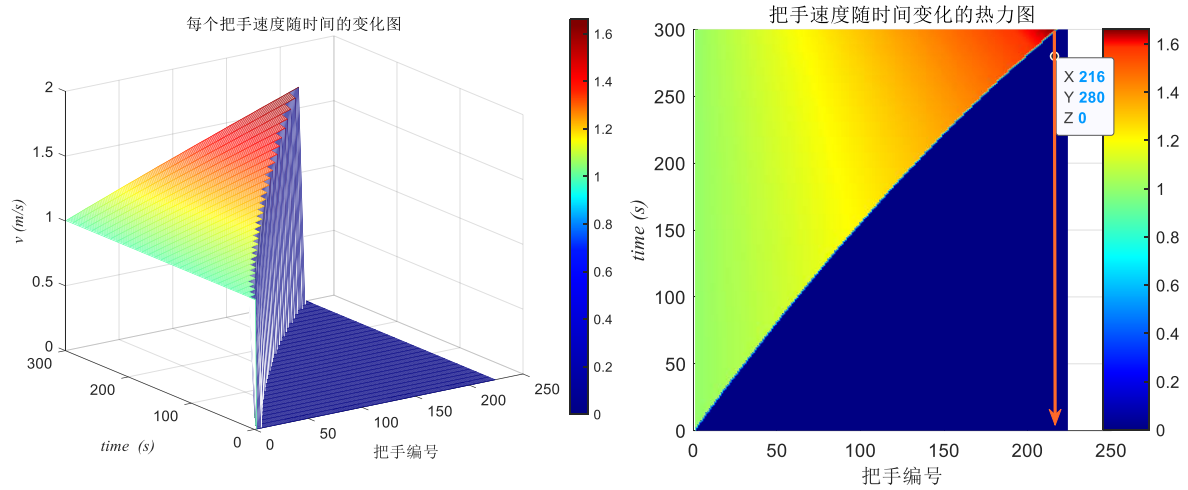

图 4. 每个把手速度随时间的变化图

由图可知：龙头的速度稳定在 1m/s，第一百七十节左右的龙身速度最快；最后一段速度始终保持在 0m/s，印证了龙尾自始至终没有进入螺线。

5.2 问题二模型的建立与求解

5.2.1 模型的建立

1. 碰撞约束模型的引入[2]

不妨将龙头和龙身的碰撞模拟成矩形i和矩形j的碰撞，矩形i和矩形j发生相撞的所有可能情况：（a）矩形i存在顶点与矩形j相撞，但矩形j的所有顶点都不会和矩形i相撞；（b）矩形j存在顶点和矩形i相撞，但矩形i的所有顶点都不会和矩形j相撞；(c)矩形i存在顶点与矩形j相撞，且矩形j存在顶点和矩形i相撞；以及（d）矩形i和矩形j的全部顶点均不相撞。如下图所示：

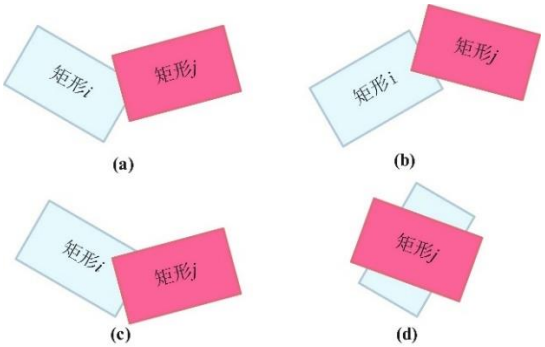

图 5. 不同情况下的碰撞示意图

约束条件为：

$$S_{\Delta PAB} + S_{\Delta PBC} + S_{\Delta PCD} + S_{\Delta PDA} > S_{\square ABCD} \tag{15}$$

其中S_{Δ}为三角形面积，$S_{\square}$为矩形面积。

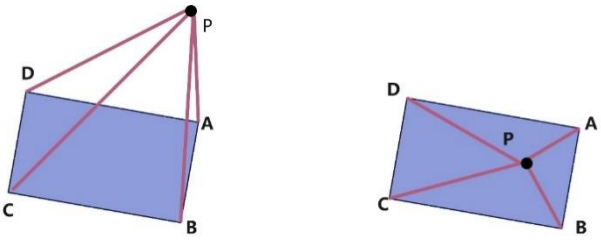

图 6. 限制点 P 处于矩形 ABCD 外部的一种基于图形面积的约束条件示意图

2. 板凳龙碰撞检测模型的建立

在本题实际研究背景下，当龙头与后面某块板凳龙快要发生碰撞而又没有碰撞时，板凳龙就已经无法行进了，而此时碰撞避免约束条件还严格成立。

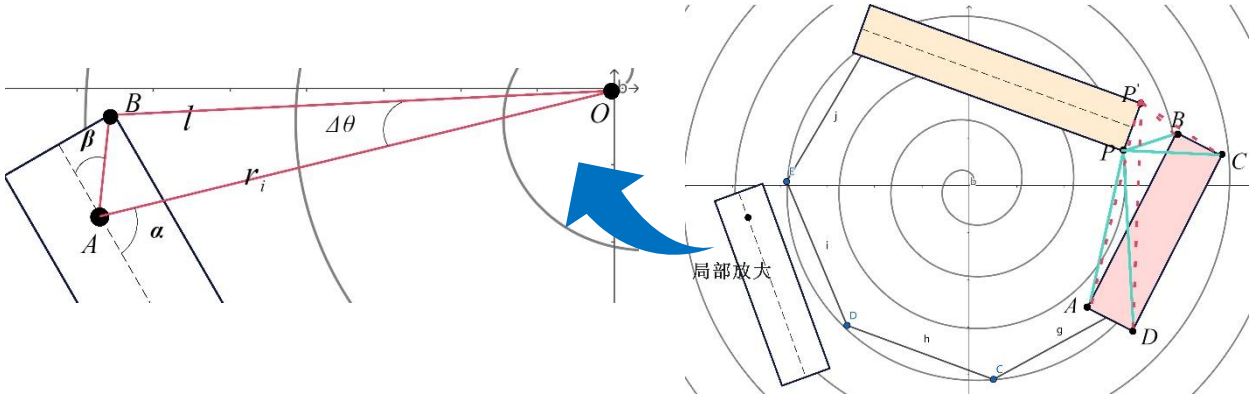

图 7. 碰撞检测示意图

因此，基于实际情况，如图 7，考虑对

$$z = S_{\Delta PAB} + S_{\Delta PBC} + S_{\Delta PCD} + S_{\Delta PDA} - S_{\square ABCD} \tag{16}$$

进行考察。可以认为存在一个极小量$\varepsilon > 0$，当$z - \varepsilon \leq 0$时，认为此时板凳龙无法前进。

5.2.2 模型的求解

1. 板凳龙各顶点的求解

为求解z，需求解每一块板凳龙的四个顶点的坐标。板凳龙顶点与板凳龙把手的几何关系如图 7 左侧放大图所示。

首先求解图 7 中的 B 点（B 点的意义是靠近前把手的右侧顶点，A 点为前把手），对ΔOAB进行解三角形，可得：

$$\begin{cases} l^2 = r_i^2 + l_{AB}^2 - 2r_i l_{AB}\cos(\pi - \beta - \alpha) \\ \cos(\Delta\theta) = \dfrac{l^2 + r_i^2 - l_{AB}^2}{2r_i l^2} \end{cases} \tag{17}$$

其中：

$$l_{AB} = \sqrt{(27.5)^2 + 15^2},\ tan\beta = \frac{15}{27.5}, r_i = b\theta_i \tag{18}$$

联立（17）（18），解得 B 点在极坐标系下坐标$(l, \theta_i - \Delta\theta)$，进而转化成笛卡尔坐标。而由于靠近前把手的左侧顶点与 B 点关于杆轴对称，其坐标可以通过坐标对称公式给出。

假设杆的直线方程为$Ax + By + C = 0$,B 点的笛卡尔坐标为(x_0, y_0),则 B 关于杆的对称点(x, y)为：

$$\begin{cases} x = x_0 - 2A\dfrac{Ax_0 + By_0 + C}{A^2 + B^2} \\ y = y_0 - 2B\dfrac{Ax_0 + By_0 + C}{A^2 + B^2} \end{cases} \tag{19}$$

同理，靠近后把手的两个顶点也由上述步骤得出。

2. 板凳龙碰撞检测

从某一时刻t起，每隔 1s 对所有板凳龙的顶点坐标进行求解。利用平面中任意三点$(x_1, y_1), (x_2, y_2), (x_3, y_3)$的面积公式

$$S = \frac{1}{2}[x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)] \tag{20}$$

联立（16）~（20），判断当前时刻下龙头与第二节龙身以后每一节龙身（因为龙头与第一节龙身始终相接，故不考虑第一节龙身）是否有$z - \varepsilon \leq 0$，若有，则当前时刻已经发生碰撞，输出上一时刻，即碰撞尚未发生的时刻。

5. 2. 3 模型的结果分析

当$t = 412s$时，龙头与龙身发生了碰撞，此时龙头、龙身、龙尾的位置和速度如表 4 所示：

表 4 发生碰撞时龙头、龙身、龙尾的位置和速度

	横坐标 $x\ (m)$	纵坐标 $y\ (m)$	速度 (m/s)
龙头（前）	-0. 184452	2. 337127	1. 000000
第一节龙身	-0. 999546	2. 154990	1. 012624
第 51 节龙身	3. 082945	1. 269779	1. 367652
第 101 节龙身	-0. 190067	4. 254348	1. 710140
第 151 节龙身	-4. 356292	2. 793605	2. 036298
第 201 节龙身	5. 908629	1. 391391	2. 329502
龙尾(后)	0. 198030	-6. 462148	2. 485154

由表 4 数据可得，发生碰撞的时候龙头、龙身、龙尾的位置如图 9 所示：

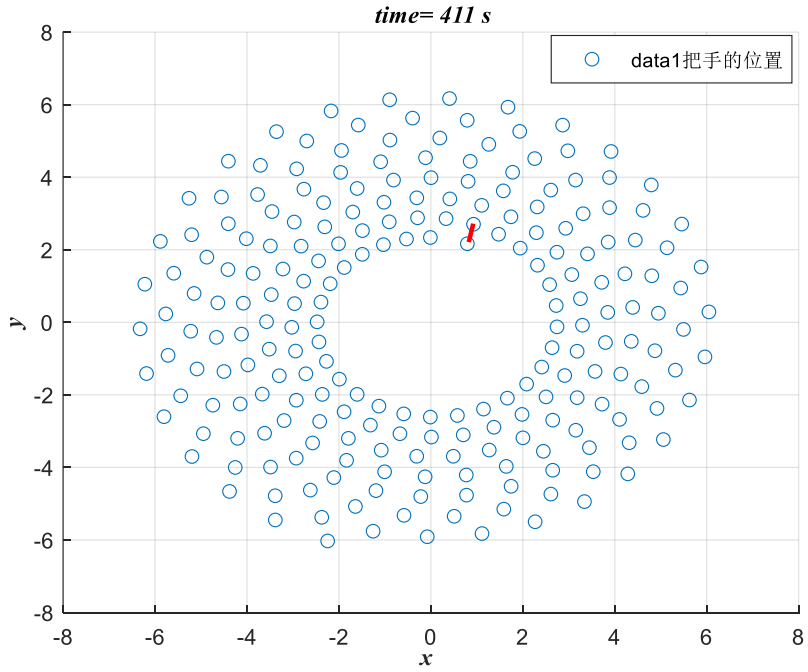

图 9. 板凳龙发生碰撞示意图

从表 5 中可以看出：$time = 411s$为终止时间，$time = 412s$，两个红色连线处会发生碰撞,发生碰撞的区域为龙头和第 28 个龙身。

表 5.碰撞时间和碰撞的板块

终止时间	碰撞时间	发生碰撞的区域
411s	412s	龙头和第 28 节龙身

5.3 问题三模型的建立与求解

5.3.1 问题三模型的建立

由题可知，若龙头前把手可以顺利进入调头空间，则在进入调头空间前龙头前把手不与后面的龙身相碰撞。对于s的边界选取，根据问题二，当发生碰撞时，龙头的极径小于给定调头空间的半径,因此作为约束条件的上边界;每一块板凳龙的宽为 0.3m，螺距最小不能低于 0.3m，否者板凳龙无法并排盘入，因此作为约束条件的下边界。综上所述，优化模型为

$$s.t.\ \begin{cases} \min s \\ z-\varepsilon>0,(\theta_i>\theta_0) \\ 0.3\le s\le 0.55 \end{cases} \tag{21}$$

5.3.2 问题三模型的求解

1.二分法思想

由问题一条件得到 螺距$s\epsilon$[0.3,0.55] $\min s = 0.3$。

利用经典的二分区间搜索算法:

	时间复杂度$O(log\ n)$
Step1	$When\ \ (max_s - min_s)$ >$1\cdot 10^{-3}$ mid=$\frac{1}{2}(max_s + min_s)$
Step2	$s = mid$ Flag=check(s)
Step3	if flag==0 min_s= mid else max_s= mid end

注解： ***check*()函数作用是传入螺距 *s* 并**检测是否碰撞，如果碰撞返回 0 ,反之返回 1.

2.约束条件边界的选取

如图 10，如果还按照取初始极角为32π，当 s 取得下边界时，龙头尚未转过一圈就进入调头空间，因此永远不会发生碰撞。此时采用二分法会因为下界恒成立而使得最优解不断向下界逼近，从而使得二分法失效。于是，本部分考虑龙头从固定点位进入，即从问题一中的 A 点进入，此时我们认为初始极角$\theta_0 = (\left[\frac{8.8}{s}\right] + 1) * 2\pi$，其中[$x$]表示不超过 x 的最大整数。

5.3.3 问题三模型求解的分析

由表 6 得出，由于发生碰撞的最大螺距是 0.4337m，当$t = 454\ s$时候，发生碰撞，龙头与第 27 个把手的板凳相撞。

因此最小螺距为 0.4338m。此时螺线分布如图 10 所示。

表 6.最小螺距、碰撞时间以及碰撞区域

最小螺距	碰撞时间	发生碰撞的区域
0.4338m	454s	龙头和第 27 节龙身把手

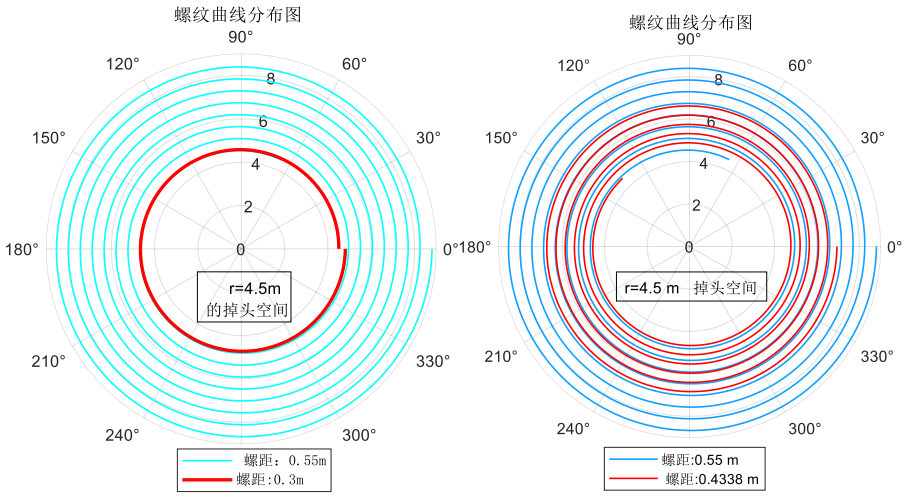

图 10. 螺距为$0.3m$和$0.4338m$时螺纹曲线分布图

5.4 问题四模型的建立与求解

5.4.1 模型的建立

1. 调头路径的建立

本题基于问题 3 设定的调头区域内完成掉头。调头的路径是两段圆弧相切连接而成，并且与盘入、盘出的螺线都相切。而调头区域盘入，盘出螺线也相切，根据**垂直关系公理**，过直线上一点有且仅有一条直线与已知直线垂直。因此 S 形曲线应由两个半圆组成，且两个半圆和调头区域的圆心在同一条直线上，且直线过极心。两个半圆直径之和等于调头区域的直径。

因此，调头路径具有唯一性且与调头区域的半径相关。

2.最小调头路径的优化模型

实际上，题目仅要求在给定的调头区域内调头，并不是刚进入给定范围内就进行调头。因此，我们把实际的调头区域的半径 R 作为目标函数进行优化。我们需要知道 R 的范围。

实际调头区域的半径上界应为题目所给定的半径值，即 4.5m；考虑对下界进行确定，因为前段圆弧半径是后一段的 2 倍，已知龙头前后把手的长度 2.86m，发现当龙头两把手之间的距离大于小半圆的半径时，即当龙头运动到图 10 位置之后，右侧会产生向上的运动趋势，而左侧也会随之上升，进而与后侧相撞，因此小半圆的直径应该大于龙头两把手之间的长度。

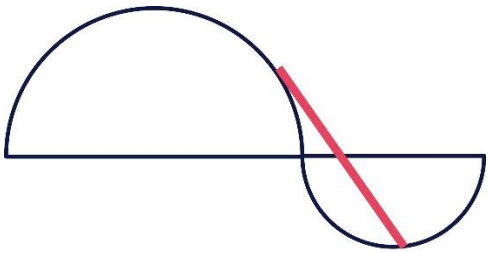

图 10. 调头路径的临界情况

并且还要求龙头进入调头区域之前，板凳不能发生碰撞。综上所述，有：

$$s.t.,\begin{cases} min\ R \\ z-\varepsilon>0,(\theta_i>\theta_{0)} \\ 4.29<R\ \leq 4.5 \\ \theta_0=\dfrac{R}{b} \\ s=2\pi b \end{cases} \tag{22}$$

3. 最小调头路径情况下板凳龙在调头路径上的运动学模型

I．板凳龙在最小调头路径上速度的确定

板凳龙在调头路径上速度的确定较为简单，由于龙头速度始终保持恒定，因此板凳龙进入调头路径后是做匀速圆周运动，速度大小等于龙头速度大小。当一块板凳龙前把手进入调头路径而后把手尚未进入调头路径时，为方便计算，我们可以把此时板凳龙后把手的速度看成前把手的速度。

II．板凳龙在最小调头路径上位置的确定

当板凳龙进入调头路径时，为确定把手位置，需要将其与两半圆的圆心和极心建立联系，具体的几何关系如图所示：

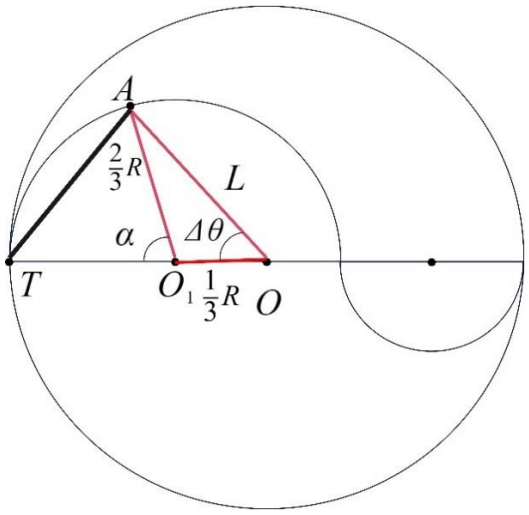

图 11. 板凳龙调头路径运动示意图

根据图 11，设$T(R,\theta)$ 调头区域圆的圆心到板凳龙前把手的长度为 L，调头区域的圆半径为 R，根据上述条件可以求得：

$$\alpha=\omega T \tag{23}$$

根据余弦定理，可以求得：

$$L^2=(\frac{R}{3})^2+(\frac{2R}{3})^2-2\cdot\frac{R}{3}\cdot\frac{2R}{3} \tag{24}$$

进而求得：

$$cos(\Delta\theta)=\frac{L^2+(\frac{R}{3})^2-(\frac{2R}{3})^2}{2L\cdot\frac{R}{3}} \tag{25}$$

此时，A 点的极坐标，可以表示为（$L,\theta-\Delta\theta$）。同理，当板凳龙运动到小半圆上时，也参照此法得出把手的极坐标。

5.4.2 模型的求解

1. 最小调头路径的求解

仿照问题三，问题三在给定调头区域下求得使得龙头能够顺利进入的最小螺距，而问题四则给定了螺距，要求求得最小调头区域。依然采用**二分法**，对 R 的范围进行逐步二分求解，找到最小调头区域。

2. 板凳龙在-100s 到 100s 的位置、速度求解

对于进入调头区域位置的求解，我们可以参照问题三，给定进入的初始位置以及初始极角，并开始计时，直到龙头进入调头区域以后，停止计时。若最终时刻超过 100s，则最终时刻往前倒退 100s 的位置和速度就是所需要推导的-100s:0s 的位置和速度；

由于前部分假设当前把手进入时，后把手的速度看成前把手的速度，因此，通过 此法可以得到 0s 以后还未进入实际调头区域内各板凳龙的速度和位置；

当板凳龙从调头区域进入盘出曲线时，由于盘出曲线与盘入曲线中心对称，因此每块板凳龙刚从调头区域出去的速度就是刚从调头区域进入的速度，位置坐标关于 $y=x$ 对称。

5.4.3 模型求解的分析

1. 最小调头曲线

通过 R 的范围进行逐步二分求解，求解出最小半径 R＝4.29m，得到最小调头曲线长度为 13.477m。

2. 龙头前把手、龙身前把手、龙尾后把手的位置分析

表 7. 龙头前把手、龙身前把手、龙尾后把手的位置

	-100s	-50 s	0 s	50 s	100 s
龙头 $x(m)$	8.500000	6.459675	-4.258409	-2.671458	6.136066
龙头 $y(m)$	-0.000000	-1.866391	-0.231811	-4.675102	-4.286513
第一节龙身$x(m)$)	0.000000	6.797319	-3.928393	-0.736434	7.352201
第一节龙身$y(m)$	0.000000	0.601487	-1.921328	-5.438957	-1.856887
第 51 节龙身$x(m)$)	0.000000	0.000000	5.085040	6.119457	-8.395825
第 51 节龙身$y(m)$	0.000000	0.000000	-0.204229	-0.211655	1.082400
第 101 节龙身$x(m)$	0.000000	0.000000	0.728058	-2.172161	2.143012
第 101 节龙身$y(m)$	0.000000	0.000000	5.440508	7.853140	-8.500000
第 151 节龙身$x(m)$	0.000000	0.000000	0.000000	0.000000	0.000000
第 151 节龙身$y(m)$	0.000000	0.000000	0.000000	0.000000	0.000000
第 201 节龙身 $x(m)$	0.000000	0.000000	0.000000	0.000000	0.000000
第 201 节龙身 $y(m)$	0.000000	0.000000	0.000000	0.000000	0.000000
龙尾(后) $x(m)$	8.500000	6.459675	-4.258409	-2.671458	6.136066
龙尾(后) $y(m)$	-0.000000	-1.866391	-0.231811	-4.675102	-4.286513

3. 龙头前把手、龙身前把手、龙尾后把手的速度分析

表 8. 龙头前把手、龙身前把手、龙尾后把手的速度

	-100s	-50 s	0 s	50 s	100 s
龙头(m/s)	1.000000	1.000000	1.000000	1.000000	1.000000
第一节龙身(m/s)	1.107074	1.012101	1.020820	1.016510	1.010904
第 51 节龙身(m/s)	0.000000	0.000000	1.032367	1.318321	1.210904
第 101 节龙身(m/s)	0.000000	0.000000	1.034367	1.554369	1.192146
第 151 节龙身(m/s)	0.000000	0.000000	1.436727	1.579422	1.179422
第 201 节龙身(m/s)	0.000000	0.000000	1.320820	1.079422	1.279422
龙尾(后) (m/s)	0.000000	0.000000	0.000000	1.119133	1.131137

5.5 问题五模型的建立与求解

5.5.1 模型的建立

在问题四的背景下，搜寻龙头的最大速度，使得板凳龙前进的速度不超过 2*m/s*。由于在调头区域做匀速圆周运动，进入该区域的板凳龙的速度会始终等于龙头速度。因此当龙头速度不超过 2*m/s* 时，就无需考虑调头区域内的速度。又根据前文研究，盘入曲线与盘出曲线中心对称，状态完成相同。因此只需要考察盘入螺线上各块板凳龙的速度。根据问题四的结果，v_1 取 1m/s，当龙头进入调头区域时，每块板凳龙的速度均不超过 2m/s，故取 1m/s 为下界，对 v_1 进行优化。参照问题一建立的运动学模型，得到如下优化模型：

$$
s.t.,\begin{cases}
\max v_1 \\
1 \le v_1 < 2 \\
\cos\alpha_i = \dfrac{r_i^2 + L^2 - r_{i+1}^2}{2Lr_i}, \cos\alpha_{i+1} = \dfrac{r_{i+1}^2 + L^2 - r_i^2}{2Lr_{i+1}} \quad sin\alpha_i = \sqrt{1 - cos^2\alpha_i} \\
\omega_i = \omega_{i-1} \cdot \dfrac{cos\alpha_{i-1} + \theta_{i-1} sin\alpha_{i-1}}{cos\alpha_i + \theta_i sin\alpha_i} \\
v_i = b\omega_i \sqrt{\theta_i^2 + 1} \\
v_i < 2, i = 2,3,\ldots,224,
\end{cases} \tag{26}
$$

5.5.2 模型的求解

由于$1 \le v_1 < 2$，区间较小，采用**二分法**效率不高，因此可以考虑采取**遍历搜索法**。为提高精度，取尽可能小的搜索间隔$\Delta v_1 = 0.001m/s$，代入（26）进行求解。对每一个时间间隔，求解各块板凳龙的速度，并判断是否满足约束条件。

5.5.3 模型的求解分析

表九. 龙头前把手、龙身前把手、龙尾后把手的速度

龙头的最大速度	遍历搜索法消耗时间
$1.1052m/s$	156.412221s

由表九可以看出，遍历搜索法消耗的时间少，效率更高。

六、 模型的评价与改进

6.1 模型的优点

对于板凳龙的运动机理模型采取了数值求解算法，运用了改进欧拉的算法，对于龙头的坐标将解析解与数值解采取误差平方和的分析方法，得到的误差在 10^{-18} 数量级，相较于 *fslove* 解决隐式的方程具有更好的精度型 ，由于该算法依靠迭代求解，所以求解速度更快；

在检测板凳龙之间是否发生碰撞时，引入碰撞约束模型，将复杂抽象的碰撞判断转化为直观的面积大小比较，大大简化模型的复杂程度。

6.2 模型的缺点

在运用碰撞约束模型时，引入了极小量ε控制模型，可能会引起误差，改变板凳龙的状态量。

6.3 模型的改进

在运用数值算法时，可以采取精度更好的数值算法，增加数据精确程度；

在对优化问题进行搜索求解时，可对搜索间隔更加细分，使得结果结果更加靠近全局最优解。

七、 参考文献

[1] 王明华, 杨继绪. 阿基米德螺线的性质与应用[J]. 数学通报, 1989, 7(5).

[2] 李柏. 复杂约束下自动驾驶车辆运动规划的计算最优控制方法研究[D].浙江大学,2018.

[3] Serpe A, Frassia M G. Task mathematical modelling design in a dynamic geometry environment: Archimedean spiral's algorithm[C]//International Conference on Numerical Computations: Theory and Algorithms. Cham: Springer International Publishing, 2019: 478-491.

[4] Liu Q, Huang G, Zhang X, et al. Speed Planning and Interpolation Algorithm of Archimedes Spiral Based on Tangential Vector[J]. International Journal of Precision Engineering and Manufacturing, 2024: 1-14.

附录

附录 1 第一问

```
clc, clear, close all;
%%初始化参数
L1 = 2.86; %龙头长度 单位：m
L2 = 1.65; %龙身龙尾长度
s = 0.55; %初始螺线宽距
b = s / (2 * pi);
delta_theta_1 = delta(L1, s);
delta_theta_2 = delta(L2, s);
dt = 0.1;
t_end = 352;
t0 = 0:dt:t_end;
t0 = t0';
len_t = length(t0);
v = zeros(len_t, 224); %定义线速度,一共 224 个把手
theta = v;
Omega = v; %定义角速度
r = v; %定义极径
v(:, 1) = 1;
%%计算第一个点的角速度

w1_funs = @(t, theta) 1 ./ (b * sqrt(1 + (32 * pi - theta) .^ 2));
theta(:, 1) = 32 * pi - dis_euler(w1_funs, 1, t_end * 10, 0);
r(:, 1) = b .* theta(:, 1);
Omega(:, 1) = 1 ./ sqrt(r(:, 1) .^ 2 + b ^ 2);

%%计算第二个的角速度
i = 2 * ones(1, 222);

while (abs(theta(i(1), 1) - theta(1, 1)) < delta_theta_1)
    i (1) = i(1) + 1;
end

%刚性杆的约束条件
calpha1 = @(t, theta_2) (L1 ^ 2 + r(t, 1) .^ 2 - (b * (32 * pi - theta_2)) .^ 2) ./ (2 *
L1 .* r(t, 1));
salpha1 = @(t, theta_2)sqrt(1 - calpha1(t, theta_2) .^ 2);
calpha2 = @(t, theta_2)((b * (32 * pi - theta_2)) .^ 2 + L1 ^ 2 - r(t, 1) .^ 2) ./ (2 * b .*
(32 * pi - theta_2) .* L1);
salpha2 = @(t, theta_2)sqrt(1 - calpha2(t, theta_2) .^ 2);
% 沿杆的分速度相等
funs = @(t, theta_2) Omega(t, 1) .* (calpha1(t, theta_2) + theta(t, 1) .* salpha1(t,
```

```
theta_2)) ./ (calpha2(t, theta_2) + (32 * pi - theta_2) .* salpha2(t, theta_2));
    y0 = 0;
    h = 0.1;
    theta(i(1):end, 2) = 32 * pi - dis_euler(funs, i(1), 3000, y0);
    r(i(1):end, 2) = b .* theta(i(1):end, 2);

    for k = i(1):length(t0)
        Omega(k, 2) = funs(k, 32 * pi - theta(k, 2));
    end

    v(i(1):end, 2) = Omega(i(1):end, 2) .* sqrt(r(i(1):end, 2) .^ 2 + b ^ 2);
    %计算第二个点以后的角速度
    flag = 0;

    for j = 3:224
        i(j - 1) = i(j - 2);

        while (abs(theta(i(j - 1) + 1, j - 1) - theta(i(j - 2) + 1, j - 1)) < delta_theta_2)
            i (j - 1) = i(j - 1) + 1;

            if i(j - 1) > 3000
                flag = 1;
                break;
            end

        end

        if flag == 1
            break;
        end

        calpha1 = @(t, theta_2) (L2 ^ 2 + r(t, j - 1) .^ 2 - (b * (32 * pi - theta_2)) .^ 2) ./
(2 * L2 .* r(t, j - 1));
        salpha1 = @(t, theta_2)sqrt(1 - calpha1(t, theta_2) .^ 2);
        calpha2 = @(t, theta_2)((b * (32 * pi - theta_2)) .^ 2 + L2 ^ 2 - r(t, j - 1) .^ 2) ./
(2 * b .* (32 * pi - theta_2) .* L2);
        salpha2 = @(t, theta_2)sqrt(1 - calpha2(t, theta_2) .^ 2);
        funs = @(t, theta_2) Omega(t, j - 1) .* (calpha1(t, theta_2) + theta(t, j - 1) .*
salpha1(t, theta_2)) ./ (calpha2(t, theta_2) + (32 * pi - theta_2) .* salpha2(t, theta_2));
        y0 = 0;
        h = 0.1;
        theta(i(j - 1):end, j) = 32 * pi - dis_euler(funs, i(j - 1), 3000, y0);
        r(i(j - 1):end, j) = b .* theta(i(j - 1):end, j);
```

```
        for k = i(j - 1):length(t0)
            Omega(k, j) = funs(k, 32 * pi - theta(k, j));
        end

        v(i(j - 1):end, j) = Omega(i(j - 1):end, j) .* sqrt(r(i(j - 1):end, j) .^ 2 + b ^ 2);
    end

    %%转化为直角坐标
    [x, y] = pol2cart(theta, r);
    y(1, 1) = 0;

    for n = 1:222

        if i(n) <= 3001
            y(i(n), n + 1) = 0;
        else
            break;
        end

    end

    %%%%将没进入盘龙的板凳坐标更新为 nan
    for i = 2:224

        for j = 1:len_t - 1

            y(j, i) = nan;

            if y(j + 1, i) ~= 0
                y(j, i) = 0;
                break;
            else
                y(j + 1, i) = nan;
            end

        end

    end

    for i = 2:224

        for j = 1:len_t - 1

            x(j, i) = nan;
```

```
            if x(j + 1, i) == 0
                x(j, i) = nan;

                if j == len_t - 1
                    x(j + 1, i) = nan;
                end

                continue;
            else
                x(j, i) = nan;
                break;
            end

        end

    end

    %%%% 螺旋线-三维空间
    t_step = 0:1:300;
    t_step = (t_step / dt) + 1;
    figure;
    plot3(t0(t_step), x(t_step), y(t_step), 'LineWidth', 2); % 使用 plot3 绘制三维曲线
    grid on; % 打开网格
    title('三维空间中的曲线 (x, y, t)', 'FontSize', 14, 'FontWeight', 'bold');
    xlabel('时间 (t)', 'FontSize', 12, 'FontWeight', 'bold'); % x 轴为时间 t
    ylabel('X 坐标', 'FontSize', 12, 'FontWeight', 'bold'); % y 轴为 x
    zlabel('Y 坐标', 'FontSize', 12, 'FontWeight', 'bold'); % z 轴为 y
    set(gca, 'FontSize', 12); % 设置坐标轴字体大小

    %%%%%速度的三维空间

    % figure;
    % handles = 1:1:224; % 把手编号

    % % 生成时间和把手编号的网格
    % [H, T] = meshgrid(handles, t0(t_step));

    % % 绘制网格图
    % mesh(H, T, v(t_step, :));

    % % 设置图形标题和轴标签
    % title('把手速度随时间的变化', 'FontSize', 14, 'FontWeight', 'bold');
    % xlabel('把手编号', 'FontSize', 12, 'FontWeight', 'bold'); % x 轴为把手编号
```

```
% ylabel('时间 (s)', 'FontSize', 12, 'FontWeight', 'bold'); % y 轴为时间
% zlabel('速度 (m/s)', 'FontSize', 12, 'FontWeight', 'bold'); % z 轴为速度

% % 打开网格，调整坐标轴
% grid on;
% set(gca, 'FontSize', 12); % 设置坐标轴字体大小

% % 去掉 shading 插值（网格不需要插值）
% colorbar; % 添加颜色条，表示速度大小

% 绘制三维曲面图
figure;
handles = 1:1:224;

[H, T] = meshgrid(handles, t0(t_step));
surf(H, T, v(t_step, :));
% 设置图形标题和轴标签
title('把手速度随时间的变化', 'FontSize', 14, 'FontWeight', 'bold');
xlabel('把手编号', 'FontSize', 12, 'FontWeight', 'bold'); % x 轴为把手编号
ylabel('时间 (s)', 'FontSize', 12, 'FontWeight', 'bold'); % y 轴为时间
zlabel('速度 (m/s)', 'FontSize', 12, 'FontWeight', 'bold'); % z 轴为速度

% 打开网格，调整坐标轴
grid on;
set(gca, 'FontSize', 12); % 设置坐标轴字体大小
shading interp; % 平滑曲面插值，去除网格线
colorbar; % 添加颜色条，表示速度大小
% % 绘制第 300 秒的散点图
% figure;
% scatter(x(end, :), y(end, :), 50, 'filled'); % 绘制散点图，'50' 是点的大小，'filled' 表示填充点
% title('300 秒时的散点图', 'FontSize', 14, 'FontWeight', 'bold'); % 设置标题
% xlabel('X 坐标', 'FontSize', 12, 'FontWeight', 'bold'); % 设置 x 轴标签
% ylabel('Y 坐标', 'FontSize', 12, 'FontWeight', 'bold'); % 设置 y 轴标签
% grid on; % 打开网格
% set(gca, 'FontSize', 12); % 设置坐标轴字体大小
% axis equal; % 保持 x 和 y 轴比例相等
% hold on
% plot(x(end, :), y(end, :))

x_tran = x'; % 转置后的 x
y_tran = y'; % 转置后的 y
v_tran = v'; % 转置后的 v
step = 0:1:300;
```

```
step_index = step / dt +1;
%  对  x, y, v  进行采样
x_1 = x_tran(:, step_index); %  按时间间隔采样  x
y_1 = y_tran(:, step_index);
v_1 = v_tran(:, step_index);

excel_position = zeros(224 * 2, 301);
excel_v = zeros(224, 301);

for i = 1:224 * 2

    if mod(i, 2) ~= 0 %为奇数
        j = floor(i / 2) + 1; % (i/2 ) +1
        excel_position(i, :) = x_1(j, :);
    else %偶数
        k = i / 2;
        excel_position(i, :) = y_1(k, :);
    end

end

% %%%%%%%%%%%%论文当中的表
% colum_step = [1 61 121 181 241 301]; %下标
% row_step1 = [0 1 51 101 151 201 223] + 1;
% row_step2 = [1 2 3 4 103 104 203 204 303 304 403 404 447 448];
% paper_matrix1 = excel_position(row_step2, colum_step); %position
% paper_matrix2 = v_1(row_step1, colum_step); %speed
% header = {' 0 s', '60 s', ' 120 s', '180 s', '240 s', '300 s'};
% Table1 = array2table(paper_matrix1, 'VariableNames', header);
% Table2 = array2table(paper_matrix2, 'VariableNames', header);
% %  将数据写入  Excel  文件
% filename1 = 'F:\EXCEL\position.xlsx';
% filename2 = 'F:\EXCEL\speed.xlsx';
% writetable(Table1, filename1);
% writetable(Table2, filename2);
% disp('paper  速度  and  位置  已写入!');

% "F:\EXCEL\result1.xlsx"
%%%%%%写入文件

%  文件路径
filename = 'F:\EXCEL\result1.xlsx';

[~, ~, raw_data] = xlsread(filename, '位置');
```

```
start_row = 2;
start_col = 2;

[row, col] = size(excel_position);
raw_data(start_row:(start_row + row - 1), start_col:(start_col + col - 1)) = num2cell(excel_position);

writecell(raw_data, filename, 'Sheet', '位置');

disp('矩阵已成功写入指定位置，并保留了文字信息');
writetable(excel_position, filename, 'Sheet', '位置');

disp('x, y 已写入 "位置" 表，速度 v 已写入 "速度" 表');
syms x a v
f = (a ./ v) .* sqrt(x ^ 2 + 1); % 定义符号表达式
I = int(f, x); % 计算不定积分

% 使用 simplify 简化表达式
simplified_I = simplify(I)

% % 使用 rewrite 将表达式转换为对数形式
log_form = rewrite(simplified_I, 'log')

% 使用 matlabFunction 将表达式转换为 MATLAB 函数，并保存到文件
matlabFunction(log_form, 'File', 'generated_function.m');
function y = dis_euler(f, a, b, y0) % % % % % % %   欧拉法
    dt = 0.1;
    s = b - a + 1;
    Y = zeros(1, s + 1);
    Y(1) = y0;

    for k = 1:s
        y_pred = Y(k) + dt * f(a + k - 1, Y(k));

        % 修正步骤（使用预测值进行修正）
        Y(k + 1) = Y(k) + (dt / 2) * (f(a + k - 1, Y(k)) + f(a + k, y_pred));
        % Y(k + 1) = Y(k) + h * f(a + k - 1, Y(k));
    end

    y = Y';
end
clc; clear; close all;
%%本文件针对于第一个版本比较欧拉法与解析式法的误差矩阵
format long g
```

```
%%% 参数
dt = 0.1;
t = 0:dt:300;
len_t = length(t);
cir_num = 16; % 16 圈
dx = 0.55; % 间隔 55cm
len_A = cir_num * dx;

%%%%%%%%%       r = a * theta
a = dx / (2 * pi);

%%%%%%%%%
theta_result = zeros(1, len_t);
diff_ = zeros(1, len_t);

% 初始猜测值
theta_guess = 32 * pi;

% 初始值 y0 的计算
y0 = (a / 2) * ((32 * pi) * sqrt((32 * pi) ^ 2 + 1) + asinh(32 * pi));

% 设置 fsolve 的高精度选项
options = optimoptions('fsolve', ...
    'Display', 'off', ... % 关闭迭代信息显示
    'TolFun', 1e-7, ... % 函数值容差
    'TolX', 1e-7, ... % 解的容差
    'MaxIter', 1000, ... % 最大迭代次数
    'MaxFunEvals', 2000); % 最大函数计算次数

for i = 1:len_t
    % 定义目标函数作为 fsolve 输入
    fun = @(theta) y0 - (a / 2) * (theta * sqrt(theta ^ 2 + 1) + asinh(theta)) - t(i);

    % 使用 fsolve 求解 theta
    [theta_result(i), diff_(i)] = fsolve(fun, theta_guess, options);

    % 更新猜测值（步长调整为 -0.01）
    theta_guess = theta_result(i) - 0.01;
end

theta_colum = theta_result'; %转置矩阵
%%初始化参数
L1 = 2.86; %龙头长度 单位：m
L2 = 1.65; %龙身龙尾长度
```

```
s = 0.55; %初始螺线宽距
b = s / (2 * pi);
delta_theta_1 = delta(L1, s);
delta_theta_2 = delta(L2, s);
t0 = 0:0.1:300;
t0 = t0';
v = zeros(length(t0), 1); %定义线速度
theta = v;
w = v; %定义角速度
r = v; %定义极径
v(:) = 1;
%%计算第一个点的角速度

w1_funs = @(t, theta) 1 ./ (b * sqrt(1 + (32 * pi - theta) .^ 2));
theta = 32 * pi - dis_euler(w1_funs, 1, 3000, 0);
r = b .* theta;
w = 1 ./ sqrt(r .^ 2 + b ^ 2);

deltaz = abs(theta - theta_colum); %欧拉法和解析式法误差矩阵
element =    power((theta - theta_colum) , 2);
mse = mean(element);



% 绘制逐元素 MSE 图
figure;
plot(element,   'LineWidth', 2);
title(['逐元素均方误差 (MSE)，总 MSE = ', num2str(mse)]);
xlabel('元素索引');
ylabel('平方误差');
grid on;
disp("误差平方和是 : ");
figure;
boxplot(element);   % 绘制箱线图
title('逐元素误差平方的箱线图', 'FontSize', 14, 'FontWeight', 'bold');   % 设置标
题
xlabel('数据组', 'FontSize', 12, 'FontWeight', 'bold');   % 设置 x 轴标签
ylabel('误差平方值', 'FontSize', 12, 'FontWeight', 'bold');   % 设置 y 轴标签
set(gca, 'FontSize', 12);   % 设置坐标轴字体大小

grid on;   % 打开网格
z = sum(element); %误差平方和
disp(z);
function theta = delta(L, s)
    %用于求解两端点径向方向间初始夹角
```

```
        %输入:L  杆长
        %s  螺距宽
        %输出:两端点径向方向间初始夹角
        r = s * 16;
        syms theta
        f = @(theta) cos(theta) - ((s * (32 * pi - theta) / (2 * pi)) ^ 2 + r ^ 2 - L ^ 2) / (2
* (s * (32 * pi - theta) / (2 * pi)) * r);
        theta = fzero(f, 0.5);
    end
    function [alpha1, alpha2] = alpha(r, b, theta, L1, L2, i)
        calpha1 = (L1 ^ 2 + r(i(1):end, 1) .^ 2 - r(i(1):end, 2) .^ 2) ./ (2 * L1 .* r(i(1):end,
1));
        alpha1(:, 1) = acos(calpha1);
        calpha2 = (r(i(1):end, 2) .^ 2 + L1 ^ 2 - r(i(1):end, 1) .^ 2) ./ (2 .* r(i(1):end) .*
L1);
        alpha2(:, 1) = acos(calpha2);

        for j = 2:223
            calpha3 = (L2 ^ 2 + r(i(n):end, n) .^ 2 - (b * (32 * pi - theta(i(n + 1:end, n +
1)) .^ 2))) ./ (2 * L2 .* r(i(n):end, 1));
            alpha1(:, j) = acos(calpha3);
            calpha4 = ((b * (32 * pi - theta(:, j + 1))) .^ 2 + L2 ^ 2 - r(:, j) .^ 2) ./ (2 *
b .* (32 * pi - theta(:, j + 1)) .* L2);
            alpha2(:, j) = acos(calpha4);
        end

    end
```

附录 2 第二问

```
clc; clear; close all;

[r0, theta0, Omega, v0] = prepare_2(412);

[x, y] = pol2cart(theta0, r0);
x = x';
y = y';
v0 = v0';
scatter(x, y);
```

```
% %%%%%%%%%%%%论文当中的表
step = [1 51 101 151 201] + 1; %下标
paper_matrix1 = x(step); %x
paper_matrix2 = y(step); %y
paper_matrix3 = v0(step);
combined_matrix = [paper_matrix1, paper_matrix2, paper_matrix3];
header = {'x', 'y', 'v'};
Table1 = array2table(combined_matrix, 'VariableNames', header);
% 将数据写入 Excel 文件
filename1 = 'F:\EXCEL\que2_position_v.xlsx';
% filename2 = 'F:\EXCEL\speed.xlsx';
writetable(Table1, filename1);
disp('paper 速度 and 位置 已写入!');
function [x_sym, y_sym] = symmetric_point(A, B, P)
    %A=[x,y]:为前把手的笛卡尔 坐标点
    %B=[x,y]:为后把手的笛卡尔 坐标点
    % p 为需要对称的点的坐标
    % 输入: A, B 为两个点，定义直线；P 为给定点 [x, y]
    % 输出: P 关于直线的对称点 [x_sym, y_sym]

    % 提取直线的两个点坐标
    x1 = A(1);
    y1 = A(2);
    x2 = B(1);
    y2 = B(2);

    % 提取给定点 P 的坐标，靠近 原点 方向 弧线内侧上的点
    x0 = P(1);
    y0 = P(2);

    % 判断是否为竖直线
    if x1 == x2
        % 如果是竖直线，对称点在 x 轴方向镜像
        x_sym = 2 * x1 - x0;
        y_sym = y0; % y 值不变
    else
        % 计算直线的斜率和截距
        m = (y2 - y1) / (x2 - x1);
        b = y1 - m * x1;

        % 计算垂足点 T 的坐标
        xt = (x0 + m * (y0 - b)) / (1 + m ^ 2);

        yt = (m * (x0 + m * (y0 - b))) / (1 + m ^ 2) + b;
```

```
        % 计算对称点 P'的坐标
        x_sym = 2 * xt - x0;
        y_sym = 2 * yt - y0;
    end

end
clc, clear, close all;
flag = 1;
z_k = 0.025; %板凳面积的占比系数
z = z_k * 0.66; %误差控制系数   m²      板凳面积的 3 %
number_i = 31; %从龙头紧接的板子（即就是 倒数第二块板子） 开始往后遍历个数

for t = 410:0.1:420

    %准备工作
    [r, theta, ~, ~] = prepare_2(t);
    %%%%%%% alpha1 当作前把手时 ： 板方向与极矢方向的夹角
    %%%%%%% alpha2 当作后把手时 ： 板方向与极矢方向的夹角
    [alpha1, alpha2] = Palpha(r);
    %特定角度
    beta0 = atan(15/27.5); % %板凳上任意一把手 坐标 与 右 or 左 顶点所连的直线 与 板凳方向的夹角

    if (beta0 > (pi / 2))
        exit;
    end

    %% 求解龙头四顶点坐标式(x1,y1),(x3,y3),  对称后的笛卡尔坐标：(x2,y2),(x4,y4)
    r_1 = sqrt(r(1) ^ 2 + 0.15 ^ 2 + 0.275 ^ 2 - 2 * r(1) * sqrt(0.15 ^ 2 + 0.275 ^ 2) * cos(pi - alpha1(1) - beta0));
    %%%增加到 角度
    theta_1 = asin(sqrt(0.15 ^ 2 + 0.275 ^ 2) * sin(pi - alpha1(1) - beta0) / r_1);
    [x_1, y_1] = pol2cart(theta(1) - theta_1, r_1);

    [front_x, front_y] = pol2cart(theta(1), r(1));
    [behind_x, behind_y] = pol2cart(theta(2), r(2));

    [x_2, y_2] = symmetric_point([front_x, front_y], [behind_x, behind_y], [x_1, y_1]); % % %正确--x(1)

    r_3 = sqrt(r(2) ^ 2 + 0.15 ^ 2 + 0.275 ^ 2 - 2 * r(2) * sqrt(0.15 ^ 2 + 0.275 ^ 2) * cos(pi - alpha2(1) - beta0));
```

```
    theta_3 = asin(sqrt(0.15 ^ 2 + 0.275 ^ 2) * sin(pi - alpha2(1) - beta0) / r_3);
    [x_3, y_3] = pol2cart(theta(2) +theta_3, r_3); % % %正确

    [x_4, y_4] = symmetric_point([front_x, front_y], [behind_x, behind_y], [x_3, y_3]);

    %%求解 任意龙身四个顶点坐标(x1i,y1i),(x2i,y2i),(x3i,y3i),(x4i,y4i)
    x1 = zeros(30, 4);
    y1 = zeros(30, 4); % % %一个板凳四个坐标
    [L, ~] = size(x1); % % % % % TEST OK!

    for i = 2:number_i
        r1 = sqrt(r(i) ^ 2 + 0.15 ^ 2 + 0.275 ^ 2 - 2 * r(i) * sqrt(0.15 ^ 2 + 0.275 ^ 2) *
cos(pi - alpha1(i) - beta0));
        theta1 = asin(sqrt(0.15 ^ 2 + 0.275 ^ 2) * sin(pi - alpha1(i) - beta0) / r1);
        [x1(i - 1, 1), y1(i - 1, 1)] = pol2cart(theta(i) - theta1, r1);

        [front_x2, front_y2] = pol2cart(theta(i), r(i));
        [behind_x2, behind_y2] = pol2cart(theta(i + 1), r(i + 1));
        % 通过 1 对称 2
        [x1(i - 1, 2), y1(i - 1, 2)] = symmetric_point([front_x2, front_y2], [behind_x2,
behind_y2], [x1(i - 1, 1), y1(i - 1, 1)]);

        r3 = sqrt(r(i + 1) ^ 2 + 0.15 ^ 2 + 0.275 ^ 2 - 2 * r(i + 1) * sqrt(0.15 ^ 2 + 0.275
^ 2) * cos(pi - alpha2(i) - beta0));
        theta3 = asin(sqrt(0.15 ^ 2 + 0.275 ^ 2) * sin(pi - alpha2(i) - beta0) / r3);
        [x1(i - 1, 3), y1(i - 1, 3)] = pol2cart(theta(i + 1) + theta3, r3);
        %%%%% 通过 3 对称 4
        [x1(i - 1, 4), y1(i - 1, 4)] = symmetric_point([front_x2, front_y2], [behind_x2,
behind_y2], [x1(i - 1, 3), y1(i - 1, 3)]);
    end

    %%碰撞检测
    count_i = 2; % count_i 为计数器:计数第几块板发生碰撞。

    for j = 2:L % % 因为：龙头始终会与其相邻的（即就是 倒数第二块板）相接，
因此从倒数第三块板开始计数。
        %%%%%%%%%% 龙头 4 个坐标与龙身坐标检测-面积法
        s1 = determin_picture(x_1, y_1, x1(j, :), y1(j, :));
        s2 = determin_picture(x_2, y_2, x1(j, :), y1(j, :));
        % s3 = determin_picture(x_3, y_3, x1(j, :), y1(j, :));
        % s4 = determin_picture(x_4, y_4, x1(j, :), y1(j, :));
        disp([t, s1, s2])

        if s1 <= z || s2 <= z % %|| s3 <= z || s4 <= z
```

```
                flag = 0;
                disp([count_i + 2, s1, s2]);
                % disp([count_i, s1, s2, s3, s4]);
                break;
            end

            count_i = count_i + 1;
        end

        %信号判断
        if flag == 0 % % % % % %已经发生了碰撞，终止循环，得出此时此刻的所有 v 坐
标
            [~, ~, ~, v] = prepare_2(t - 1);
            v = v';
            disp("在 ts 发生了碰撞");
            disp(t);
            break;
        end

end

if flag == 1
        disp('增加时间');
end

[x, y] = pol2cart(theta, r);
function [r0, theta0, w0, v0] = prepare_2(t)
        %%初始化参数
        L1 = 2.86; %龙头长度 单位：m
        L2 = 1.65; %龙身龙尾长度
        s = 0.55; %初始螺线宽距
        b = s / (2 * pi);
        delta_theta_1 = delta(L1, s);
        delta_theta_2 = delta(L2, s);
        dt = 0.1;
        t0 = 0:dt:t;
        t0 = t0';
        len_t = length(t0);
        v = zeros(len_t, 224); %定义线速度,一共 224 个把手
        theta = v;
        Omega = v; %定义角速度
        r = v; %定义极径
        v(:, 1) = 1;
        %%计算第一个点的角速度
```

```
w1_funs = @(t, theta) 1 ./ (b * sqrt(1 + (32 * pi - theta) .^ 2));
theta(:, 1) = 32 * pi - dis_euler(w1_funs, 1, t * 10, 0);
r(:, 1) = b .* theta(:, 1);
Omega(:, 1) = 1 ./ sqrt(r(:, 1) .^ 2 + b ^ 2);

%%计算第二个的角速度
i = 2 * ones(1, 222);

while (abs(theta(i(1), 1) - theta(1, 1)) < delta_theta_1)
    i (1) = i(1) + 1;
end

%刚性杆的约束条件
calpha1 = @(t, theta_2) (L1 ^ 2 + r(t, 1) .^ 2 - (b * (32 * pi - theta_2)) .^ 2) ./ (2 *
L1 .* r(t, 1));
salpha1 = @(t, theta_2)sqrt(1 - calpha1(t, theta_2) .^ 2);
calpha2 = @(t, theta_2)((b * (32 * pi - theta_2)) .^ 2 + L1 ^ 2 - r(t, 1) .^ 2) ./ (2 * b .*
(32 * pi - theta_2) .* L1);
salpha2 = @(t, theta_2)sqrt(1 - calpha2(t, theta_2) .^ 2);
% 沿杆的分速度相等
funs = @(t, theta_2) Omega(t, 1) .* (calpha1(t, theta_2) + theta(t, 1) .* salpha1(t,
theta_2)) ./ (calpha2(t, theta_2) + (32 * pi - theta_2) .* salpha2(t, theta_2));
y0 = 0;
h = 0.1;
theta(i(1):end, 2) = 32 * pi - dis_euler(funs, i(1), t * 10, y0);
r(i(1):end, 2) = b .* theta(i(1):end, 2);

for k = i(1):length(t0)
    Omega(k, 2) = funs(k, 32 * pi - theta(k, 2));
end

v(i(1):end, 2) = Omega(i(1):end, 2) .* sqrt(r(i(1):end, 2) .^ 2 + b ^ 2);
%计算第二个点以后的角速度
flag = 0;

for j = 3:224
    i(j - 1) = i(j - 2);

    while (abs(theta(i(j - 1) + 1, j - 1) - theta(i(j - 2) + 1, j - 1)) < delta_theta_2)
        i (j - 1) = i(j - 1) + 1;

        if i(j - 1) > t * 10
            flag = 1;
            break;
```

```
            end

        end

        if flag == 1
            break;
        end

        calpha1 = @(t, theta_2) (L2 ^ 2 + r(t, j - 1) .^ 2 - (b * (32 * pi - theta_2)) .^ 2) ./
(2 * L2 .* r(t, j - 1));
        salpha1 = @(t, theta_2)sqrt(1 - calpha1(t, theta_2) .^ 2);
        calpha2 = @(t, theta_2)((b * (32 * pi - theta_2)) .^ 2 + L2 ^ 2 - r(t, j - 1) .^ 2) ./
(2 * b .* (32 * pi - theta_2) .* L2);
        salpha2 = @(t, theta_2)sqrt(1 - calpha2(t, theta_2) .^ 2);
        funs = @(t, theta_2) Omega(t, j - 1) .* (calpha1(t, theta_2) + theta(t, j - 1) .*
salpha1(t, theta_2)) ./ (calpha2(t, theta_2) + (32 * pi - theta_2) .* salpha2(t, theta_2));
        y0 = 0;
        h = 0.1;
        theta(i(j - 1):end, j) = 32 * pi - dis_euler(funs, i(j - 1), t * 10, y0);
        r(i(j - 1):end, j) = b .* theta(i(j - 1):end, j);

        for k = i(j - 1):length(t0)
            Omega(k, j) = funs(k, 32 * pi - theta(k, j));
        end

        v(i(j - 1):end, j) = Omega(i(j - 1):end, j) .* sqrt(r(i(j - 1):end, j) .^ 2 + b ^ 2);
    end

    r0 = r(t * 10 + 1, :);
    theta0 = theta(t * 10 + 1, :);
    w0 = Omega(t * 10 + 1, :);
    v0 = v(t * 10 + 1, :);

end
function [alpha1, alpha2] = Palpha(r) % % % % % %  求两个  alpha  角度  / rad
    %%初始化参数
    L1 = 2.86; %龙头长度  单位：m
    L2 = 1.65; %龙身龙尾长度
    alpha1 = zeros(1, 223);
    alpha2 = alpha1;
    calpha1 = alpha1;
    calpha2 = alpha1;

    calpha1(1) = (L1 ^ 2 + r(1) .^ 2 - r(2) .^ 2) ./ (2 * L1 .* r(1));
```

```
    alpha1(1) = acos(calpha1(1));
    calpha2(1) = ((r(2)) .^ 2 + L1 ^ 2 - r(1) .^ 2) ./ (2 * r(2) .* L1);
    alpha2(1) = acos(calpha2(1));

    for i = 2:223
        calpha1(i) = (L2 ^ 2 + r(i) .^ 2 - r(i + 1) .^ 2) ./ (2 * L2 .* r(i));
        alpha1(i) = acos(calpha1(i));
        calpha2(i) = (r(i + 1) .^ 2 + L2 ^ 2 - r(i) .^ 2) ./ (2 * r(i + 1) .* L2);
        alpha2(i) = acos(calpha2(i));
    end

end
function log_form = generated_function(a,v,x)
%GENERATED_FUNCTION
%    LOG_FORM = GENERATED_FUNCTION(A,V,X)

%    This function was generated by the Symbolic Math Toolbox version 9.2.
%    07-Sep-2024 23:03:44

t2 = x.^2;
t3 = t2+1.0;
t4 = sqrt(t3);
log_form = (a.*(log(t4+x)+t4.*x))./(v.*2.0);
```

附录 3 第三问

```
Main.m
clc, clear, close all;
R = 4.5; % 最后的截止半径
b_max = 0.55;
b_min = 0.3;
dt = 1;
i = 1;
%%开始计时
tic;
while abs(b_max -b_min) > 0.01
    middle = (b_max + b_min) / 2;
    dx = middle; %螺距
    a = dx / (2 * pi);
    theta = R * 2 * pi / dx;
```

```
        n = floor(8.8 / dx) + 1;
        y0 = (a / 2) * ((2 * n * pi) * sqrt((2 * n * pi) ^ 2 + 1) + asinh(2 * n
* pi)); %32 pi
        %%%%%得到时间末尾 t
        t_ = y0 - (a / 2) * (theta * sqrt(theta ^ 2 + 1) + asinh(theta));
        t_end = floor(t_);
        my_flag = check(t_end, dx, dt, n); % % %返回 1  没有发生碰撞

        if my_flag == 1 % %中间 b 符合情况
            b_max = middle;
        else
            b_min = middle;
        end

        % max_ = zeros(1, 100);
        % min_ = zeros(1, 100);
        %     max_(i) = b_max;
        %     min_(i) = b_min;
        i = i + 1;
    end
    % 结束计时并获取时间
    elapsedTime = toc;
    disp(['二分法运行时间: ' num2str(elapsedTime) ' s']);
    disp(i);
    disp(dx);

    function [alpha1, alpha2] = Palpha(r) % % % % % % 求两个 alpha 角度 / rad
        %%初始化参数
        L1 = 2.86; %龙头长度 单位：m
        L2 = 1.65; %龙身龙尾长度
        alpha1 = zeros(1, 223);
        alpha2 = alpha1;
        calpha1 = alpha1;
        calpha2 = alpha1;

        calpha1(1) = (L1 ^ 2 + r(1) .^ 2 - r(2) .^ 2) ./ (2 * L1 .* r(1));
        alpha1(1) = acos(calpha1(1));
        calpha2(1) = ((r(2)) .^ 2 + L1 ^ 2 - r(1) .^ 2) ./ (2 * r(2) .* L1);
        alpha2(1) = acos(calpha2(1));

        for i = 2:223
            calpha1(i) = (L2 ^ 2 + r(i) .^ 2 - r(i + 1) .^ 2) ./ (2 * L2 .*
r(i));
            alpha1(i) = acos(calpha1(i));
```

```
            calpha2(i) = (r(i + 1) .^ 2 + L2 ^ 2 - r(i) .^ 2) ./ (2 * r(i + 1) .*
L2);
            alpha2(i) = acos(calpha2(i));
        end

    end


    function [r0, theta0, w0, v0] = prepare_2(t, dx, dt, n)
        %%初始化参数
        L1 = 2.86; %龙头长度 单位：m
        L2 = 1.65; %龙身龙尾长度
        s = dx; %初始螺线宽距
        b = s / (2 * pi);
        delta_theta_1 = delta(L1, s, n);
        delta_theta_2 = delta(L2, s, n);
        t0 = 0:dt:t;
        t0 = t0';
        len_t = length(t0);
        v = zeros(len_t, 224); %定义线速度,一共 224 个把手
        theta = v;
        Omega = v; %定义角速度
        r = v; %定义极径
        v(:, 1) = 1;
        %%计算第一个点的角速度
        w1_funs = @(t, theta) 1 ./ (b * sqrt(1 + (2 * n * pi - theta) .^ 2));
        theta(:, 1) = 2 * n * pi - dis_euler(w1_funs, 1, t / dt, 0, dt);
        r(:, 1) = b .* theta(:, 1);
        Omega(:, 1) = 1 ./ sqrt(r(:, 1) .^ 2 + b ^ 2);

        %%计算第二个的角速度
        i = 2 * ones(1, 222);

        while (abs(theta(i(1), 1) - theta(1, 1)) < delta_theta_1)
            i (1) = i(1) + 1;

            if i(1) > len_t
                break;
            end

        end

        %刚性杆的约束条件
        calpha1 = @(t, theta_2) (L1 ^ 2 + r(t, 1) .^ 2 - (b * (2 * n * pi -
```

```
theta_2)) .^ 2) ./ (2 * L1 .* r(t, 1));
        salpha1 = @(t, theta_2)sqrt(1 - calpha1(t, theta_2) .^ 2);
        calpha2 = @(t, theta_2)((b * (2 * n * pi - theta_2)) .^ 2 + L1 ^ 2 -
r(t, 1) .^ 2) ./ (2 * b .* (2 * n * pi - theta_2) .* L1);
        salpha2 = @(t, theta_2)sqrt(1 - calpha2(t, theta_2) .^ 2);
        % 沿杆的分速度相等
        funs = @(t, theta_2) Omega(t, 1) .* (calpha1(t, theta_2) + theta(t, 1) .*
salpha1(t, theta_2)) ./ (calpha2(t, theta_2) + (2 * n * pi - theta_2) .*
salpha2(t, theta_2));
        y0 = 0;
        h = 0.1;
        theta(i(1):end, 2) = 2 * n * pi - dis_euler(funs, i(1), t / dt, y0, dt);
        r(i(1):end, 2) = b .* theta(i(1):end, 2);

        for k = i(1):length(t0)
            Omega(k, 2) = funs(k, 2 * n * pi - theta(k, 2));
        end

        v(i(1):end, 2) = Omega(i(1):end, 2) .* sqrt(r(i(1):end, 2) .^ 2 + b ^
2);
        %计算第二个点以后的角速度
        flag = 0;

        for j = 3:224
            i(j - 1) = i(j - 2);

            while (abs(theta(i(j - 1) + 1, j - 1) - theta(i(j - 2) + 1, j - 1))
< delta_theta_2)
                i (j - 1) = i(j - 1) + 1;

                if i(j - 1) > (t / dt)
                    flag = 1;
                    break;
                end

            end

            if flag == 1
                break;
            end

            calpha1 = @(t, theta_2) (L2 ^ 2 + r(t, j - 1) .^ 2 - (b * (2 * n *
pi - theta_2)) .^ 2) ./ (2 * L2 .* r(t, j - 1));
            salpha1 = @(t, theta_2)sqrt(1 - calpha1(t, theta_2) .^ 2);
```

```
            calpha2 = @(t, theta_2)((b * (2 * n * pi - theta_2)) .^ 2 + L2 ^ 2 -
r(t, j - 1) .^ 2) ./ (2 * b .* (2 * n * pi - theta_2) .* L2);
            salpha2 = @(t, theta_2)sqrt(1 - calpha2(t, theta_2) .^ 2);
            funs = @(t, theta_2) Omega(t, j - 1) .* (calpha1(t, theta_2) +
theta(t, j - 1) .* salpha1(t, theta_2)) ./ (calpha2(t, theta_2) + (2 * n * pi -
theta_2) .* salpha2(t, theta_2));
            y0 = 0;
            h = 0.1;
            theta(i(j - 1):end, j) = 2 * n * pi - dis_euler(funs, i(j - 1), t /
dt, y0, dt);
            r(i(j - 1):end, j) = b .* theta(i(j - 1):end, j);

            for k = i(j - 1):length(t0)
                Omega(k, j) = funs(k, 2 * n * pi - theta(k, j));
            end

            v(i(j - 1):end, j) = Omega(i(j - 1):end, j) .* sqrt(r(i(j - 1):end,
j) .^ 2 + b ^ 2);
        end

        r0 = r(t / dt + 1, :);
        theta0 = theta(t / dt + 1, :);
        w0 = Omega(t / dt + 1, :);
        v0 = v(t / dt + 1, :);
    end

    function [x_sym, y_sym] = symmetric_point(A, B, P)
        %A=[x,y]:为前把手的笛卡尔 坐标点
        %B=[x,y]:为后把手的笛卡尔 坐标点
        % p 为需要对称的点的坐标
        % 输入: A, B 为两个点，定义直线；P 为给定点 [x, y]
        % 输出: P 关于直线的对称点 [x_sym, y_sym]

        % 提取直线的两个点坐标
        x1 = A(1);
        y1 = A(2);
        x2 = B(1);
        y2 = B(2);

        % 提取给定点 P 的坐标，靠近 原点 方向 弧线内侧上的点
        x0 = P(1);
        y0 = P(2);

        % 判断是否为竖直线
```

```
        if x1 == x2
            % 如果是竖直线，对称点在 x 轴方向镜像
            x_sym = 2 * x1 - x0;
            y_sym = y0; % y 值不变
        else
            % 计算直线的斜率和截距
            m = (y2 - y1) / (x2 - x1);
            b = y1 - m * x1;

            % 计算垂足点 T 的坐标
            xt = (x0 + m * (y0 - b)) / (1 + m ^ 2);

            yt = (m * (x0 + m * (y0 - b))) / (1 + m ^ 2) + b;

            % 计算对称点 P'的坐标
            x_sym = 2 * xt - x0;
            y_sym = 2 * yt - y0;
        end

    end



    clc; clear; close all;

    [r0, theta0, Omega, v0] = prepare_2(412);

    [x, y] = pol2cart(theta0, r0);
    x = x';
    y = y';
    v0 = v0';
    scatter(x, y);
    % %%%%%%%%%%论文当中的表
    step = [1 51 101 151 201] + 1; %下标
    paper_matrix1 = x(step); %x
    paper_matrix2 = y(step); %y
    paper_matrix3 = v0(step);
    combined_matrix = [paper_matrix1, paper_matrix2, paper_matrix3];
    header = {'x', 'y', 'v'};
    Table1 = array2table(combined_matrix, 'VariableNames', header);
    % 将数据写入 Excel 文件
    filename1 = 'F:\EXCEL\que2_position_v.xlsx';
    % filename2 = 'F:\EXCEL\speed.xlsx';
    writetable(Table1, filename1);
```

```
disp('paper 速度 and 位置 已写入!');


function y = dis_euler(f, a, b, y0,dt) % % % % % % % % 欧拉法
    s = b - a + 1;
    Y = zeros(1, s + 1);
    Y(1) = y0;

    for k = 1:s
        y_pred = Y(k) + dt * f(a + k - 1, Y(k));

        % 修正步骤（使用预测值进行修正）
        Y(k + 1) = Y(k) + (dt / 2) * (f(a + k - 1, Y(k)) + f(a + k, y_pred));
        % Y(k + 1) = Y(k) + h * f(a + k - 1, Y(k));
    end

    y = Y'; %转置
end


function s = determin_picture(x_1, y_1, x1, y1)
    P = [x_1 y_1]; % 要检测的点即就是论文中的 p 点
    %%%%% 发生碰撞的板的 4 个坐标
    %%%%% 检测面积
    s1_x = [x_1 x1(1) x1(2)];
    s1_y = [y_1 y1(1) y1(2)];
    s1 = polyarea(s1_x, s1_y);

    s2_x = [x_1 x1(1) x1(3)];
    s2_y = [y_1 y1(1) y1(3)];
    s2 = polyarea(s2_x, s2_y);

    s3_x = [x_1 x1(3) x1(4)];
    s3_y = [y_1 y1(3) y1(4)];
    s3 = polyarea(s3_x, s3_y);

    s4_x = [x_1 x1(2) x1(4)];
    s4_y = [y_1 y1(2) y1(4)];
    s4 = polyarea(s4_x, s4_y);

    s5 = 0.66; % m²
    s = s4 + s3 + s2 + s1 -s5;
end
```

```
    function theta_result = delta(L, s, n)
        %用于求解两端点径向方向间初始夹角
        %输入:L 杆长
        %s 螺距宽
        %输出:两端点径向方向间初始夹角
        r = s * n;
        %     syms theta
        % 设置 fsolve 的选项
        options = optimoptions('fsolve', ...
            'TolX', 1e-12, ...                      % 变量变化精度
            'TolFun', 1e-12, ...
             'Display', 'final', ...    % 函数值变化精度
            'MaxIterations', 1000, ...              % 最大迭代次数
            'MaxFunctionEvaluations', 5000, ...     % 最大函数评估次数
            'Algorithm', 'levenberg-marquardt');
        f = @(theta) cos(theta) - ((s * (2 * n * pi - theta) / (2 * pi)) ^ 2 +
r ^ 2 - L ^ 2) / (2 * (s * (2 * n * pi - theta) / (2 * pi)) * r);

        % 使用 vpasolve 求解
        theta_result = fsolve(f, 0.5);
    end



    function flag = check(t_end, dx, dt, n) %t 为到达 theta 角度所需的时间，dx 为螺
距
        flag = 1; % % % % % flag=1,没有发生碰撞，代表可以继续二分，flag=0，发生了碰
撞
        z_k = 0.025; %板凳面积的占比系数
        z = z_k * 0.66; %误差控制系数  m²   板凳面积的 3 %
        number_i = 31; %从龙头紧接的板子（即就是 倒数第二块板子） 开始往后遍历个数

        t_begin = floor(t_end / 2);

        if t_end <= t_begin
            disp("t_end < = t_begin");
            % exit;
        end

        for t = t_begin:dt:t_end
            %准备工作
            [r, theta, ~, ~] = prepare_2(t, dx, dt, n);
            %%%%%% alpha1 当作前把手时 ：  板方向与极矢方向的夹角
            %%%%%% alpha2 当作后把手时 ：  板方向与极矢方向的夹角
            [alpha1, alpha2] = Palpha(r);
```

```
            %特定角度
            beta0 = atan(15/27.5); % %板凳上任意一把手 坐标 与 右 or 左 顶点所连的
直线 与 板凳方向的夹角

            if (beta0 > (pi / 2))
                disp('beta0 > (pi / 2)');
                % exit;
            end

            %% 求解龙头四顶点坐标式(x1,y1),(x3,y3),    对称后的笛卡尔坐标：
(x2,y2),(x4,y4)
            r_1 = sqrt(r(1) ^ 2 + 0.15 ^ 2 + 0.275 ^ 2 - 2 * r(1) * sqrt(0.15 ^
2 + 0.275 ^ 2) * cos(pi - alpha1(1) - beta0));
            %%%增加到 角度
            theta_1 = asin(sqrt(0.15 ^ 2 + 0.275 ^ 2) * sin(pi - alpha1(1) -
beta0) / r_1);
            [x_1, y_1] = pol2cart(theta(1) - theta_1, r_1);

            [front_x, front_y] = pol2cart(theta(1), r(1));
            [behind_x, behind_y] = pol2cart(theta(2), r(2));

            [x_2,   y_2]   =   symmetric_point([front_x,   front_y],   [behind_x,
behind_y], [x_1, y_1]); % % %正确--x(1)

            %          r_3 = sqrt(r(2) ^ 2 + 0.15 ^ 2 + 0.275 ^ 2 - 2 * r(2) *
sqrt(0.15 ^ 2 + 0.275 ^ 2) * cos(pi - alpha2(1) - beta0));
            %           theta_3 = asin(sqrt(0.15 ^ 2 + 0.275 ^ 2) * sin(pi -
alpha2(1) - beta0) / r_3);
            %         [x_3, y_3] = pol2cart(theta(2) +theta_3, r_3); % % %正确

            %         [x_4, y_4] = symmetric_point([front_x, front_y], [behind_x,
behind_y], [x_3, y_3]);

            %%求解 任意龙身四个顶点坐标(x1i,y1i),(x2i,y2i),(x3i,y3i),(x4i,y4i)
            x1 = zeros(30, 4);
            y1 = zeros(30, 4); % % %一个板凳四个坐标
            [L, ~] = size(x1); % % % % % TEST OK!

            for i = 2:number_i
                r1 = sqrt(r(i) ^ 2 + 0.15 ^ 2 + 0.275 ^ 2 - 2 * r(i) * sqrt(0.15
^ 2 + 0.275 ^ 2) * cos(pi - alpha1(i) - beta0));
                theta1 = asin(sqrt(0.15 ^ 2 + 0.275 ^ 2) * sin(pi - alpha1(i) -
beta0) / r1);
                [x1(i - 1, 1), y1(i - 1, 1)] = pol2cart(theta(i) - theta1, r1);
```

```
                [front_x2, front_y2] = pol2cart(theta(i), r(i));
                [behind_x2, behind_y2] = pol2cart(theta(i + 1), r(i + 1));
                % 通过 1 对称 2
                [x1(i - 1, 2), y1(i - 1, 2)] = symmetric_point([front_x2,
front_y2], [behind_x2, behind_y2], [x1(i - 1, 1), y1(i - 1, 1)]);

                r3 = sqrt(r(i + 1) ^ 2 + 0.15 ^ 2 + 0.275 ^ 2 - 2 * r(i + 1) *
sqrt(0.15 ^ 2 + 0.275 ^ 2) * cos(pi - alpha2(i) - beta0));
                theta3 = asin(sqrt(0.15 ^ 2 + 0.275 ^ 2) * sin(pi - alpha2(i) -
beta0) / r3);
                [x1(i - 1, 3), y1(i - 1, 3)] = pol2cart(theta(i + 1) + theta3,
r3);
                %%%%% 通过 3 对称 4
                [x1(i - 1, 4), y1(i - 1, 4)] = symmetric_point([front_x2,
front_y2], [behind_x2, behind_y2], [x1(i - 1, 3), y1(i - 1, 3)]);
            end

            %%碰撞检测
            count_i = 2; % count_i 为计数器:计数第几块板发生碰撞。

            for j = 2:L % % 因为：龙头始终会与其相邻的（即就是 倒数第二块板）相接，因
此从倒数第三块板开始计数。
                %%%%%%%%%% 龙头 4 个坐标与龙身坐标检测-面积法
                s1 = determin_picture(x_1, y_1, x1(j, :), y1(j, :));
                s2 = determin_picture(x_2, y_2, x1(j, :), y1(j, :));
                % s3 = determin_picture(x_3, y_3, x1(j, :), y1(j, :));
                % s4 = determin_picture(x_4, y_4, x1(j, :), y1(j, :));
                % disp([t, s1, s2])

                if s1 <= z || s2 <= z % %|| s3 <= z || s4 <= z
                    flag = 0; % % % % 已发生碰撞
                    disp(" 已经发生碰撞!");

                    disp([count_i + 2, s1, s2]);
                    % disp([count_i, s1, s2, s3, s4]);
                    break;
                end

                count_i = count_i + 1;
            end

            %信号判断
            if flag == 0 % % % % %已经发生了碰撞，终止循环，得出此时此刻的所有 v 坐标
```

```
                % [~, ~, ~, v] = prepare_2(t - 1);
                % v = v';
                disp("在 ts 发生了碰撞");
                disp(t);
                break;
            end

        end

        % [x, y] = pol2cart(theta, r);
    end
```

附录 4 第四问

```
clc; clear; close all;
format long g
dt = 1;
dx = 1.7; %m
v0 = 1;
i = 1;
a = dx / (2 * pi);
r_max = 4.5;
r_min = (2.86 * 3) / 2; % % % % % %需对 R 进行特殊额条件判断
%%%%%%%二分搜索->找最小的 r
while r_max - r_min > 0.00001
    middle = (r_max + r_min) / 2;
    R = middle; % % % % %将 middle  赋值给中间区域

    theta = R * 2 * pi / dx;
    n = round(8.8 / dx);
    y0 = (a / 2) * ((2 * n * pi) * sqrt((2 * n * pi) ^ 2 + 1) + asinh(2 * n * pi)); %32 pi
    %%%%%%得到时间末尾 t
    t_ = y0 - (a / 2) * (theta * sqrt(theta ^ 2 + 1) + asinh(theta));
    t_end = floor(t_);
    my_flag = check(t_end, dx, dt, n); % % %返回 1    没有发生碰撞

    if my_flag == 1 % %中间  r 符合情况
```

```
            r_max = middle;
        else
            r_min = middle;
        end

        i = i + 1;
end

disp(i);
disp(R);
disp(r_min);
disp(r_min);

function y = dis_euler(f, a, b, y0,dt) % % % % % % %    欧拉法
        s = b - a + 1;
        Y = zeros(1, s + 1);
        Y(1) = y0;

        for k = 1:s
            y_pred = Y(k) + dt * f(a + k - 1, Y(k));

            % 修正步骤（使用预测值进行修正）
            Y(k + 1) = Y(k) + (dt / 2) * (f(a + k - 1, Y(k)) + f(a + k, y_pred));
            % Y(k + 1) = Y(k) + h * f(a + k - 1, Y(k));
        end

        y = Y'; %转置
end

function [alpha1, alpha2] = Palpha(r) % % % % % % 求两个 alpha 角度 / rad
        %%初始化参数
        L1 = 2.86; %龙头长度 单位：m
        L2 = 1.65; %龙身龙尾长度
        alpha1 = zeros(1, 223);
        alpha2 = alpha1;
        calpha1 = alpha1;
        calpha2 = alpha1;

        calpha1(1) = (L1 ^ 2 + r(1) .^ 2 - r(2) .^ 2) ./ (2 * L1 .* r(1));
        alpha1(1) = acos(calpha1(1));
        calpha2(1) = ((r(2)) .^ 2 + L1 ^ 2 - r(1) .^ 2) ./ (2 * r(2) .* L1);
        alpha2(1) = acos(calpha2(1));

        for i = 2:223
```

```
        calpha1(i) = (L2 ^ 2 + r(i) .^ 2 - r(i + 1) .^ 2) ./ (2 * L2 .* r(i));
        alpha1(i) = acos(calpha1(i));
        calpha2(i) = (r(i + 1) .^ 2 + L2 ^ 2 - r(i) .^ 2) ./ (2 * r(i + 1) .* L2);
        alpha2(i) = acos(calpha2(i));
    end

end
function [r0, theta0, w0, v0] = prepare_2(t, dx, dt, n)
    %%初始化参数
    L1 = 2.86; %龙头长度 单位：m
    L2 = 1.65; %龙身龙尾长度
    s = dx; %初始螺线宽距
    b = s / (2 * pi);
    delta_theta_1 = delta(L1, s, n);
    delta_theta_2 = delta(L2, s, n);
    t0 = 0:dt:t;
    t0 = t0';
    len_t = length(t0);
    v = zeros(len_t, 224); %定义线速度,一共 224 个把手
    theta = v;
    Omega = v; %定义角速度
    r = v; %定义极径
    v(:, 1) = 1;
    %%计算第一个点的角速度
    w1_funs = @(t, theta) 1 ./ (b * sqrt(1 + (2 * n * pi - theta) .^ 2));
    theta(:, 1) = 2 * n * pi - dis_euler(w1_funs, 1, t / dt, 0, dt);
    r(:, 1) = b .* theta(:, 1);
    Omega(:, 1) = 1 ./ sqrt(r(:, 1) .^ 2 + b ^ 2);

    %%计算第二个的角速度
    i = 2 * ones(1, 222);

    while (abs(theta(i(1), 1) - theta(1, 1)) < delta_theta_1)
        i (1) = i(1) + 1;

        if i(1) > len_t
            break;
        end

    end

    %刚性杆的约束条件
    calpha1 = @(t, theta_2) (L1 ^ 2 + r(t, 1) .^ 2 - (b * (2 * n * pi - theta_2)) .^ 2) ./ (2 *
L1 .* r(t, 1));
```

```
    salpha1 = @(t, theta_2)sqrt(1 - calpha1(t, theta_2) .^ 2);
    calpha2 = @(t, theta_2)((b * (2 * n * pi - theta_2)) .^ 2 + L1 ^ 2 - r(t, 1) .^ 2) ./ (2 *
b .* (2 * n * pi - theta_2) .* L1);
    salpha2 = @(t, theta_2)sqrt(1 - calpha2(t, theta_2) .^ 2);
    % 沿杆的分速度相等
    funs = @(t, theta_2) Omega(t, 1) .* (calpha1(t, theta_2) + theta(t, 1) .* salpha1(t,
theta_2)) ./ (calpha2(t, theta_2) + (2 * n * pi - theta_2) .* salpha2(t, theta_2));
    y0 = 0;
    h = 0.1;
    theta(i(1):end, 2) = 2 * n * pi - dis_euler(funs, i(1), t / dt, y0, dt);
    r(i(1):end, 2) = b .* theta(i(1):end, 2);

    for k = i(1):length(t0)
        Omega(k, 2) = funs(k, 2 * n * pi - theta(k, 2));
    end

    v(i(1):end, 2) = Omega(i(1):end, 2) .* sqrt(r(i(1):end, 2) .^ 2 + b ^ 2);
    %计算第二个点以后的角速度
    flag = 0;

    for j = 3:224
        i(j - 1) = i(j - 2);

        while (abs(theta(i(j - 1) + 1, j - 1) - theta(i(j - 2) + 1, j - 1)) < delta_theta_2)
            i (j - 1) = i(j - 1) + 1;

            if i(j - 1) > (t / dt)
                flag = 1;
                break;
            end

        end

        if flag == 1
            break;
        end

        calpha1 = @(t, theta_2) (L2 ^ 2 + r(t, j - 1) .^ 2 - (b * (2 * n * pi - theta_2)) .^
2) ./ (2 * L2 .* r(t, j - 1));
        salpha1 = @(t, theta_2)sqrt(1 - calpha1(t, theta_2) .^ 2);
        calpha2 = @(t, theta_2)((b * (2 * n * pi - theta_2)) .^ 2 + L2 ^ 2 - r(t, j - 1) .^
2) ./ (2 * b .* (2 * n * pi - theta_2) .* L2);
        salpha2 = @(t, theta_2)sqrt(1 - calpha2(t, theta_2) .^ 2);
        funs = @(t, theta_2) Omega(t, j - 1) .* (calpha1(t, theta_2) + theta(t, j - 1) .*
```

```
salpha1(t, theta_2)) ./ (calpha2(t, theta_2) + (2 * n * pi - theta_2) .* salpha2(t, theta_2));
            y0 = 0;
            h = 0.1;
            theta(i(j - 1):end, j) = 2 * n * pi - dis_euler(funs, i(j - 1), t / dt, y0, dt);
            r(i(j - 1):end, j) = b .* theta(i(j - 1):end, j);

            for k = i(j - 1):length(t0)
                Omega(k, j) = funs(k, 2 * n * pi - theta(k, j));
            end

            v(i(j - 1):end, j) = Omega(i(j - 1):end, j) .* sqrt(r(i(j - 1):end, j) .^ 2 + b ^ 2);
        end

        r0 = r(t / dt + 1, :);
        theta0 = theta(t / dt + 1, :);
        w0 = Omega(t / dt + 1, :);
        v0 = v(t / dt + 1, :);
end
function [r0, theta0, w0, v0] = prepare_3(t, dx, dt, n)
        %%初始化参数
        L1 = 2.86; %龙头长度 单位：m
        L2 = 1.65; %龙身龙尾长度
        s = dx; %初始螺线宽距
        b = s / (2 * pi);
        delta_theta_1 = delta(L1, s, n);
        delta_theta_2 = delta(L2, s, n);
        t0 = 0:dt:t;
        t0 = t0';
        len_t = length(t0);
        v = zeros(len_t, 224); %定义线速度,一共 224 个把手
        theta = v;
        Omega = v; %定义角速度
        r = v; %定义极径
        v(:, 1) = 1;
        %%计算第一个点的角速度
        w1_funs = @(t, theta) 1 ./ (b * sqrt(1 + (2 * n * pi - theta) .^ 2));
        theta(:, 1) = 2 * n * pi - dis_euler(w1_funs, 1, t / dt, 0, dt);
        r(:, 1) = b .* theta(:, 1);
        Omega(:, 1) = 1 ./ sqrt(r(:, 1) .^ 2 + b ^ 2);

        %%计算第二个的角速度
        i = 2 * ones(1, 222);

        while (abs(theta(i(1), 1) - theta(1, 1)) < delta_theta_1)
```

```
        i (1) = i(1) + 1;

        if i(1) > len_t
            break;
        end

    end

    %刚性杆的约束条件
    calpha1 = @(t, theta_2) (L1 ^ 2 + r(t, 1) .^ 2 - (b * (2 * n * pi - theta_2)) .^ 2) ./ (2 *
L1 .* r(t, 1));
    salpha1 = @(t, theta_2)sqrt(1 - calpha1(t, theta_2) .^ 2);
    calpha2 = @(t, theta_2)((b * (2 * n * pi - theta_2)) .^ 2 + L1 ^ 2 - r(t, 1) .^ 2) ./ (2 *
b .* (2 * n * pi - theta_2) .* L1);
    salpha2 = @(t, theta_2)sqrt(1 - calpha2(t, theta_2) .^ 2);
    % 沿杆的分速度相等
    funs = @(t, theta_2) Omega(t, 1) .* (calpha1(t, theta_2) + theta(t, 1) .* salpha1(t,
theta_2)) ./ (calpha2(t, theta_2) + (2 * n * pi - theta_2) .* salpha2(t, theta_2));
    y0 = 0;
    h = 0.1;
    theta(i(1):end, 2) = 2 * n * pi - dis_euler(funs, i(1), t / dt, y0, dt);
    r(i(1):end, 2) = b .* theta(i(1):end, 2);

    for k = i(1):length(t0)
        Omega(k, 2) = funs(k, 2 * n * pi - theta(k, 2));
    end

    v(i(1):end, 2) = Omega(i(1):end, 2) .* sqrt(r(i(1):end, 2) .^ 2 + b ^ 2);
    %计算第二个点以后的角速度
    flag = 0;

    for j = 3:224
        i(j - 1) = i(j - 2);

        while (abs(theta(i(j - 1) + 1, j - 1) - theta(i(j - 2) + 1, j - 1)) < delta_theta_2)
            i (j - 1) = i(j - 1) + 1;

            if i(j - 1) > (t / dt)
                flag = 1;
                break;
            end

        end
```

```
            if flag == 1
                break;
            end

            calpha1 = @(t, theta_2) (L2 ^ 2 + r(t, j - 1) .^ 2 - (b * (2 * n * pi - theta_2)) .^
2) ./ (2 * L2 .* r(t, j - 1));
            salpha1 = @(t, theta_2)sqrt(1 - calpha1(t, theta_2) .^ 2);
            calpha2 = @(t, theta_2)((b * (2 * n * pi - theta_2)) .^ 2 + L2 ^ 2 - r(t, j - 1) .^
2) ./ (2 * b .* (2 * n * pi - theta_2) .* L2);
            salpha2 = @(t, theta_2)sqrt(1 - calpha2(t, theta_2) .^ 2);
            funs = @(t, theta_2) Omega(t, j - 1) .* (calpha1(t, theta_2) + theta(t, j - 1) .*
salpha1(t, theta_2)) ./ (calpha2(t, theta_2) + (2 * n * pi - theta_2) .* salpha2(t, theta_2));
            y0 = 0;
            h = 0.1;
            theta(i(j - 1):end, j) = 2 * n * pi - dis_euler(funs, i(j - 1), t / dt, y0, dt);
            r(i(j - 1):end, j) = b .* theta(i(j - 1):end, j);

            for k = i(j - 1):length(t0)
                Omega(k, j) = funs(k, 2 * n * pi - theta(k, j));
            end

            v(i(j - 1):end, j) = Omega(i(j - 1):end, j) .* sqrt(r(i(j - 1):end, j) .^ 2 + b ^ 2);
        end

        r0 = r;
        theta0 = theta;
        w0 = Omega;
        v0 = v;
end
function [x_sym, y_sym] = symmetric_point(A, B, P)
        %A=[x,y]:为前把手的笛卡尔 坐标点
        %B=[x,y]:为后把手的笛卡尔 坐标点
        % p 为需要对称的点的坐标
        % 输入: A, B 为两个点，定义直线；P 为给定点 [x, y]
        % 输出: P 关于直线的对称点 [x_sym, y_sym]

        % 提取直线的两个点坐标
        x1 = A(1);
        y1 = A(2);
        x2 = B(1);
        y2 = B(2);

        % 提取给定点 P 的坐标，靠近 原点 方向 弧线内侧上的点
        x0 = P(1);
```

```
    y0 = P(2);

    % 判断是否为竖直线
    if x1 == x2
        % 如果是竖直线，对称点在 x 轴方向镜像
        x_sym = 2 * x1 - x0;
        y_sym = y0; % y 值不变
    else
        % 计算直线的斜率和截距
        m = (y2 - y1) / (x2 - x1);
        b = y1 - m * x1;

        % 计算垂足点 T 的坐标
        xt = (x0 + m * (y0 - b)) / (1 + m ^ 2);

        yt = (m * (x0 + m * (y0 - b))) / (1 + m ^ 2) + b;

        % 计算对称点 P'的坐标
        x_sym = 2 * xt - x0;
        y_sym = 2 * yt - y0;
    end

end
function flag = check(t_end, dx, dt, n) %t 为到达 theta 角度所需的时间，dx 为螺距
    flag = 1; % % % % % flag=1,没有发生碰撞，代表可以继续二分，flag=0，发生了碰撞
    z_k = 0.025; %板凳面积的占比系数
    z = z_k * 0.66; %误差控制系数   m²     板凳面积的 3 %
    number_i = 31; %从龙头紧接的板子（即就是 倒数第二块板子） 开始往后遍历个数

    t_begin = floor(t_end / 2);

    if t_end <= t_begin
        disp("t_end < = t_begin");
        % exit;
    end

    for t = t_begin:dt:t_end
        %准备工作
        [r, theta, ~, ~] = prepare_2(t, dx, dt, n);
        %%%%%% alpha1 当作前把手时 ： 板方向与极矢方向的夹角
        %%%%%% alpha2 当作后把手时 ： 板方向与极矢方向的夹角
        [alpha1, alpha2] = Palpha(r);
```

```
        %特定角度
        beta0 = atan(15/27.5); % %板凳上任意一把手 坐标 与 右 or 左 顶点所连的直线 与 板凳方向的夹角

        if (beta0 > (pi / 2))
            disp('beta0 > (pi / 2)');
            % exit;
        end

        %%求解龙头四顶点坐标式(x1,y1),(x3,y3),  对称后的笛卡尔坐标：(x2,y2),(x4,y4)
        r_1 = sqrt(r(1) ^ 2 + 0.15 ^ 2 + 0.275 ^ 2 - 2 * r(1) * sqrt(0.15 ^ 2 + 0.275 ^ 2) * cos(pi - alpha1(1) - beta0));
        %%%增加到 角度
        theta_1 = asin(sqrt(0.15 ^ 2 + 0.275 ^ 2) * sin(pi - alpha1(1) - beta0) / r_1);
        [x_1, y_1] = pol2cart(theta(1) - theta_1, r_1);

        [front_x, front_y] = pol2cart(theta(1), r(1));
        [behind_x, behind_y] = pol2cart(theta(2), r(2));

        [x_2, y_2] = symmetric_point([front_x, front_y], [behind_x, behind_y], [x_1, y_1]); % % %正确--x(1)

        %           r_3 = sqrt(r(2) ^ 2 + 0.15 ^ 2 + 0.275 ^ 2 - 2 * r(2) * sqrt(0.15 ^ 2 + 0.275 ^ 2) * cos(pi - alpha2(1) - beta0));
        %           theta_3 = asin(sqrt(0.15 ^ 2 + 0.275 ^ 2) * sin(pi - alpha2(1) - beta0) / r_3);
        %           [x_3, y_3] = pol2cart(theta(2) +theta_3, r_3); % % %正确

        %             [x_4, y_4] = symmetric_point([front_x, front_y], [behind_x, behind_y], [x_3, y_3]);

        %%求解 任意龙身四个顶点坐标(x1i,y1i),(x2i,y2i),(x3i,y3i),(x4i,y4i)
        x1 = zeros(30, 4);
        y1 = zeros(30, 4); % % %一个板凳四个坐标
        [L, ~] = size(x1); % % % % % TEST OK!

        for i = 2:number_i
            r1 = sqrt(r(i) ^ 2 + 0.15 ^ 2 + 0.275 ^ 2 - 2 * r(i) * sqrt(0.15 ^ 2 + 0.275 ^ 2) * cos(pi - alpha1(i) - beta0));
            theta1 = asin(sqrt(0.15 ^ 2 + 0.275 ^ 2) * sin(pi - alpha1(i) - beta0) / r1);
            [x1(i - 1, 1), y1(i - 1, 1)] = pol2cart(theta(i) - theta1, r1);

            [front_x2, front_y2] = pol2cart(theta(i), r(i));
```

```
            [behind_x2, behind_y2] = pol2cart(theta(i + 1), r(i + 1));
            %  通过 1 对称 2
            [x1(i - 1, 2), y1(i - 1, 2)] = symmetric_point([front_x2, front_y2], [behind_x2, behind_y2], [x1(i - 1, 1), y1(i - 1, 1)]);

            r3 = sqrt(r(i + 1) ^ 2 + 0.15 ^ 2 + 0.275 ^ 2 - 2 * r(i + 1) * sqrt(0.15 ^ 2 + 0.275 ^ 2) * cos(pi - alpha2(i) - beta0));
            theta3 = asin(sqrt(0.15 ^ 2 + 0.275 ^ 2) * sin(pi - alpha2(i) - beta0) / r3);
            [x1(i - 1, 3), y1(i - 1, 3)] = pol2cart(theta(i + 1) + theta3, r3);
            %%%%%  通过 3 对称 4
            [x1(i - 1, 4), y1(i - 1, 4)] = symmetric_point([front_x2, front_y2], [behind_x2, behind_y2], [x1(i - 1, 3), y1(i - 1, 3)]);
        end

        %%碰撞检测
        count_i = 2; % count_i 为计数器:计数第几块板发生碰撞。

        for j = 2:L %%  因为：龙头始终会与其相邻的（即就是 倒数第二块板）相接，因此从倒数第三块板开始计数。
            %%%%%%%%%%%  龙头 4 个坐标与龙身坐标检测-面积法
            s1 = determin_picture(x_1, y_1, x1(j, :), y1(j, :));
            s2 = determin_picture(x_2, y_2, x1(j, :), y1(j, :));
            % s3 = determin_picture(x_3, y_3, x1(j, :), y1(j, :));
            % s4 = determin_picture(x_4, y_4, x1(j, :), y1(j, :));
            % disp([t, s1, s2])

            if s1 <= z || s2 <= z % %|| s3 <= z || s4 <= z
                flag = 0; % % % %  已发生碰撞
                disp("  已经发生碰撞!");

                disp([count_i + 2, s1, s2]);
                % disp([count_i, s1, s2, s3, s4]);
                break;
            end

            count_i = count_i + 1;
        end

        %信号判断
        if flag == 0 % % % % %已经发生了碰撞，终止循环，得出此时此刻的所有 v  坐标
            % [~, ~, ~, v] = prepare_2(t - 1);
            % v = v';
            disp("在 ts  发生了碰撞");
```

```
                disp(t);
                break;
            end

        end

        % [x, y] = pol2cart(theta, r);
end
function theta_result = delta(L, s, n)
        %用于求解两端点径向方向间初始夹角
        %输入:L 杆长
        %s 螺距宽
        %输出:两端点径向方向间初始夹角
        r = s * n;
        %         syms theta
        % 设置 fsolve 的选项
        options = optimoptions('fsolve', ...
            'TolX', 1e-12, ...                                  % 变量变化精度
            'TolFun', 1e-12, ...
             'Display', 'final', ...        % 函数值变化精度
            'MaxIterations', 1000, ...                          % 最大迭代次数
            'MaxFunctionEvaluations', 5000, ...         % 最大函数评估次数
            'Algorithm', 'levenberg-marquardt');
        f = @(theta) cos(theta) - ((s * (2 * n * pi - theta) / (2 * pi)) ^ 2 + r ^ 2 - L ^ 2) / (2 * (s
* (2 * n * pi - theta) / (2 * pi)) * r);

        % 使用 vpasolve 求解
        theta_result = fsolve(f, 0.5);
end
function s = determin_picture(x_1, y_1, x1, y1)
        P = [x_1 y_1]; % 要检测的点即就是论文中的 p 点
        %%%%% 发生碰撞的板的 4 个坐标
        %%%%% 检测面积
        s1_x = [x_1 x1(1) x1(2)];
        s1_y = [y_1 y1(1) y1(2)];
        s1 = polyarea(s1_x, s1_y);

        s2_x = [x_1 x1(1) x1(3)];
        s2_y = [y_1 y1(1) y1(3)];
        s2 = polyarea(s2_x, s2_y);

        s3_x = [x_1 x1(3) x1(4)];
        s3_y = [y_1 y1(3) y1(4)];
        s3 = polyarea(s3_x, s3_y);
```

```
    s4_x = [x_1 x1(2) x1(4)];
    s4_y = [y_1 y1(2) y1(4)];
    s4 = polyarea(s4_x, s4_y);

    s5 = 0.66; % m²
    s = s4 + s3 + s2 + s1 -s5;
end
clc; clear; close all;
format long g
dt = 1;
dx = 1.7; %m
v0 = 1;
a = dx / (2 * pi);
R = (2.83 * 3) / 2;
n = round(8.8 / dx);
theta = R * 2 * pi / dx;
y0 = (a / 2) * ((2 * n * pi) * sqrt((2 * n * pi) ^ 2 + 1) + asinh(2 * n * pi));
t_ = y0 - (a / 2) * (theta * sqrt(theta ^ 2 + 1) + asinh(theta));
t_end = floor(t_);
[r0, theta0, Omega, v_0] = prepare_3(t_end, dx, dt, n);

[x, y] = pol2cart(theta0, r0);
% x = x';
% y = y';
% v_0 = v_0';
% scatter(x, y);

% %%%%%%%%%%%%论文当中的表
% step = [1 51 101 151 201] + 1; %下标
% paper_matrix1 = x(step); %#ok<> %x
% paper_matrix2 = y(step); %y
% paper_matrix3 = v_0(step);
% combined_matrix = [paper_matrix1, paper_matrix2, paper_matrix3];
% header = {'x', 'y', 'v'};
% Table1 = array2table(combined_matrix, 'VariableNames', header);
% % 将数据写入 Excel 文件
% filename1 = 'F:\EXCEL\que2_position_v.xlsx';
% % filename2 = 'F:\EXCEL\speed.xlsx';
% writetable(Table1, filename1);
% disp('paper 速度 and 位置 已写入!');

% step = 0:1:100;
% step_index = step / dt +1;
```

```
%% 对 x, y, v 进行采样
% x_1 = x(:, step_index); % 按时间间隔采样 x
% y_1 = y(:, step_index);
% v_1 = v_0(:, step_index);

x_1 = x';
y_1 = y';
v_1 = v_0';
excel_position = zeros(224 * 2, 101);
excel_v = zeros(224, 101);

for i = 1:224 * 2

    if mod(i, 2) ~= 0 %为奇数
        j = floor(i / 2) + 1; % (i/2 ) +1
        excel_position(i, :) = x_1(j, :);
    else %偶数
        k = i / 2;
        excel_position(i, :) = y_1(k, :);
    end

end

% 文件路径
filename = 'F:\EXCEL\result4.xlsx';

[~, ~, raw_data] = xlsread(filename, '位置');
start_row = 2;
start_col = 2;

[row, col] = size(excel_position);
raw_data(start_row:(start_row + row - 1), start_col:(start_col + col - 1)) =
num2cell(excel_position);

writecell(raw_data, filename, 'Sheet', '位置');

disp('矩阵已成功写入指定位置，并保留了文字信息');
writetable(excel_position, filename, 'Sheet', '位置');

disp('x, y 已写入 "位置" 表，速度 v 已写入 "速度" 表');
clc; clear; close all;

% 读取 Excel 文件
filename = 'F:\EXCEL\result6.xlsx'; % 替换为你的文件名
```

```
data = readtable(filename);

% 将表格数据转换为矩阵以便操作
data_matrix = table2array(data);
[r, c] = size(data);
% % 使用循环将左上角的值赋给右下角
for i = 178:c % % % %列的循环

    for j = 46:1:r

        if data_matrix(j, i) == 0 && data_matrix(j - 1, i - 2) ~= 0
            data_matrix(j, i) = data_matrix(j - 1, i - 2);
        end

    end

end

% for j = 18:33

%     for i = 158:c

%         if data_matrix(j, i) == 0
%             data_matrix(j, i) = data_matrix(j, 236 - i);
%         else continue;

%         end

%     end

% end

% 将矩阵转换回表格格式
updated_data = array2table(data_matrix, 'VariableNames',
data.Properties.VariableNames);

% 保存到 Excel 文件
% 确保写入时路径是正确的
writetable(updated_data, 'F:/EXCEL/result6.xlsx');
disp("数据已成功写入到 Excel 文件");
```

附录 5 第五问

```
clc,clear,close all;
t = 100;
dt = 0.1;
dx = 1.7;
n = round(16*0.55/dx);
flag =1;
for v = 1:0.0001:2
  [~, ~, ~, v0] = new_prepare(t, dx, dt, n,v);
 for j = 1:224
     for i = 1:1001
         if v0(i,j)>=2
          flag = 0;
         end
     end
 end
 if flag == 0
         disp(v);
         break;
end
end
function [r0, theta0, w0, v0] = new_prepare(t, dx, dt, n,v0)
    %%初始化参数
    L1 = 2.86; %龙头长度 单位：m
    L2 = 1.65; %龙身龙尾长度
    s = dx; %初始螺线宽距
    b = s / (2 * pi);
    delta_theta_1 = delta(L1, s, n);
    delta_theta_2 = delta(L2, s, n);
    t0 = 0:dt:t;
    t0 = t0';
    len_t = length(t0);
    v = zeros(len_t, 224); %定义线速度,一共 224 个把手
    theta = v;
    Omega = v; %定义角速度
    r = v; %定义极径
    v(:, 1) = v0;
    %%计算第一个点的角速度
    w1_funs = @(t, theta) v(1,1) ./ (b * sqrt(1 + (2 * n * pi
- theta) .^ 2));
    theta(:, 1) = 2 * n * pi - dis_euler(w1_funs, 1, t / dt, 0,
dt);
    r(:, 1) = b .* theta(:, 1);
```

```
        Omega(:, 1) = v(1,1) ./ sqrt(r(:, 1) .^ 2 + b ^ 2);
        %%计算第二个的角速度
        i = 2 * ones(1, 222);

        while (abs(theta(i(1), 1) - theta(1, 1)) < delta_theta_1)
            i (1) = i(1) + 1;

            if i(1) > len_t
                break;
            end

        end

        %刚性杆的约束条件
        calpha1 = @(t, theta_2) (L1 ^ 2 + r(t, 1) .^ 2 - (b * (2 *
n * pi - theta_2)) .^ 2) ./ (2 * L1 .* r(t, 1));
        salpha1 = @(t, theta_2)sqrt(1 - calpha1(t, theta_2) .^ 2);
        calpha2 = @(t, theta_2)((b * (2 * n * pi - theta_2)) .^ 2
+ L1 ^ 2 - r(t, 1) .^ 2) ./ (2 * b .* (2 * n * pi - theta_2) .*
L1);
        salpha2 = @(t, theta_2)sqrt(1 - calpha2(t, theta_2) .^ 2);
        % 沿杆的分速度相等
        funs = @(t, theta_2) Omega(t, 1) .* (calpha1(t, theta_2) +
theta(t, 1) .* salpha1(t, theta_2)) ./ (calpha2(t, theta_2) + (2 *
n * pi - theta_2) .* salpha2(t, theta_2));
        y0 = 0;
        h = 0.1;
        theta(i(1):end, 2) = 2 * n * pi - dis_euler(funs, i(1), t
/ dt, y0, dt);
        r(i(1):end, 2) = b .* theta(i(1):end, 2);

        for k = i(1):length(t0)
            Omega(k, 2) = funs(k, 2 * n * pi - theta(k, 2));
        end

        v(i(1):end, 2) = Omega(i(1):end, 2) .* sqrt(r(i(1):end,
2) .^ 2 + b ^ 2);
        %计算第二个点以后的角速度
        flag = 0;

        for j = 3:224
            i(j - 1) = i(j - 2);

            while (abs(theta(i(j - 1) + 1, j - 1) - theta(i(j - 2)
```

```
+ 1, j - 1)) < delta_theta_2)
                i (j - 1) = i(j - 1) + 1;

                if i(j - 1) > (t / dt)
                    flag = 1;
                    break;
                end

            end

            if flag == 1
                break;
            end

            calpha1 = @(t, theta_2) (L2 ^ 2 + r(t, j - 1) .^ 2 - (b
* (2 * n * pi - theta_2)) .^ 2) ./ (2 * L2 .* r(t, j - 1));
            salpha1 = @(t, theta_2)sqrt(1 - calpha1(t, theta_2) .^
2);
            calpha2 = @(t, theta_2)((b * (2 * n * pi - theta_2)) .^
2 + L2 ^ 2 - r(t, j - 1) .^ 2) ./ (2 * b .* (2 * n * pi - theta_2) .*
L2);
            salpha2 = @(t, theta_2)sqrt(1 - calpha2(t, theta_2) .^
2);
            funs = @(t, theta_2) Omega(t, j - 1) .* (calpha1(t,
theta_2) + theta(t, j - 1) .* salpha1(t, theta_2)) ./ (calpha2(t,
theta_2) + (2 * n * pi - theta_2) .* salpha2(t, theta_2));
            y0 = 0;
            h = 0.1;
            theta(i(j - 1):end, j) = 2 * n * pi - dis_euler(funs,
i(j - 1), t / dt, y0, dt);
            r(i(j - 1):end, j) = b .* theta(i(j - 1):end, j);

            for k = i(j - 1):length(t0)
                Omega(k, j) = funs(k, 2 * n * pi - theta(k, j));
            end

            v(i(j - 1):end, j) = Omega(i(j - 1):end, j) .*
sqrt(r(i(j - 1):end, j) .^ 2 + b ^ 2);
        end



        r0 = r;
        theta0 = theta;
        w0 = Omega;
```

```
        v0 = v;
    end
function y = dis_euler(f, a, b, y0,dt) % % % % % % %    欧拉法
    s = b - a + 1;
    Y = zeros(1, s + 1);
    Y(1) = y0;

    for k = 1:s
        y_pred = Y(k) + dt * f(a + k - 1, Y(k));

        % 修正步骤（使用预测值进行修正）
        Y(k + 1) = Y(k) + (dt / 2) * (f(a + k - 1, Y(k)) + f(a + k, y_pred));
        % Y(k + 1) = Y(k) + h * f(a + k - 1, Y(k));
    end

    y = Y'; %转置
end

function theta_result = delta(L, s, n)
    %用于求解两端点径向方向间初始夹角
    %输入:L 杆长
    %s 螺距宽
    %输出:两端点径向方向间初始夹角
    r = s * n;
    %       syms theta
    % 设置 fsolve 的选项
    options = optimoptions('fsolve', ...
        'TolX', 1e-12, ...                              % 变量变化精度
        'TolFun', 1e-12, ...
         'Display', 'final', ...       % 函数值变化精度
        'MaxIterations', 1000, ...                      % 最大迭代次数
        'MaxFunctionEvaluations', 5000, ...         % 最大函数评估次数
        'Algorithm', 'levenberg-marquardt');
    f = @(theta) cos(theta) - ((s * (2 * n * pi - theta) / (2 * pi)) ^ 2 + r ^ 2 - L ^ 2) / (2 * (s
* (2 * n * pi - theta) / (2 * pi)) * r);

    % 使用 vpasolve 求解
    theta_result = fsolve(f, 0.5);
end
```