基于等距螺线模型的“板凳龙”动态模拟与路径优化研究

摘 要

“板凳龙”，作为蕴含丰富文化底蕴的传统民俗活动，其表演艺术的优化成为研究焦点。本文旨在探讨如何通过创新策略，在确保舞龙队伍灵活盘绕与流畅过渡的基础上，实现盘龙所需空间的最小化与行进速度的最大化，进而提升表演效率与观赏性。

针对问题一，本文基于等距螺线理论，结合题目数据构建了其极坐标方程，并应用笛卡尔与极坐标转换法则，建立了**速度、时间与极角的精确关系**。通过Matlab**仿真模拟**，结合板长条件，精确计算出舞龙队每秒的位置与速度，并将关键数据整理入图表（见图表1-2至1-3），由于速度的矢量性，文章通过（图表1-4）使得**矢量性可视化**，同时将所有数据安全地归档至“result1.xlsx”文件中，以确保数据的专业性、完整性与可追溯性。

针对问题二，本文建立了**碰撞检测模型**，该模型能够在复杂的螺旋运动中准确捕捉碰撞时刻。模型目标是计算出板凳龙何时因碰撞停止，约束条件包括龙头前把手的中心与龙身距离。本文以板凳两把手中心连线作为线段，使用**线段相交**算法进行碰撞检测，使用**几何方法**计算龙头前把手与龙身其他部分的距离，当其小于两个板凳宽之和时，即为碰撞发生。通过Matlab**仿真模拟**，板凳龙将于**447s**时因发生碰撞而停止运动。最后将位置速度安全地归档至“result2.xlsx”文件中。

针对问题三，本文首先对复杂模型进行了合理的**简化**处理，通过构建**三角形结构**来准确提取关键点。基于问题一中推导的速度、时间与极角之间的关系，以及问题二中关于**碰撞发生机制**的条件，本文构建了角度随时间变化的**微分方程模型**。设定了统一的时间单位1作为分析基准，采用了**遍历与优化**策略。具体而言，在螺距P的0.40到0.50区间内，以0.005为步长进行了系统的遍历，利用Matlab的**优化迭代**功能，最终确定螺距最小值为**0.475**。

针对问题四，本文设计了**S形调头曲线优化**模型和**动态路径规划**模型。模型使用两段圆弧表示S形曲线，目标为缩短曲线长度，约束条件包括该S形曲线两次与掉头区域相切以及掉头区域边界的限制。本文使用**迭代搜索**和**仿真模拟**来进行对最短路径的规划，通过时间步长的累积求出结果。经计算，优化后的S形曲线满足所有约束条件，同时提升了掉头效率。此外在板凳龙盘入盘出过程中运用微积分相关知识，列出位置信息函数和速度函数，从而求解出调头前后龙头、龙身和龙尾各把手的位置坐标和速度信息，最后将位置速度安全地归档至“result4.xlsx”文件中。

针对问题五，基于问题四对于板凳龙的盘入盘出过程的**模拟**，通过**控制变量**法修改龙头的初始速度，然后利用问题四所使用的**仿真模拟**，对修改速度后的板凳龙进行仿真，利用微积分等相关**数学知识**求出各部分龙身的速度信息。此外利用**二分查找**法对龙头速度进行筛选，确保各部分龙身的最大速度不会超过2，经过若干次搜索，得出结果，当龙头速度为**1.25** 时，各部分龙身的速度恰好不超过2。

**关键词：微分方程；动态路径规划；仿真模拟；迭代搜索优化；二分查找**

# 

# 一、问题重述

“板凳龙”，这一承载着深厚文化传统的民俗表演，其核心在于如何在保持舞龙队伍灵活盘旋与顺畅流转的同时，寻求空间利用的最优化与行进速度的提升。某板凳龙由223节板凳组成，第1节为龙头，后面221节为龙身，最后1节为龙尾。龙头板长为341 cm，其余板凳均为220 cm，所有板凳的宽度均为30 cm。每节板凳上均有两个孔，孔的中心距离最近的板头27.5 cm。

**问题1：**计算舞龙队以1m∕s。在沿着螺距为55cm的等距螺线顺时针盘入的过程中，从初始时刻到300秒内，龙头、龙身和龙尾各把手的每秒位置和速度，并保存结果于特定文件中，同时在论文中给出特殊点的相关信息。

**问题2：**在问题一的基础上，要求确定舞龙队盘入的终止时刻，即因为板凳龙之间发生碰撞而无法继续盘入的时刻。并保存此时板凳龙各部分的位置和速度，并保存结果于特定文件中，同时在论文中给出特殊点的相关信息。

**问题3：**从盘入过程到盘出过程中，舞龙队将由顺时针盘入调头切换为逆时针盘出，要求找出最小的螺距，使得龙头前把手可以恰好沿着相应的螺线盘入到调头空间的边界，同时保证板凳之间不发生碰撞。

**问题4：**盘出螺线与盘入螺线关于螺线中心呈中心对称，舞龙队在问题3设定的调头空间内完成调头，调头路径是由两段圆弧相切连接而成的S形曲线，要求分析调头路径的圆弧是否可以调整以缩短调头曲线且保持各部分相切，同时计算从-100秒到100秒的过程中每秒舞龙队的位置和速度，在论文中给出特定时刻的相关数据。

**问题5：**基于问题四所设计的S形路径，尝试确定龙头的最大行进速度，确保舞龙队各把手的最大速度不超过2 m/s，并提供特定时刻的各把手位置和速度。

# 二、问题分析

## 2.1问题一的分析

问题一要求确定每秒整个舞龙队的位置和速度，文章从等距螺线入手，根据题目所给数据建立等距螺线极坐标方程，接着根据笛卡尔坐标系与极坐标系之间固有的转换法则，实现两者间的自由转换，最终建立速度时间以及极角之间的关系，通过Matlab仿真模拟以及速度时间以及极角的关系，确定每秒整个舞龙队的位置和速度。

## 2.2问题二的分析

问题二要求确定舞龙队沿螺线盘入的终止时刻，并确保板凳之间不发生碰撞。首先，需要定义碰撞，即某两板凳在舞龙过程中的直接接触。为此，通过建立板凳的物理模型，利用其尺寸确定碰撞条件并实时检测碰撞是否发生。当模型检测到发生碰撞时，记录该时刻为盘入的终止时刻，并利用数学方法计算各个时刻舞龙队的各部分的位置和速度。

## 2.3问题三的分析

问题三要求确定舞龙队在调头过程中所需的最小螺距，以确保龙头前把手能够沿着相应的螺线盘入到调头空间的边界。首先知道螺距越小，螺线的圈数越多，整体形状越紧凑，因此需要计算出最小的螺距，以便龙头前把手顺利到达边界。通过利用螺线的数学方程，计算龙头前把手在不同时间点的位置，并确定调头空间的边界位置，判断在不同螺距下龙头前把手到达边界的条件。最终记录下满足条件的最小螺距，以确保舞龙队在调头过程中能够顺利进入调头空间。

## 2.4问题四的分析

问题四要求舞龙队在调头过程中沿着设定的路径完成调头，并且对于掉头路径有详细要求。为解决该问题，首先分析调头路径的几何特性，求出圆弧的起始和结束位置，以及掉头路径与螺线的相切位置，以确保路径满足题目要求。此外问题还涉及路径优化，通过优化方法调整缩短调头曲线。利用几何和微积分方法，计算板凳龙各部分在盘入和盘出过程中的位置坐标及速度信息。

## 2.5问题五的分析

问题五要求在舞龙队沿问题四优化出的路径行进，要求各部分的最大速度不超过一定值，进而求出满足该条件的龙头的最大行进速度。基于问题四对盘入盘出过程的模拟，通过逐步调整龙头的速度变量，计算出各把手整个过程的最大速度，使其恰好符合速度限制条件。最终将计算得出的符合条件的龙头最大行进速度记录。

# 三、模型假设

1. 通过假设舞龙队沿螺线均匀运动，忽略摩擦、空气阻力等复杂因素，专注于研究舞龙队的整体运动特性。
2. 假设龙头速度恒定简化了速度对位置和时间关系的计算，使得模型更易于处理和预测。
3. 假设连接是刚性的，忽略连接点的微小形变，简化了模型的动力学分析。
4. 假设板凳上的孔不会对舞龙队的运动造成阻碍，且孔的存在仅用于连接而不会影响板凳的物理特性。
5. 简化碰撞条件为把手重叠，使得碰撞检测算法更容易实现，虽然这可能忽略了其他形式的碰撞。
6. 假设调头空间是固定的，确保模型的一致性和可重复性，便于分析和优化调头策略。
7. 假设舞龙队运动是连续的，忽略复杂的运动模式，使得运动轨迹的预测更加直接和简单。
8. 假设数学模型和数值方法能够达到所需精度，避免深入探讨数值误差，专注于模型的逻辑和结构。
9. 假设速度均匀分配简化了对舞龙队内部动力学的分析，使得整体运动的预测更加一致。

# 四、符号说明

|  |  |  |
| --- | --- | --- |
| **符号** | **说明** | **单位** |
|  | 原点到螺线上任意一点的径向距离 |  |
|  | 点与极轴之间的夹角 | 弧度 |
|  | 螺旋起始位置到中心距离 |  |
|  | 螺线的螺距特性 |  |
| P | 螺距 |  |
|  | 第个把手的横坐标 |  |
|  | 第个把手的纵坐标 |  |
|  | 第块板凳的长度 |  |
|  | 螺线放弧长 |  |
|  | 把手中心点i和把手中心点j的距离 |  |
|  | 把手中心点i的速度 |  |

# 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

首先，需要明确等距螺线的数学表达式。等距螺线（也称为阿基米德螺线）的极坐标方程可以表示为：

|  |  |
| --- | --- |
|  | 1-1 |

其中，r 代表从原点（即坐标系的中心）到螺线上任意一点的径向距离；θ 则表示该点与极轴（通常是x轴的正半轴）之间的夹角，以弧度为单位；而 a 和 b 是两个关键的常数项。a 在此方程中扮演着螺旋起始位置的角色，螺线从距离原点8.8米的某点开始绘制。b 代表了螺线的螺距特性，即每当角度 θ 增加一个单位（无论是弧度还是度，但在此上下文中应保持一致使用弧度），径向距离 r 将随之增加的量。这个增加的数值直接关联到螺线的紧密程度或展开速度，其中 （p为螺距）。

综上所述，等距螺线（阿基米德螺线）在极坐标系下的最终数学表达式为：

|  |  |
| --- | --- |
|  | 1-2 |

根据笛卡尔坐标系与极坐标系之间固有的转换法则，可以将极坐标的表达式精妙地转化为笛卡尔坐标系中的对应形式（其中n代表第n个位置）：

|  |  |
| --- | --- |
|  | 1-3 |

为了求阿基米德螺线在弧度制下的弧线微分长，本文需要使用弧长公式，该公式在极坐标下可以表示为：

|  |  |
| --- | --- |
| d | 1-4 |

首先，对阿基米德螺线的极坐标方程 求关于 θ 的导数，得到：

，然后，将 r 和 代入弧长公式中，得到： 。

化简得：

|  |  |
| --- | --- |
|  | 1-5 |

本表达式给出了阿基米德螺线上任意一小段弧长的微分形式，在此需要对 ds 进行积分，即：

|  |  |
| --- | --- |
|  | 1-6 |

舞龙队沿着具有55厘米螺距的等距螺线顺时针方向进行盘旋运动。根据题目设定，每一节板凳（包括龙头、龙身及龙尾）的把手中心均严格位于螺线的几何路径上。当相邻两节板凳的把手均成功嵌入螺线并稳定排列时，可以明确计算出每节板凳的板长，即两个相邻把手中心之间的直线距离,板长为：

|  |  |
| --- | --- |
|  | 1-7 |

### 5.1.2模型的求解

通过速度与时间之间的基本关系式进行推导，可以轻易地得出以下结论：

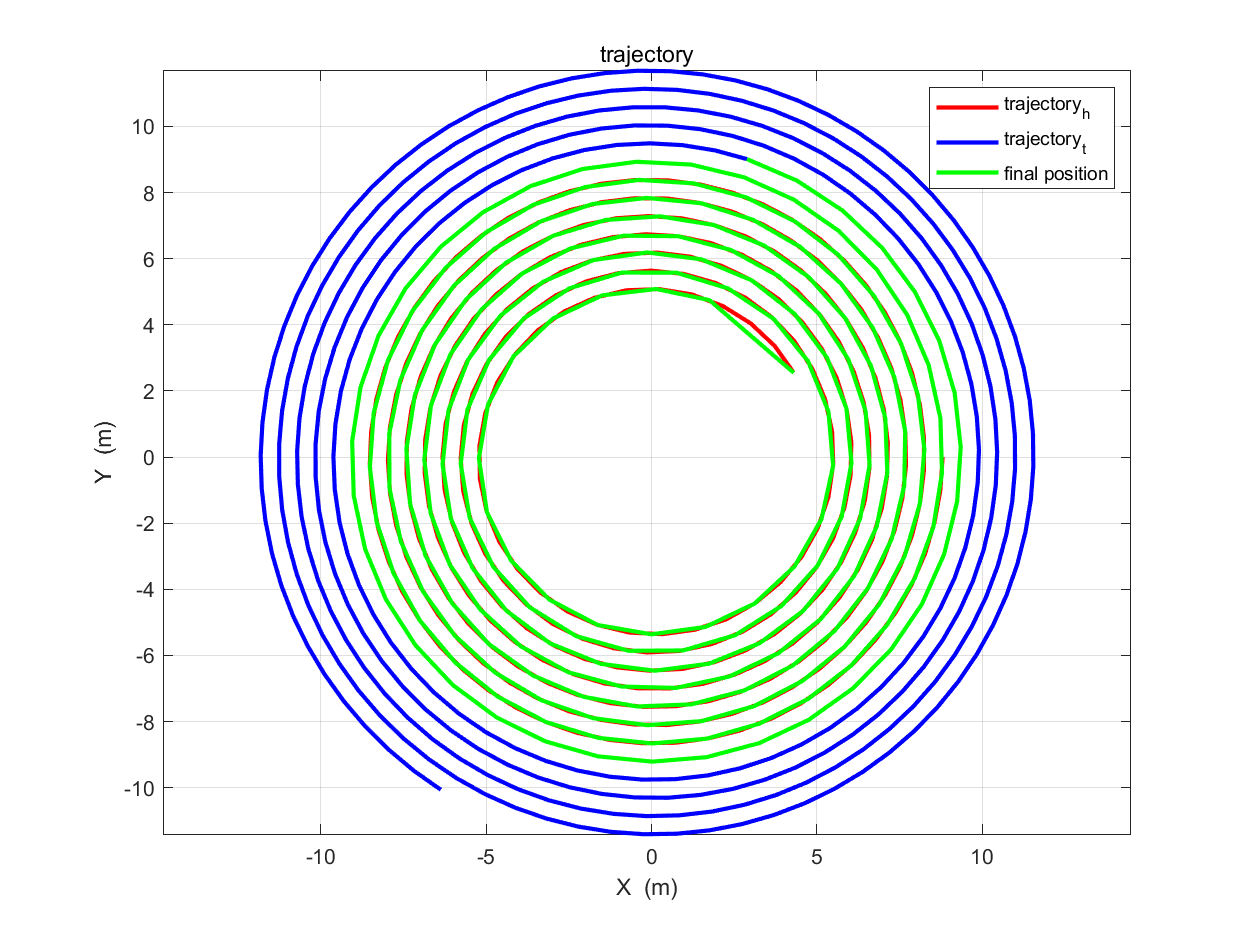
给定速度v与时间t的关系，可以直接推导出。进一步地，将这一结论与先前给出的公式1-2等距螺线极坐标方程、公式1-5弧长微分相结合，经过简单的代数运算，能够轻松地获得：

|  |  |
| --- | --- |
|  | 1-8 |

对于龙头前把手的位置，依据题目设定的条件，其速度需恒定维持在1。在此前提下，前把手的运动轨迹仅受时间及极角变化的驱动。利用Matlab这一强大的仿真工具，能够精确地模拟出龙头前把手在不同时间点的位置及其恒定速度。

当考虑龙头后把手及其他位置时，由于速度参数可能发生变化，本文采取的策略是增加板长尺寸限制，并据此重新对前把手的运动进行模拟仿真。此过程中，本文继续运用了公式1-8作为核心计算依据，公式1-7板长公式作为辅助工具，并在后续点的坐标计算中引入了额外的约束条件，以确保计算的准确性和合理性。通过这些步骤，能够进一步求解出龙头其他位置时把手的具体位置以及相应的速度变化。关于Matlab的详细实现代码，请参阅附录2。

通过Matlab仿真模拟，对公式1-8进行了深入细致的求解过程，成功获得了龙头轨迹、龙尾轨迹以及它们的最终位置。这些结果已经在以下图（图表 1-1）中直观地展示出来，便于进一步分析和理解。



图表 1-1

图表1-2展示了在0 s、60 s、120 s、180 s、240 s和300 s时，龙头前把手（即龙头）、龙头后面第1、51、101、151、201节龙身前把手以及龙尾后把手的X和Y位置坐标。这些数据反映了板凳龙随时间变化的运动轨迹。具体数据如下：



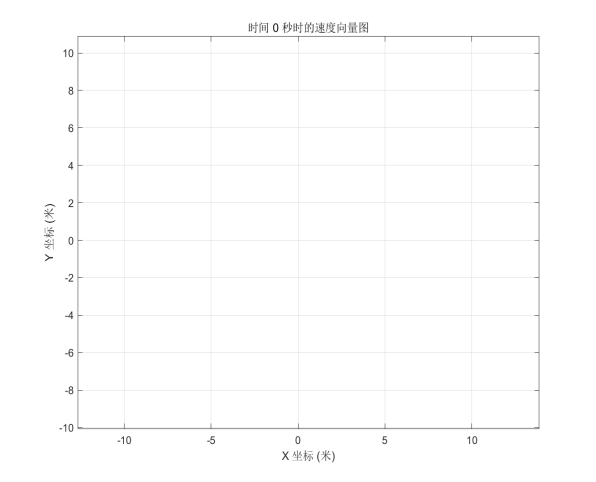
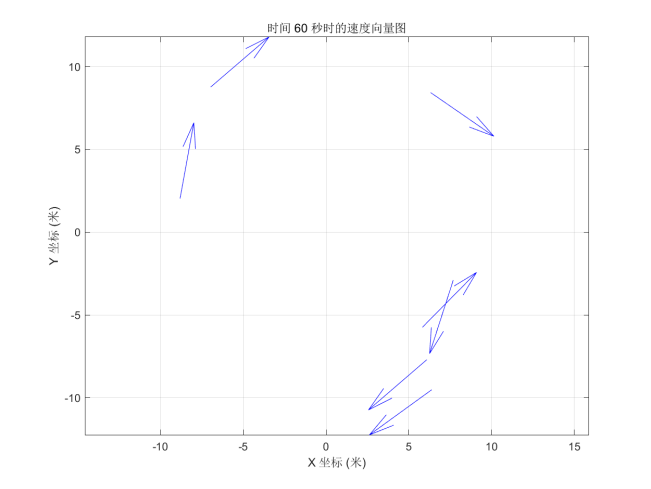
图表 1-2

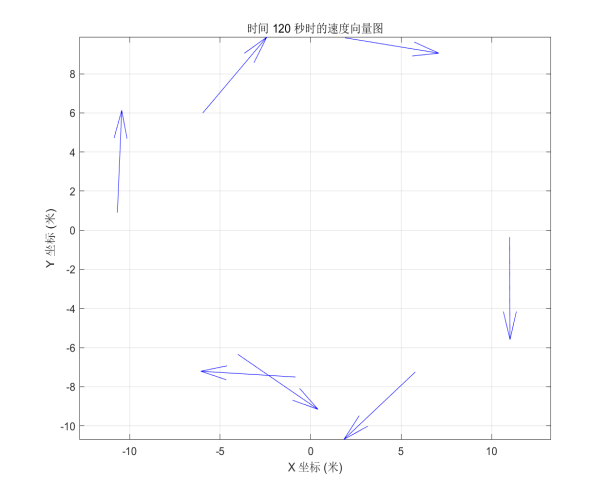
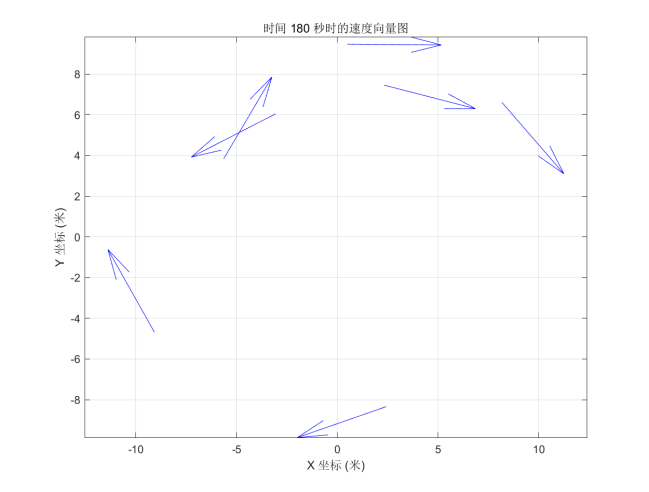
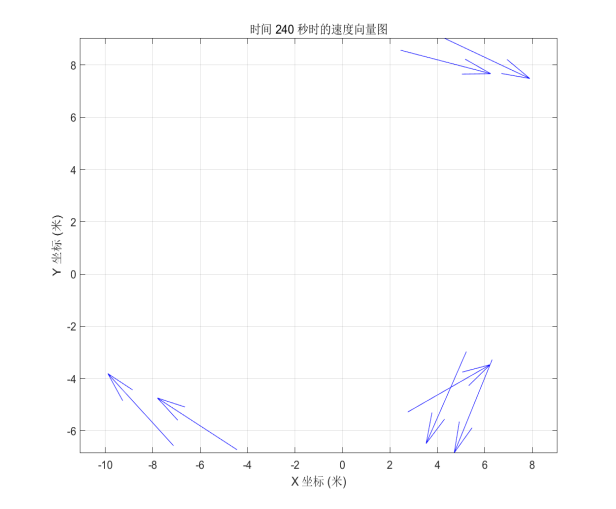
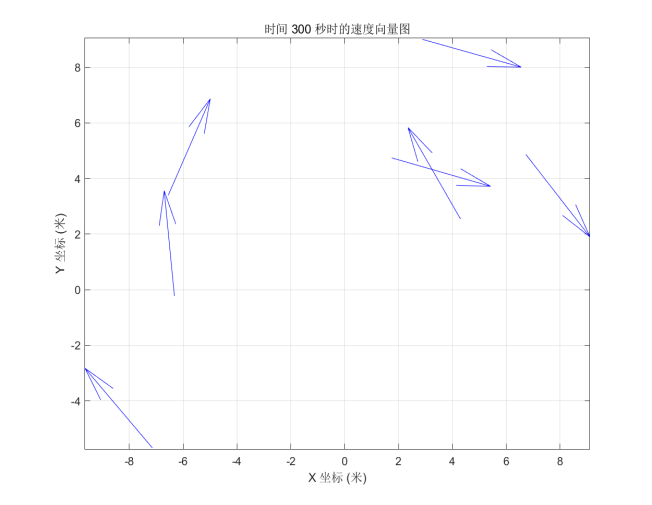
图表1-3展示了在0 s、60 s、120 s、180 s、240 s和300 s时，龙头前把手（即龙头）、龙头后面第1、51、101、151、201节龙身前把手以及龙尾后把手速度。具体数据如下：



图表 1-3

由于速度的矢量性，在接下来的图表4中展示0 s、60 s、120 s、180 s、240 s和300 s时龙头前把手、龙头后面第1、51、101、151、201节龙身前把手以及龙尾后把手速度矢量图：

图表 1-4

在数据采集过程中，确保所有测量均遵循既定的科学标准与操作规程，以减少误差并提高数据的可靠性。随后，将收集到的所有数据以结构化格式整理，并安全地保存至名为“result1.xlsx”的文本文件中。

## 5.2问题二模型的建立与求解

### 5.2.1问题二模型的建立

针对舞龙队沿螺线轨迹盘入过程的深入探索，解除原先设定的前300秒模拟时间限制，以构建一个更为全面且动态的分析框架。在这一框架下，文章首先为龙头及每一节龙身设定碰撞检测箱体积，这些反映各部分的实际空间占用，确保碰撞检测的精确。

为此本文设计并实现一个碰撞检测模型算法，该算法用逐秒时间步进策略来模拟舞龙队的动态运动过程。这一算法的核心在于实时计算与判断舞龙队中龙身之间的相对位置关系。当算法检测到任意两节板凳之间的实际距离缩减至小于两板凳宽度之和时，即判定为发生了碰撞事件，并依据此进行后续的模拟响应或调整。

在舞龙队表演中，为了检测任意两个板凳是否发生碰撞，首先需要定义一个距离公式来计算这两个中心点之间的距离。

假设把手中心点 *i* 的坐标为，把手中心点 *j* 的坐标为 ，则它们之间的欧几里得距离 可以通过以下公式计算：

|  |  |
| --- | --- |
|  | 2-1 |

在舞龙队盘入过程中，如果任意两个相邻把手（代表龙身的不同部分，通常连接在板凳上）中心点的距离小于两板凳宽度之和的实际长度（这里假设为2d，其中d是单个板凳的宽度），可以认为舞龙队伍在该位置发生了碰撞，并且盘入过程应该在该时刻结束：

|  |  |
| --- | --- |
|  | 2-2 |

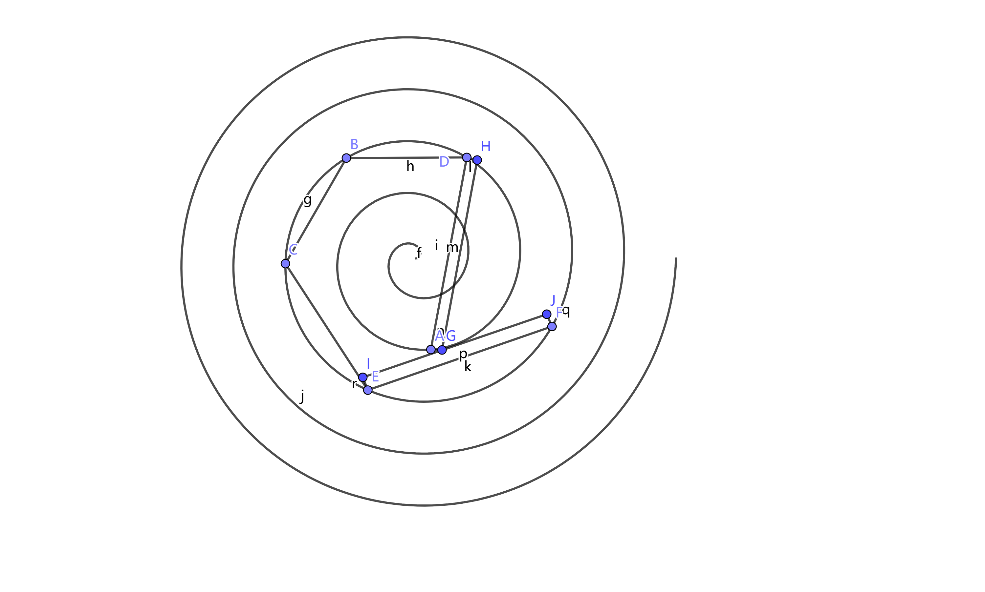
利用极坐标系下等距螺线的数学模型，可以计算出舞龙队在任意时间的位置坐标：

|  |  |
| --- | --- |
|  | 2-3 |

同时由公式（7）的一般形式，结合中分差分法进行数值微分：

|  |  |
| --- | --- |
|  | 2-4 |

当两把手之间的距离小于板凳的宽度时，龙头必定与龙身某一处相撞，导致盘入过程结束。记录此时刻的时间，以此时刻作为盘入终止时刻，同时记录此时刻龙头、所需龙身和龙尾的位置和速度，进行导出并记录。然后继续使用问题一所使用的仿真模拟算法，收集记录问题所需数据。



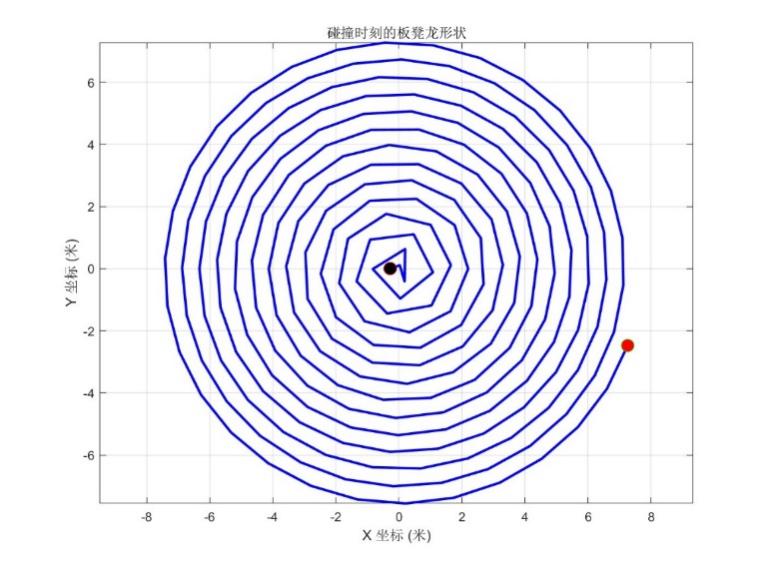
图表 2-1

该模型考虑的主要因素有：首先是龙头及龙身的宽度，其宽度将会导致舞龙盘入较为小的部分时会发生碰撞情况，导致队伍停止前进。然后是板凳龙的运动特性，其速度及把手所在位置前进方式，这将决定最后舞龙结束的位置和特点。最后是螺线的数学性质，在舞龙的过程中，其龙头所能运动的范围将会因为螺线的特殊性质而收到限制，最终导致舞龙的结束。

### 5.2.2问题二模型的求解过程

设定初始条件，包括初始角度、初始速度  和时间步长。从初始时刻开始，逐步增加时间 ，使用上述公式计算每个时间点的舞龙队位置和速度，并在每个时间步长内，使用碰撞检测条件检查是否发生碰撞。

一旦检测到碰撞，记录当前时刻作为盘入的终止时刻，并在终止时刻，记录舞龙队各部分的位置和速度，同时生成该时刻的板凳龙的形状，如图表2-2：



图表 2-2

将相关数据输入程序并做好数据保存，使用Matlab进行数据可视化以及分析（关于Matlab的详细实现代码，请参阅附录3），可得盘入过程结束是该板凳龙的位置情况及所需数据如下（图表2-3）：

此外，由仿真模拟及可视化的结果得到，龙头将于447s时与龙身相撞，从而盘入过程结束，下表为出在结束时刻时龙头前把手及龙头后面第1、51、101、151、201条龙身前把手和龙尾后把手的位置和速度：

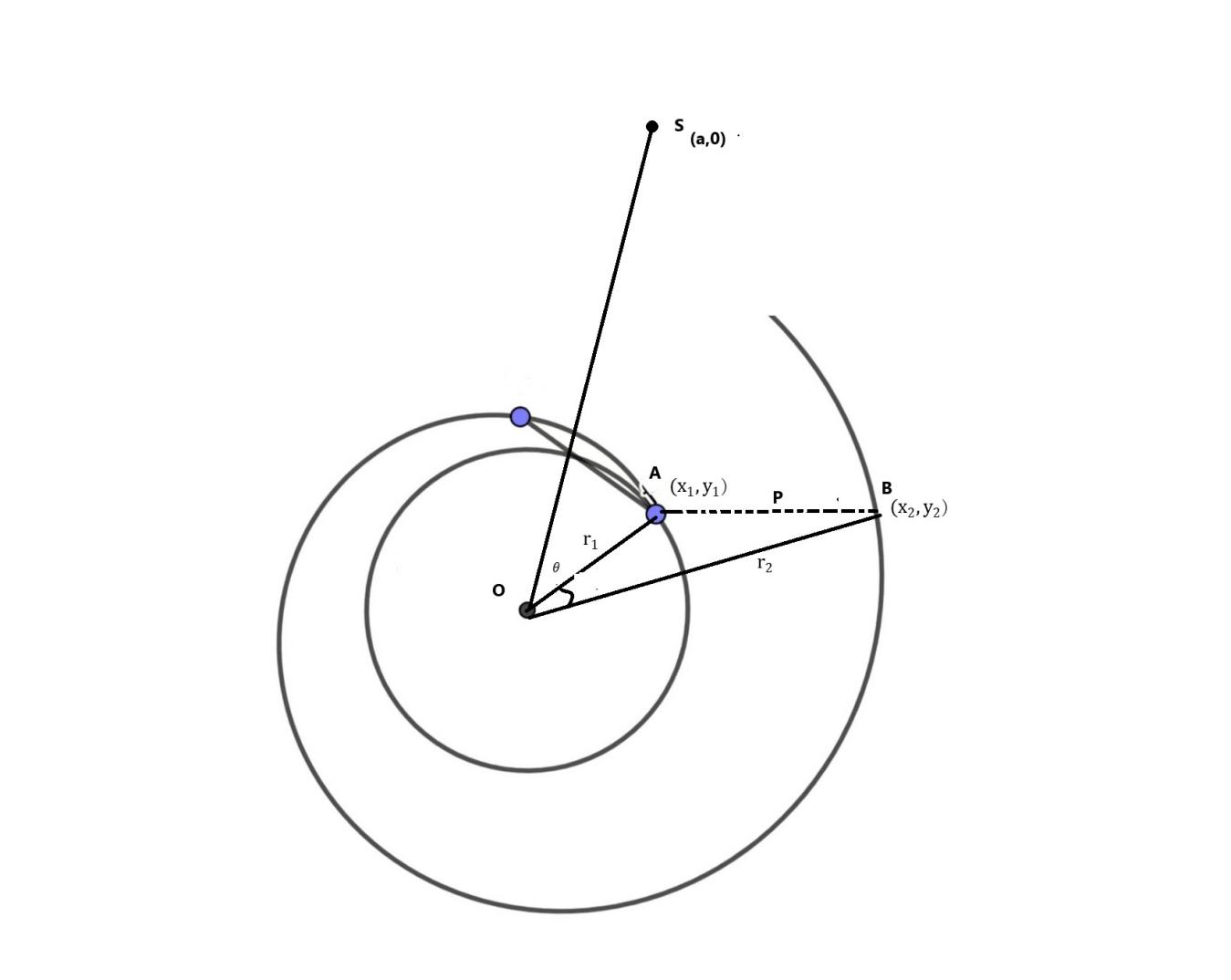
图表 2-3

|  |  |  |  |
| --- | --- | --- | --- |
| 龙头龙身龙尾\数据种类 |  |  |  |
| 龙头前把手 | -0.2738 | 0.0037 | 0.4088 |
| 1 | 0.0306 | 0.1095 | 0.4665 |
| 51 | -3.3245 | -1.4071 | 0.9938 |
| 101 | -2.5464 | 4.4568 | 0.9969 |
| 151 | -6.1185 | 1.517 | 0.9980 |
| 201 | -0.4413 | 7.2794 | 0.9985 |
| 龙尾后把手 | 7.2609 | -2.4752 | 0.9986 |

## 5.3问题三模型的建立与求解

### 5.3.1模型的建立

在问题三模型建立时，可以聚焦于等距螺线的最后两个周期，以便更清晰地分析。设想这样的场景：S点标记了当前螺旋轨迹的起始不确定位置，而A点则是舞龙队伍开始盘绕进入掉头空间的具体位置。B点则是通过从原点O出发，沿OA方向延伸并与螺旋线相交而确定的点。



为了简化理解和计算，本文采用一种示意性的调整：尽管在实际情况中B点可能位于A点的上方或下方极近的位置，但在此为了直观展示，将B点暂时置于A点的水平位置上。这样的处理虽为示意，却能有效说明问题的核心——即AB之间的直线距离P，实际上代表了等距螺线的一个关键参数：螺距。

经过这样简化之后，便很容易得到螺距的表达式：

|  |  |
| --- | --- |
|  | 3-1 |

对于A点B点，结合公式1-1等距螺线的极坐标方程，可以得到：

|  |  |
| --- | --- |
|  | 3-2 |

因为与之间相差一个周期，故而 。

先以切入半径（即4.5米）作为基础，这是螺旋线开始时的关键参考尺寸。接着，还需要考虑到从螺旋线的起始点S（虽然具体位置不确定，但可以假设为某一虚拟的起点）到A点（舞龙队盘入的具体位置）之间的相对位置关系。这一相对位置关系主要通过角度来确定：即起始位置到A点连线与螺旋线起始切线（或某一参考方向，如水平方向）之间的夹角。基于这个角度，可以计算出在该角度下，从切入半径边缘（即螺旋线开始处）到A点所沿螺旋线方向延伸的直线距离：

|  |  |
| --- | --- |
|  | 3-3 |

5.3.2 模型的求解

通过模型建立对问题的简化之后，对于题目中多余的参数，尚且可以联立不确定的起点S 的相关内容，对于如上模型建立S点是不确定的，但能够确定S点的具体参数，如，， 。

此时，再结合问题一盘入规则，以及问题二盘入限制，使得螺距从问题一的0.55m开始递减，可以尝试缩小螺距范围，例如将0.50的螺距带入前两问的代码中，通过对第二问的盘入限制进行分析，其并不会影响最终的结果，故而螺距从0.5开始递减。其次确定下限，木板的宽度为0.3米，并且考虑到木板斜放问题，可以将下限提升至0.40，故而目前仅需对螺距从0.40遍历到0.50。

可以根据本模型建立过程以及问题一模型的建立得到如下最终模型（此时龙头的速度仍为1）：

|  |  |
| --- | --- |
|  | 3-4 |

时间从1开始，每次时间增加1，求解微分方程，即可得龙头在不同时间点的角度和位置，结合问题二的碰撞原理，当其发生碰撞时可退出本次模拟，在不发生碰撞前提下，达到了边界值，即：

|  |  |
| --- | --- |
|  | 3-5 |

对所有可能出现的结果遍历，取螺距变化量为0.005，这个长度相对于木板长度等特别小，可以忽略不计，经Matlab仿真模拟（Matlab代码详见附录4）之后，得到最小螺距应为0.475，由于结果运行数量庞大，论文中不予展示，可运行代码查看。

## 5.4问题四模型的建立与求解

### 5.4.1问题四模型的建立

在问题四中，需要面对的是修改螺线螺距大小后，板凳龙如何在给定的直径为9米的圆形掉头区域内完成盘出过程的挑战。需要验证新的螺线路径是否能够有效引导板凳龙进入并完全位于掉头区域内，进而完成盘出动作。这要求建立一个动态路径规划模型，以确保板凳龙能够顺利沿着新的螺线路径移动，并在适当的时候进入掉头区域。此外其修改后的螺距至少要满足：

|  |  |
| --- | --- |
|  | 4-1 |

对于盘入与盘出螺线，由之前的问题可以得到其参数方程为：

|  |  |
| --- | --- |
|  | 4-2 |

另外对问题进行分析，可得到盘入螺线与盘出螺线为中心对称的关系，其参数方程为：

|  |  |
| --- | --- |
|  | 4-3 |

此后，本文依旧使用仿真模拟对板凳龙的盘入过程进行模拟，使得龙头前把手恰好沿着相应的螺线盘入到调头空间的边界，已知调头路径是一条与盘入、盘出螺线均相切的由两段圆弧组成的S形曲线，故假设两端圆弧 的圆心分别为和，半径分别为。由此可以得到两端圆弧的方程为：

|  |  |
| --- | --- |
|  | 4-4 |

对两端圆弧方程分别求导，即可得到圆弧上各个点速度的函数表达式。结束盘入过程进入掉头区域后，各点要一直位于掉头区域的内部，且因为在舞龙过程中，龙头的线速度恒为 ，故得到以下限定条件：

|  |  |
| --- | --- |
|  | 4-5 |

在盘入阶段，如果将掉头开始的时刻设为0时刻，即在的过程中，其位置函数可得：

|  |  |
| --- | --- |
|  | 4-6 |

同样，对位置函数求导可以得到各个点速度的函数表达式，如下：

|  |  |
| --- | --- |
|  | 4-7 |

由此同理可以得到，在盘出阶段时，假设掉头阶段时长为T，其位置函数和速度函数与上式类似。在限制区域内掉头的过程中，结合圆弧方程即可得出：

当龙头位于第一段圆弧上，即时，有以下公式：

|  |  |
| --- | --- |
|  | 4-8 |

当龙头位于第二段圆弧上，即时，有以下公式：

|  |  |
| --- | --- |
|  | 4-9 |

在板凳龙结束掉头阶段并进入盘出阶段时，其运动轨迹相对于盘入阶段呈现出中心对称的特性。将掉头区域的中心点视为对称中心，那么盘出阶段的每一刻位置都可以通过在盘入阶段对应时刻的位置上进行中心对称变换来得到。由于位置函数和速度函数在盘入阶段已经通过仿真模拟和其他方法求解得到，因此可以利用这些结果来推导盘出阶段的位置和速度。

### 5.4.2模型求解

将数据及公式等经过仿真模拟及最短路径动态规划及优化算法（Matlab代码详见附录 5）后，可以分别求出在−100 s、−50 s、0 s、50 s、100 s时，龙头前把手、龙头后面第1、 51、101、151、201节龙身前把手和龙尾后把手的位置（下表为简略数据）：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **位置\时刻（x，y）** | **-100** | **-50** | **0** | **50** | **100** |
| **龙头前把手** | (7.78, 3.72) | (6.61, 1.90) | (-2.71, -3.59) | (1.70, 6.07) | (-2.80, 7.67) |
| **1节龙身** | (6.21, 6.11) | (5.37, 4.48) | (-0.06, -4.67) | (4.14, 4.58) | (0.03, 8.07) |
| **51节龙身** | (-10.61, 2.83) | (-3.63, -8.96) | (2.46, -7.78) | (-1.34, -6.17) | (2.41, 3.83) |
| **101节龙身** | (-11.92, -4.80) | (10.13, -5.97) | (3.01, 10.11) | (-7.79, 4.89) | (-7.40, 1.69) |
| **151节龙身** | (-14.35, -1.98) | (12.97, -3.81) | (-7.00, 10.34) | (-4.29, -10.53) | (9.60, -3.18) |
| **201节龙身** | (-11.95,10.6) | (10.52,-10.8) | (-6.87, 12.39) | (0.69, -13.17) | (8.25, 8.88) |
| **龙尾后把手** | (-1.01, -16.53) | (0.19, 15.72) | (-1.93, -14.71) | (5.54, 12.76) | (-10.78, -7.10) |

此外，在−100 s、−50 s、0 s、50 s、100 s时，龙头前把手、龙头后面第1、 51、101、151、201节龙身前把手和龙尾后把手速度信息如下表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **位置\时刻** | **-100** | **-50** | **0** | **50** | **100** |
| **龙头前把手** | 0.9994 | 0.9991 | 0.9954 | 0.9990 | 0.9994 |
| **1节龙身** | 0.9994 | 0.9989 | 0.9811 | 0.9993 | 0.9995 |
| **51节龙身** | 0.9990 | 0.9982 | 0.9788 | 0.9998 | 1.0103 |
| **101节龙身** | 0.9988 | 0.9979 | 0.9783 | 0.9989 | 0.9646 |
| **151节龙身** | 0.9987 | 0.9978 | 0.9781 | 0.9986 | 0.9641 |
| **201节龙身** | 0.9987 | 0.9977 | 0.9780 | 0.9984 | 0.9639 |
| **龙尾后把手** | 0.9987 | 0.9977 | 0.9780 | 0.9984 | 0.9638 |

该板凳龙在盘入，掉头及盘出过程中的其他数据已保存于文件“result4.xlsx”中。

## 5.5问题五模型的建立与求解

### 5.5.1问题五模型的建立

对于问题五，提出一种自适应二分速度搜索算法，用来求解该动态速度约束优化模型，这个算法的核心思想是使用二分搜索方法来高效地找到满足所有约束条件的最大龙头速度。

即通过设定速度的范围和收敛阈值，将目标速度设置为范围的中间值，检测其他所有的把手的最大速度，若任何把手的速度超过目标速度，则模拟结束并返回失败；

若所有把手的速度都不超过目标速度，则模拟成功并返回速度值。

由公式4-4及公式1-6结合，以及圆弧本身的性质，和其中心对称的条件，可以得到以下公式：

|  |  |
| --- | --- |
|  | 5-1 |

第一段圆弧和第二段圆弧公式因为其中心对称的条件，可进行简化，如下：

|  |  |
| --- | --- |
|  | 5-2 |

同理，对位置函数进行求导可以得到该点的速度函数，即：

|  |  |
| --- | --- |
|  | 5-3 |

题目要求了所有把手的最大速度均不超过2米每秒，则可以得到限制条件：

|  |  |
| --- | --- |
|  | 5-4 |

### 5.5.2模型的求解

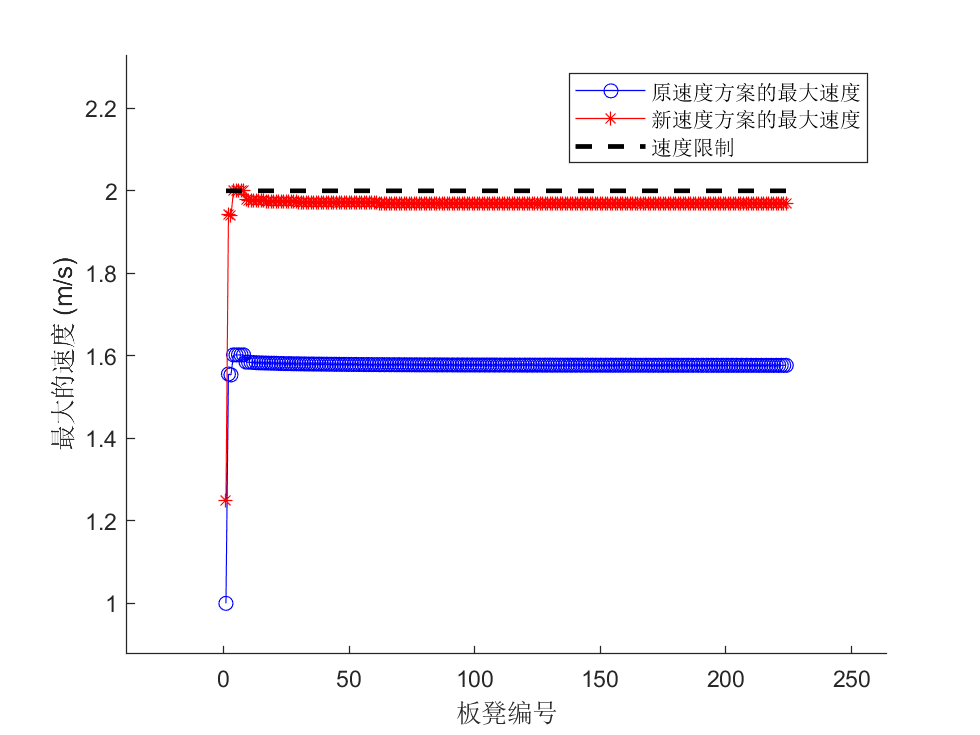
设定速度搜索范围，其中可以设为一个较大的值，例如5米每秒，而可以设置为0米每秒，然后设置收敛阈值=0.01。而中间速度：

|  |  |
| --- | --- |
|  | 5-5 |

若单次循环结束后，若三个速度之间的关系满足：

|  |  |
| --- | --- |
|  | 5-5 |

则该搜索过程结束，返回记录并输出得到的结果，可以得到允许的最大龙头速度:为1.25 m/s。（Matlab代码详见附录 6）其各龙身的最大速度如下图所示：



图表 5-1

# 六、模型的评价、改进与推广

## 6.1模型的优点

动态模拟模型为舞龙队提供了一个全面分析的视角，通过模拟不同时间点的状态，它不仅揭示了整个队伍的运动趋势，还能够帮助我们发现并规避潜在的问题和风险，从而在实际表演前进行必要的调整。此外，与传统的实际系统测试相比，动态模拟在成本和时间上都更为经济高效。

碰撞模型通过实时检测和预测潜在碰撞，为避免这些碰撞提供了重要的决策支持。引入碰撞箱体积和安全距离阈值的概念，进一步增强了模型的实用性和安全性，确保了舞龙队在表演过程中的流畅性和安全性。

路径优化模型则专注于提高舞龙队表演的效率和观赏性。通过精确模拟和优化，模型减少了预测误差，同时考虑了舞龙队在调头过程中的动态路径规划，确保了路径的连续性和光滑性。此外，模型还探索了如何通过调整来缩短调头曲线，这不仅提高了表演的紧凑性，也增加了其艺术性和观众的观赏体验。

## 6.2模型的缺点

碰撞模型与动态模拟的有效性在很大程度上依赖于输入数据的准确性和完整性。由于模拟过程通常需要大量的计算资源和时间，因此在资源有限的情况下可能面临挑战。

优化模型带来资源消耗大、过度拟合、增加模型复杂性、稳定性问题、忽视实际应用限制、依赖数据质量等缺点

## 6.3模型的改进

首先，增强数据收集和处理能力，确保输入数据的准确性和完整性，以提高模型的预测性和可靠性；其次，采用先进的算法和技术，如机器学习和人工智能，以提升模型的计算效率和适应性；最后，考虑实际应用中的限制和用户需求，使模型更加实用和用户友好。通过这些改进措施，可以确保模型更加精确、高效和可靠性。

# 七、参考文献

[1]阿基米德螺线[J].新世纪智能,2024,(45):49.

[2]满万鑫, 李新洪, 周思引, 安继萍, 张治彬, 胡港旋, 张国辉. 复杂约束多体系统显式动力学表示及其应用[J]. 应用力学学报, 1-8.

[3]冯俊杰. 多机器人系统避障与最优协调[D]. 广西大学, 2014.

[4]赵慧南.复杂环境下移动机器人的动态目标检测与跟踪控制策略分析[J].时代教育,2018,(13):128+4-5.

[5] 刘佳旭,陈嵩,蔡声泽,许超,褚健.基于鲁棒控制的自适应分数阶梯度优化算法设计（英文）[J].控制理论与应用,2024,41(07):1187-1196.

1. 张叶茂.基于实时交通信息的动态路径规划算法研究[J].西部交通科技,2017,(02):80-83+112.

[7]P. van den Driessche,James Watmough. Reproduction numbers and sub-threshold endemic equilibria for compartmental models of disease transmission[J]. Mathematical Biosciences,2002,180(1).[1]P. van den Driessche,James Watmough. Reproduction numbers and sub-threshold endemic equilibria for compartmental models of disease transmission[J]. Mathematical Biosciences,2002,180(1).

附录

|  |
| --- |
| 附录1 |
|  |
| 1. result1.xlsx 文件在支撑材料中，用于保存问题一结果 2. result2.xlsx 文件在支撑材料中，用于保存问题二结果 3. result4.xlsx 文件在支撑材料中，用于保存问题四结果 |

|  |
| --- |
| 附录2 |
| Matlab 解决问题一 |
| 1. clear 2. % 定义常量 3. Lh = 3.41; % 龙头长度（米） 4. Lb = 2.20; % 龙身长度（米） 5. Le = Lb - 2 \* 0.275; % 有效长度（米） 6. p = 0.55; % 螺距（米） 7. a = p / (2 \* pi); % 螺线参数 8. vh = 1; % 龙头速度（米/秒） 9. T = 300; % 总时间（秒） 10. dt = 1; % 时间步长（秒） 11. n = 223; % 板凳总数 12. % 初始化数组 13. t = 0:dt:T; 14. th = zeros(n, length(t)); 15. x = zeros(n, length(t)); 16. y = zeros(n, length(t)); 17. vx = zeros(n, length(t)); 18. vy = zeros(n, length(t)); 19. % 初始化位置 20. th(1, 1) = 16 \* 2 \* pi; % 龙头初始位置在第16圈 21. for i = 2:n 22. if i == 2 23. ds = Lh; 24. else 25. ds = Le; 26. end 27. dth = ds / (a \* th(i-1, 1)); 28. th(i, 1) = th(i-1, 1) + dth; 29. end 30. % 计算初始位置的x和y坐标 31. for i = 1:n 32. x(i, 1) = a \* th(i, 1) \* cos(th(i, 1)); 33. y(i, 1) = a \* th(i, 1) \* sin(th(i, 1)); 34. end 35. % 主循环 36. for j = 2:length(t) 37. % 计算龙头新的theta（顺时针盘入，theta减小） 38. ds = vh \* dt; 39. dth = ds / (a \* th(1, j-1)); 40. th(1, j) = th(1, j-1) - dth; 41. % 计算龙头位置 42. x(1, j) = a \* th(1, j) \* cos(th(1, j)); 43. y(1, j) = a \* th(1, j) \* sin(th(1, j)); 44. % 计算龙头速度 45. vx(1, j) = -vh \* sin(th(1, j)); 46. vy(1, j) = vh \* cos(th(1, j)); 47. % 计算龙身和龙尾位置和速度 48. for i = 2:n 49. if i == 2 50. ds = Lh; 51. else 52. ds = Le; 53. end 54. dth = ds / (a \* th(i-1, j)); 55. th(i, j) = th(i-1, j) + dth; 57. x(i, j) = a \* th(i, j) \* cos(th(i, j)); 58. y(i, j) = a \* th(i, j) \* sin(th(i, j)); 60. % 计算速度 61. vx(i, j) = (x(i, j) - x(i, j-1)) / dt; 62. vy(i, j) = (y(i, j) - y(i, j-1)) / dt; 63. end 64. end 65. % 保存到result1.1.xlsx 66. result1 = zeros(2\*n, length(t)); % 修改为2\*n行，存储位置和速度 67. for i = 1:n 68. result1(2\*i-1, :) = x(i, :); % 存储x坐标 69. result1(2\*i, :) = y(i, :);   % 存储y坐标 70. end 71. % 添加龙身速度到result1 72. velocity = zeros(n, length(t)); % 初始化速度矩阵 73. for j = 1:length(t) 74. for i = 2:n 75. velocity(i, j) = sqrt(vx(i, j)^2 + vy(i, j)^2); 76. end 77. end 78. % 将速度添加到result1矩阵的后面 79. result1 = [result1; velocity]; 80. writematrix(round(result1', 6), 'result1.1.xlsx'); 81. % 保存论文中需要的表格数据 82. selT = [0, 60, 120, 180, 240, 300]; 83. selS = [1,  52, 102, 152, 202, 223]; 84. posTable = zeros(14, 6); 85. velTable = zeros(7, 6); 86. for i = 1:length(selT) 87. tIdx = selT(i) + 1; 88. for j = 1:length(selS) 89. sIdx = selS(j); 90. posTable(2\*j-1, i) = x(sIdx, tIdx); 91. posTable(2\*j, i) = y(sIdx, tIdx); 92. velTable(j, i) = sqrt(vx(sIdx, tIdx)^2 + vy(sIdx, tIdx)^2); 93. end 94. end 95. % 保存位置表格 96. posTableRounded = round(posTable, 6); 97. writematrix(posTableRounded, 'Position.xlsx'); 98. % 保存速度表格 99. velTableRounded = round(velTable, 6); 100. writematrix(velTableRounded, 'Speed.xlsx'); 101. % 论文展现的位置以及速度 102. disp('Position:'); 103. disp(posTableRounded); 104. disp('Speed:'); 105. disp(velTableRounded); 106. % 绘制运动轨迹图 107. figure('Position', [100, 100, 800, 600]); 108. plot(x(1,:), y(1,:), 'r-', 'LineWidth', 2); 109. hold on; 110. plot(x(end,:), y(end,:), 'b-', 'LineWidth', 2); 111. plot(x(:,end), y(:,end), 'g-', 'LineWidth', 2); 112. title('Trajectory'); 113. xlabel('X (m)'); 114. ylabel('Y (m)'); 115. legend('Head Trajectory', 'Tail Trajectory', 'Final Position'); 116. grid on; 117. axis equal; 118. saveas(gcf, 'trajectory.png'); 119. saveas(gcf, 'trajectory.fig'); 120. % 绘制特定时间点的速度向量图 121. selT = [0, 60, 120, 180, 240, 300]; 122. selS = [1, 2, 52, 102, 152, 202, 223]; % 龙头、龙头后第1、51、101、151、201节和龙尾 123. for k = 1:length(selT) 124. timeIdx = selT(k) + 1; % MATLAB索引从1开始 125. % 提取当前时间点的位置和速度 126. xCur = x(:, timeIdx); 127. yCur = y(:, timeIdx); 128. vxCur = vx(:, timeIdx); 129. vyCur = vy(:, timeIdx); 131. % 筛选特定段的位置和速度 132. xSel = xCur(selS); 133. ySel = yCur(selS); 134. vxSel = vxCur(selS); 135. vySel = vyCur(selS); 137. % 绘制速度向量图 138. figure('Position', [100 + 200\*(k-1), 100, 800, 600]); % 偏移窗口位置以便查看所有图形 139. quiver(xSel, ySel, vxSel, vySel, 0.5, 'b'); 140. title(sprintf('速度向量图：时间 %d 秒', selT(k))); 141. xlabel('X 坐标 (米)'); 142. ylabel('Y 坐标 (米)'); 143. axis equal; 144. grid on; 146. % 保存图形 147. saveas(gcf, sprintf('SpeedVectors\_%d秒.png', selT(k))); 148. end |

|  |
| --- |
| 附录3 |
| Matlab 解决问题二 |
| 1. clc; clear; close all; 3. % 定义常量 4. Lh = 3.41;    % 龙头长度（米） 5. Lb = 2.2;     % 龙身长度（米） 6. Leff = Lb - 2\*0.275; % 有效长度（米） 7. Lext = 0.275; % 外延长度（米） 8. Sp = 0.55;    % 螺距（米） 9. a = Sp / (2\*pi); % 螺线参数 10. Vh = 1;        % 龙头速度（米/秒） 11. T = 500;       % 模拟总时间（秒） 12. dt = 1;        % 时间步长（秒） 13. N = 223;       % 板凳总数 14. tol = 1;       % 碰撞检测容差 15. theta\_s = 16\*2\*pi; % 起始角度（第16圈） 17. % 初始化数组 18. t = 0:dt:T; 19. th = zeros(N, length(t)); 20. x = zeros(N, length(t)); 21. y = zeros(N, length(t)); 22. vx = zeros(N, length(t)); 23. vy = zeros(N, length(t)); 25. % 初始化角度和位置 26. th(1) = theta\_s; 27. x(1) = a \* th(1) \* cos(th(1)); 28. y(1) = a \* th(1) \* sin(th(1)); 30. % 计算后续板凳的位置 31. for i = 2:N 32. if i == 2 33. ds = Lh; 34. else 35. ds = Leff; 36. end 37. r\_prev = a \* th(i-1); 38. r\_next = r\_prev + ds / (2\*pi\*(i-1 + theta\_s/(2\*pi))); 39. th\_next = atan2(r\_next, a); 40. delta\_th = ds / (r\_prev + ds/2); 41. th(i) = th(i-1) + delta\_th; 42. x(i) = (r\_prev + ds/2) \* cos(th(i)); 43. y(i) = (r\_prev + ds/2) \* sin(th(i)); 44. end 46. % 主循环 47. col\_time = -1; 48. for j = 2:length(t) 49. delta\_s = Vh \* dt; 50. delta\_th = delta\_s ./ (a \* th(:, j-1)); 51. th(:, j) = th(:, j-1) - delta\_th; 52. x(:, j) = a \* th(:, j) .\* cos(th(:, j)); 53. y(:, j) = a \* th(:, j) .\* sin(th(:, j)); 54. vx(:, j) = (x(:, j) - x(:, j-1)) / dt; 55. vy(:, j) = (y(:, j) - y(:, j-1)) / dt; 57. % 碰撞检测 58. seg\_start = [x(:, j) - Lext \* cos(th(:, j)), y(:, j) - Lext \* sin(th(:, j))]; 59. seg\_end = [x(:, j) + (Lb + Lext) \* cos(th(:, j)), y(:, j) + (Lb + Lext) \* sin(th(:, j))]; 60. [intersect, ~] = lineSegmentIntersect(reshape([seg\_start, seg\_end]', 4, [])'); 61. intersect = reshape(intersect, N, N); 62. intersect = triu(intersect, 1); 63. if any(intersect(:)) 64. col\_time = t(j); 65. break; 66. end 67. end 69. % 保存和显示结果 70. j\_final = length(t); 71. result = zeros(N, 3); 72. result(:,1) = x(:, j\_final); 73. result(:,2) = y(:, j\_final); 74. result(:,3) = sqrt(vx(end, j\_final)^2 + vy(end, j\_final)^2); 75. writematrix(round(result, 6), 'result2.xlsx'); 77. % 特定板凳的位置和速度 78. selected = [1, 2, 52, 102, 152, 202, 223]; 79. pos\_table = zeros(2\*length(selected), 1); 80. vel\_table = zeros(length(selected), 1); 81. for i = 1:length(selected) 82. idx = selected(i); 83. pos\_table(2\*i-1) = x(idx, j\_final); 84. pos\_table(2\*i) = y(idx, j\_final); 85. vel\_table(i) = sqrt(vx(idx, j\_final)^2 + vy(idx, j\_final)^2); 86. end 88. % 打印和保存表格 89. disp('位置：'); 90. disp(round(pos\_table, 6)); 91. disp('速度：'); 92. disp(round(vel\_table, 6)); 93. disp(['碰撞时刻：', num2str(col\_time), ' 秒']); 95. % 绘制碰撞时刻的板凳龙形状 96. figure; 97. plot(x(:, j\_final), y(:, j\_final), 'b-', 'LineWidth', 2); 98. hold on; 99. plot(x(1, j\_final), y(1, j\_final), 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'black'); 100. plot(x(end, j\_final), y(end, j\_final), 'go', 'MarkerSize', 10, 'MarkerFaceColor', 'r'); 101. title('碰撞时刻的板凳龙形状'); 102. xlabel('X 坐标 (米)'); 103. ylabel('Y 坐标 (米)'); 104. axis equal; 105. grid on; 107. % 线段相交检测函数 108. function [I,P] = lineSegmentIntersect(XY) 109. X1 = XY(:,1); Y1 = XY(:,2); 110. X2 = XY(:,3); Y2 = XY(:,4); 111. [X1, X3] = meshgrid(X1, X1); 112. [Y1, Y3] = meshgrid(Y1, Y1); 113. [X2, X4] = meshgrid(X2, X2); 114. [Y2, Y4] = meshgrid(Y2, Y2); 115. D = (X1-X2).\*(Y3-Y4) - (Y1-Y2).\*(X3-X4); 116. ua = ((X1-X3).\*(Y3-Y4) - (Y1-Y3).\*(X3-X4)) ./ D; 117. ub = ((X1-X3).\*(Y1-Y2) - (Y1-Y3).\*(X1-X2)) ./ D; 118. I = (ua >= 0 & ua <= 1 & ub >= 0 & ub <= 1); 119. P = [X1 + ua.\*(X2-X1), Y1 + ua.\*(Y2-Y1)]; 120. end |

|  |
| --- |
| 附录4 |
| Matlab 解决问题三 |
| 1. clear; close all; clc; 2. warning off 3. ts = 1;             % 时间步长（秒） 4. pitch\_range = (50:-0.5:40) \* 1e-2; % 螺距取值范围（米） 5. min\_ang = zeros(size(pitch\_range)); % 每个螺距对应的最小角度  8. for p\_idx = 1:length(pitch\_range) 9. p = pitch\_range(p\_idx); 10. k = p / (2\*pi);      % 螺线方程系数 11. hl = 341e-2;         % 龙头长度（米） 12. hh\_d = hl - 27.5e-2\*2;% 龙头把手距离 13. bl = 220e-2;         % 龙身和龙尾长度（米） 14. bh\_d = bl - 27.5e-2\*2;% 其他凳子把手距离 16. % 确定初始位置 17. init\_turns = ceil(4.5/p) + 3; 18. init\_ang = 2\*pi\*init\_turns; 20. coll\_flag = 0; 21. iter\_count = 0; 23. n\_segs = 223; % 龙头+龙身+龙尾总数 24. x\_pos = nan(n\_segs+1, 3); 25. y\_pos = nan(n\_segs+1, 3); 26. ang\_data = nan(n\_segs+1, 3); 27. ang\_data(1,3) = init\_ang; 29. % 盘入模拟 30. while coll\_flag == 0 31. iter\_count = iter\_count + 1; 32. x\_pos(:,1) = x\_pos(:,3); 33. y\_pos(:,1) = y\_pos(:,3); 34. ang\_data(:,1) = ang\_data(:,3); 36. % 求解龙头运动 37. tspan = [0, ts/2, ts]; 38. [~, new\_angs] = ode45(@(t,theta) -1/(k\*sqrt(1+theta^2)), tspan, ang\_data(1,1)); 39. new\_x = k \* new\_angs .\* cos(new\_angs); 40. new\_y = k \* new\_angs .\* sin(new\_angs); 42. % 更新位置和角度 43. x\_pos(1,:) = new\_x; 44. y\_pos(1,:) = new\_y; 45. ang\_data(1,:) = new\_angs; 47. % 计算龙身和龙尾位置 48. for t\_idx = 2:length(tspan) 49. for seg\_idx = 2:n\_segs+1 50. h\_d = hh\_d\*(seg\_idx<=2) + bh\_d\*(seg\_idx>2); 51. new\_ang = solve\_angle(p, x\_pos(seg\_idx-1,t\_idx), y\_pos(seg\_idx-1,t\_idx), ang\_data(seg\_idx-1,t\_idx), h\_d); 52. ang\_data(seg\_idx,t\_idx) = new\_ang; 53. x\_pos(seg\_idx,t\_idx) = k \* new\_ang \* cos(new\_ang); 54. y\_pos(seg\_idx,t\_idx) = k \* new\_ang \* sin(new\_ang); 55. end 56. end 58. % 碰 撞 59. for seg\_idx = 1:round(n\_segs/2) 60. x1 = x\_pos(seg\_idx,2); x2 = x\_pos(seg\_idx+1,2); 61. y1 = y\_pos(seg\_idx,2); y2 = y\_pos(seg\_idx+1,2); 62. ang1 = ang\_data(seg\_idx,2); 63. ang2 = ang\_data(seg\_idx+1,2); 65. outer\_idxs = find((ang1+2\*pi-ang\_data(:,2))>0); 66. outer\_idxs = outer\_idxs(max(1,end-2):end); 67. inner\_idxs = find(ang\_data(:,2)-(ang2+2\*pi)>0); 68. if isempty(inner\_idxs) 69. break; 70. else 71. inner\_idxs = inner\_idxs(1:min(3,length(inner\_idxs))); 72. end 73. check\_idxs = outer\_idxs(1):inner\_idxs(end); 75. for c\_idx = 1:length(check\_idxs)-1 76. pt1 = [x\_pos(check\_idxs(c\_idx),2); y\_pos(check\_idxs(c\_idx),2)]; 77. pt2 = [x\_pos(check\_idxs(c\_idx+1),2); y\_pos(check\_idxs(c\_idx+1),2)]; 78. coll = check\_intersection(head\_length\*(seg\_idx<=1)+body\_length\*(seg\_idx>1), [x1;y1], [x2;y2], body\_length, pt1, pt2, 10, 20); 79. if ~isempty(coll) 80. coll\_flag = 1; 81. break; 82. end 83. end 84. if coll\_flag == 1 85. break; 86. end 87. end 89. % 检查到达调头空间边界 90. if sqrt(x\_pos(1,end)^2 + y\_pos(1,end)^2) <= 4.5 91. coll\_flag = 1; 92. end 94. fprintf('当前螺距: %.2f m, 迭代步数: %d\n', p, iter\_count); 95. end 97. min\_ang(p\_idx) = ang\_data(1,end); 98. end 99. save('problem3\_tmp\_data'); 101. % 找到最优螺距（注意：需要定义theoretical\_min\_angle） 102. [~, optimal\_idx] = min(abs(min\_ang - theoretical\_min\_angle)); 103. optimal\_pitch = pitch\_range(optimal\_idx); 104. fprintf('最优螺距: %.4f m\n', optimal\_pitch); 106. % 辅助函数...（保持不变） 107. %% 辅助函数 108. function new\_angle = solve\_angle(pitch, x1, y1, angle1, distance) 109. spiral\_coeff = pitch / (2\*pi); 110. angle\_func = @(angle) (spiral\_coeff\*angle.\*cos(angle)-x1).^2 + (spiral\_coeff\*angle.\*sin(angle)-y1).^2 - distance^2; 111. options = optimoptions('fsolve', 'Display', 'off'); 112. new\_angle = fsolve(angle\_func, angle1+0.1, options); 113. while new\_angle <= angle1 || abs(spiral\_coeff\*new\_angle-spiral\_coeff\*angle1) > pitch/2 114. new\_angle = fsolve(angle\_func, new\_angle+0.1, options); 115. end 116. end |

|  |
| --- |
| 附录5 |
| Matlab 解决问题四 |
| 1. clc; clear; close all; 3. sp = 1.7; % 螺距（米） 4. sc = sp / (2\*pi); % 螺线系数 5. hl = 341e-2; % 龙头长度 6. hhd = hl - 27.5e-2\*2; % 龙头把手孔距 7. bl = 220e-2; % 龙身和龙尾长度 8. bhd = bl - 27.5e-2\*2; % 凳子把手孔距 9. hv = 1; % 龙头速度 11. th = 5\*2\*pi:-0.01:0; % 角度范围 12. r = sc \* th; % 螺线半径 13. xin = r .\* cos(th); % 盘入线x坐标 14. yin = r .\* sin(th); % 盘入线y坐标 16. figure('Name', '盘入盘出螺线及调头'); 17. plot(xin, yin, 'r-', 'LineWidth', 0.01); % 盘入螺线 18. hold on; 20. th\_out = th - pi; % 盘出螺线角度 21. rout = sc \* (th\_out + pi); % 盘出螺线半径 22. xout = rout .\* cos(th\_out); % 盘出线x坐标 23. yout = rout .\* sin(th\_out); % 盘出线y坐标 24. plot(xout, yout, 'b-', 'LineWidth', 0.01); % 盘出螺线 26. % 绘制调头区域 27. tr = 4.5; % 调头区半径 28. th\_c = linspace(0, 2\*pi, 100); 29. xc = tr \* cos(th\_c); % 圆x坐标 30. yc = tr \* sin(th\_c); % 圆y坐标 31. plot(xc, yc, 'k--', 'LineWidth', 2); % 绘制调头区域 33. xlabel('x (m)'); 34. ylabel('y (m)'); 35. title('盘入盘出螺线及调头'); 36. legend('盘入螺线', '盘出螺线', '边界'); 37. axis equal; 38. grid on; 40. th\_in\_end = tr / sc; % 盘入螺线终点角度 41. th\_out\_start = th\_in\_end - pi; % 盘出螺线起点角度 43. % 计算盘入螺线终点斜率 44. slope\_end = (sc\*sin(th\_in\_end) + tr\*cos(th\_in\_end)) / ... 45. (sc\*cos(th\_in\_end) - tr\*sin(th\_in\_end)); 47. th\_max1 = atan(-1/slope\_end) + pi; % 第一段圆弧 48. th\_delta = atan(tan(th\_in\_end)) + pi - th\_max1; % 角度差 50. rc1\_c2 = tr / cos(th\_delta); % 圆心距 51. rc2 = rc1\_c2 / 3; % 第二段圆弧半径 52. rc1 = rc2 \* 2; % 第一段圆弧半径 54. phi = 2 \* th\_delta; % 圆弧角度 55. arc\_len\_c1 = rc1 \* (pi - phi); % 第一段圆弧长度 56. arc\_len\_c2 = rc2 \* (pi - phi); % 第二段圆弧长度 58. th\_min1 = th\_max1 - arc\_len\_c1 / rc1; % 第一段圆弧最小角度 59. th\_min2 = th\_min1 - pi; % 第二段圆弧最小角度 60. th\_max2 = th\_min2 + arc\_len\_c2 / rc2; % 第二段圆弧最大角度 62. % 第一段圆弧圆心坐标 63. xc1 = tr \* cos(th\_in\_end) + rc1 \* cos(th\_max1 - pi); 64. yc1 = tr \* sin(th\_in\_end) + rc1 \* sin(th\_max1 - pi); 66. % 第二段圆弧圆心坐标 67. xc2 = tr \* cos(th\_out\_start) - rc2 \* cos(th\_max2); 68. yc2 = tr \* sin(th\_out\_start) - rc2 \* sin(th\_max2); 70. % 绘制调头曲线 71. t1 = linspace(th\_min1, th\_max1, 50); 72. xarc1 = xc1 + rc1 \* cos(t1); 73. yarc1 = yc1 + rc1 \* sin(t1); 74. plot(xarc1, yarc1, 'g-', 'LineWidth', 2); 76. t2 = linspace(th\_min2, th\_max2, 50); 77. xarc2 = xc2 + rc2 \* cos(t2); 78. yarc2 = yc2 + rc2 \* sin(t2); 79. plot(xarc2, yarc2, 'm-', 'LineWidth', 2); 81. plot([xc1, xc2], [yc1, yc2], 'ko', 'MarkerFaceColor', 'k'); % 绘制圆心 83. legend('盘入螺线', '盘出螺线', '调头边界', '调头曲线1', '调头曲线2', '圆心'); 85. ts = 1; % 时间步长（秒） 86. tot\_seg = 223; % 总段数 88. % 盘入阶段 89. tin = 0:ts:100; 90. dtheta\_dt = @(t, th) 1 ./ (sc \* sqrt(1 + th.^2)); 91. [tin, th\_in] = ode45(dtheta\_dt, tin, th\_in\_end); 92. xhead\_in = sc \* th\_in .\* cos(th\_in); 93. yhead\_in = sc \* th\_in .\* sin(th\_in); 95. % 调头阶段 96. tturn = 0:ts:(arc\_len\_c1 + arc\_len\_c2); 97. th\_c1 = th\_max1 - tturn(tturn <= arc\_len\_c1) / rc1; 98. th\_c2 = th\_min2 + (tturn(tturn > arc\_len\_c1) - arc\_len\_c1) / rc2; 99. xhead\_turn = [rc1 \* cos(th\_c1) + xc1, rc2 \* cos(th\_c2) + xc2]; 100. yhead\_turn = [rc1 \* sin(th\_c1) + yc1, rc2 \* sin(th\_c2) + yc2]; 102. % 盘出阶段 103. tout = 0:ts:(100 - length(tturn)\*ts); 104. dtheta\_dt\_out = @(t, th) 1 ./ (sc \* sqrt(1 + (th + pi).^2)); 105. [tout, th\_out] = ode45(dtheta\_dt\_out, tout, th\_out\_start); 106. xhead\_out = sc \* (th\_out + pi) .\* cos(th\_out); 107. yhead\_out = sc \* (th\_out + pi) .\* sin(th\_out); 109. % 合并所有轨迹 110. ttotal = [-flip(tin); tturn'; tout + tturn(end)] - 100; 111. xhead\_total = [flip(xhead\_in); xhead\_turn'; xhead\_out]; 112. yhead\_total = [flip(yhead\_in); yhead\_turn'; yhead\_out]; 114. % 计算速度 115. vx = diff(xhead\_total) / ts; 116. vy = diff(yhead\_total) / ts; 117. vtotal = sqrt(vx.^2 + vy.^2); 119. % 辅助函数（保持原名，但内部变量名简化） 120. function [x, y] = calculate\_next\_point\_spiral\_in(sp, x1, y1, th1, d) 121. k = sp / (2\*pi); 122. fun = @(th) (k\*th.\*cos(th)-x1).^2 + (k\*th.\*sin(th)-y1).^2 - d^2; 123. options = optimset('Display', 'off'); 124. th = fsolve(fun, th1 + 0.1, options); 125. while th <= th1 || abs(k\*th - k\*th1) > sp/2 126. th = fsolve(fun, th + 0.1, options); 127. end 128. x = k \* th \* cos(th); 129. y = k \* th \* sin(th); 130. end 132. function [x, y] = calculate\_next\_point\_spiral\_out(spiral\_pitch, x1, y1, theta1, d) 133. k = spiral\_pitch / (2\*pi); % 计算螺线系数 134. % 定义距离方程，该方程描述了在螺线上距离(x1, y1)为d的点 135. fun = @(theta) ((k\*(theta+pi).\*cos(theta) - x1).^2 + ... 136. (k\*(theta+pi).\*sin(theta) - y1).^2 - d^2); 137. options = optimoptions('fsolve', 'Display', 'off'); 138. % 初始猜测为theta1 - 0.1，因为螺线是向外扩展的，所以尝试减小角度 139. theta = fsolve(fun, theta1 - 0.1, options); 141. % 检查解是否在预期范围内，如果不是，则重新求解 142. while theta >= theta1 || abs(k\*theta - k\*theta1) > spiral\_pitch/2 143. theta = fsolve(fun, theta - 0.1, options); 144. end 146. % 根据求解得到的theta计算下一个点的坐标 147. x = k \* (theta + pi) \* cos(theta); 148. y = k \* (theta + pi) \* sin(theta); 149. end 150. function [x, y] = calculate\_next\_point\_arc(x1, y1, theta1, d, r, xc, yc) 152. % 计算角度增量，这里使用了正弦定理的近似（假设d远小于r） 153. delta\_theta = 2 \* asin(d / (2 \* r)); 154. % 计算新的角度 155. theta = theta1 + delta\_theta; 156. % 根据新的角度和圆心计算点的坐标 157. x = xc + r \* cos(theta); 158. y = yc + r \* sin(theta); 159. end |

|  |
| --- |
| 附录6 |
| Matlab 解决问题五 |
| 1. % 问题5：确定龙头的最大行进速度 2. clear 3. load problem4\_save\_data 4. % 计算速度 5. velocity\_x = diff(x\_positions, 1, 2) / time\_step; 6. velocity\_y = diff(y\_positions, 1, 2) / time\_step; 7. % 注意：这里diff操作会导致数组大小减少，需要确保后续处理与此兼容 9. % 计算合速度 10. velocity = sqrt(velocity\_x.^2 + velocity\_y.^2); 12. % 每个板凳的最大速度（考虑diff减少了数组大小，需要处理边界） 13. % 假设每个板凳的数据是一个行，时间步长是列 14. num\_benches = size(x\_positions, 1); 15. num\_timesteps = size(x\_positions, 2) - 1; % diff后减少一个时间步 16. max\_velocities = zeros(num\_benches, 1); 17. for i = 1:num\_benches 18. max\_velocities(i) = max(sqrt(velocity\_x(i,:).^2 + velocity\_y(i,:).^2)); 19. end 21. % 所有板凳中的最大速度 22. overall\_max\_velocity = max(max\_velocities); 24. % 最大速度与当前龙头速度的比值 25. velocity\_ratio = overall\_max\_velocity / head\_speed; 27. % 允许的最大龙头速度 28. max\_allowed\_speed = 2 / velocity\_ratio; 30. fprintf('最大龙头速度: %.2f m/s\n', max\_allowed\_speed); 32. % 调整速度（这里不直接修改原速度数据，而是计算新的速度矩阵） 33. new\_velocities = zeros(size(velocity)); % 保持原速度矩阵的形状 34. for i = 1:num\_benches 35. for j = 1:num\_timesteps 36. % 只对原速度不为零的地方进行调整（如果考虑速度可能为0的情况） 37. if velocity(i,j) > 0 38. new\_velocities(i,j) = velocity(i,j) \* (max\_allowed\_speed / head\_speed); 39. % 但实际上，我们应该使用全局的速度限制2 m/s来直接限制每个板凳的速度 40. % 如果上面的乘法导致速度超过2 m/s，则直接设为2 m/s 41. if new\_velocities(i,j) > 2 42. new\_velocities(i,j) = 2; 43. end 44. end 45. end 46. end 48. % 验证新的速度矩阵是否所有元素都不超过2 m/s（这步其实是多余的，因为我们在上一步已经处理了） 49. % 但为了保持代码的一致性，还是写上 50. if all(new\_velocities(:) <= 2) 51. disp('验证通过：所有板凳的速度均不超过2 m/s'); 52. else 53. warning('理论上不应出现此警告，因为已在计算中限制了速度'); 54. end 56. % 可视化新旧速度对比（这里只展示每个板凳的最大速度） 57. figure('Name', '新速度与原速度对比'); 58. hold on; 59. plot(max\_velocities, 'bo-', 'DisplayName', '原速度方案的最大速度'); % 原速度的最大值 60. plot(max(new\_velocities, [], 2), 'r\*-', 'DisplayName', '新速度方案的最大速度'); % 注意这里可能需要调整以正确显示 61. % 但由于new\_velocities可能有时间步长的维度，直接max可能不是预期的。我们可以直接绘制new\_velocities的每行最大值 62. plot([1, num\_benches], [2, 2], 'k--', 'LineWidth', 2, 'DisplayName', '速度限制'); 63. xlabel('板凳编号'); 64. ylabel('最大的速度 (m/s)'); 65. legend show; |