基于有限差分法的“球-鼓-绳”力学模型

摘 要

“同心协力”项目可以增加团队成员合作意识，培养团队精神。本文通过对“球-鼓-绳”三者之间的关系进行研究，以使连续颠球次数最多为目标，给出团队成员之间的协作策略

针对问题一，分别利用隔离法和整体法对球和鼓进行**受力分析**，分析“球-鼓”运动规律，发现若鼓在竖直方向上与球一同做周期运动，且球在水平位置与上升状态的鼓碰撞，此时连续颠球次数最多。由**动量定理**和**动能定理**求解得“球-鼓”运动的$v-t$图像，通过**面积法**，解得颠球高度为45cm，符合题目要求。对“鼓-绳”进行分析，得出绳子拉力变化情况和**最佳协作策略**：队员用力方向沿绳，同时出力，发力大小相同，使得鼓始终保持水平运动，球始终保持竖直运动，二者运动频率一致

针对问题二，设置空间直角坐标系和平动坐标系。基于“鼓-绳”的受力分析，建立同心鼓的**力学模型**：定位绳子的施力点和受力点，运用**转动定律**求得鼓在平动坐标系上的角加速度和在竖直方向上的平动加速度。通过**有限差分法**，求得鼓在平动坐标轴上的角位置微量，基于仿射变换等坐标变化，实现迭代运算。从而对倾角的变化给出严格的求解。根据题目条件，求解的鼓面倾斜角度参见表一

针对问题三，因为鼓面产生倾斜角度影响了最佳策略，在鼓未与球碰撞的情况下，需要对“鼓-绳”进行调整，使得鼓恢复水平运动状态。采用**控制变量法**，基于用力大小、发力时机两个因素对模型进行讨论，得到其对鼓面倾斜角度的影响。文中给予了相应的调整策略

针对问题四，对“球-鼓”进行受力分析，得到若要使球的运动恢复竖直状态，第二次碰撞时鼓面的倾斜角度。基于二问模型，调节人数绳长等参数并对坐标轴进行旋转变化。采用**网格搜索算法**，寻找所有队员的发力时机及力度，结果参见表二

本文在“**鼓面水平运动，球垂直运动**”的最佳策略下，考虑到了队员不能精确控制发力时机和力度使得鼓面倾斜的问题，分析了**两种情况**：若鼓面倾斜后，球未与之碰撞，则需要对鼓面进行调整，使其水平；若鼓面倾斜后，已经与球碰撞，则需要先恢复球运动的竖直状态，然后再恢复鼓运动的水平状态，并给出了相应调整策略

关键词：网格搜索算法 转动定律 有限差分法 球-绳-鼓

一. 问题的重述

1.1 问题的背景

同心鼓击球游戏是在素质拓展活动中经常见到的破冰游戏，意在考验队员们的默契和团队协作能力。多根等长的绳子与鼓身相连且沿鼓周均匀分布，游戏队员手持绳子的末端，通过恰当的用力使鼓做有规律的运动，从而尽量达到用鼓颠球的次数最多的目的。其中，队员自身的用力的角度、方向、力度，以及队员之间的相互配合对鼓运动的控制显得尤为重要，成为游戏成败的关键。

1.2 问题的提出

根据题中物理情景的描述，我们已经得知所选用同心鼓的鼓面直径、鼓身高度以及鼓身质量分别为 40cm，22cm，3.6kg。排球的质量为 270g。对游戏队员的要求为：人数至少为 8 人，且队员间至少存在着 60cm 的最小距离。同时，题目规定了球起始下落高度，以及游戏结束的条件，最终目的是使得连续颠球的次数最多。

本问题考察的是利用力学和数学分析手段对同心鼓和球的运动进行分析，从而以实现颠球次数最多的目的。因此，我们考虑解决以下 4 个问题：

（1）在参与游戏每个队员都可以精确控制其用力方向、时机和力度的理想情况下，求解可以使得颠球次数最多的团队获胜策略。

（2）假设实际情况中，队员的人数，绳长，鼓面初始位置，以及队员的发力和用力大小都已给出。建立模型求解出队员的发力时机和力度与鼓面倾斜角度的关系，并求出 0.1s 时的鼓面倾斜角度。

（3）根据问题 2 中所假设的实际情形，若仍然想使颠球的次数最多，是否需要对问题 1 给出的策略进行调整，如何调整？

（4）在人数为 10，绳长为 2m 的给定条件下，并且球的反弹高度，反弹方向和偏离倾角已经确定的情况下，为了将球调整为竖直方向，队员的发力时机和力度应当如何调整，并分析在实际中的应用效果如何。

二. 问题的分析

2.1 问题一的分析

问题一要求在人能够精准控制用力方向、时机和角度的条件下求出能使得颠球次数最多的协作策略以及该策略下的颠球高度。针对这一问题，我们首先利用隔离法对对鼓和球进行受力分析，得到鼓和球在各自独立运动时所受的作用力，进而推到分析出球和鼓的运动$v-t$图。其次，在球和鼓面发生碰撞时，认为球与鼓面发生了完全弹性碰撞。此外，根据球与鼓在碰撞过程中遵循能量

守恒定理和动量守恒定律，可以得到二者碰撞后的速度。在球与鼓相撞位置问题的求解中，当鼓运动到水平位置时，速度达到最大，此时球若与鼓碰撞，可以获得最大的速度，从而可以更加容易地达到上升高度 40cm 的要求。

在理想状态下，鼓在竖直方向上简谐运动，要制定策略使其频率尽可能的与球运动的频率一致，从而可使球与鼓每一次碰撞的位置相同。团队成员需要控制用力方向使得鼓面保持水平，同时出力以避免鼓沿某个受力方向偏移，力度应根据鼓和球此时的位置而即使调整，最终结合这两部分的分析，可以整合出球和鼓的实际运动方程，从而可以得出该策略下的颠球高度。

2.2 问题二的分析

问题二要求给出在实际情况下，团队成员的用力力度大小以及用力时机的微小差别造成的鼓面倾角的变化关系。我们定义一个时钟为 1/1000s，若要研究鼓面倾角的变化，我们须知道对应于每一个时钟鼓面的运动参数：角速度、角加速度、平动加速度，质心运动速度。因此，我们将对鼓进行动力学分析，从平动和转动两个方面进行分析。

我们首先建立坐标系，对各个时刻施力点和受力点的位置进行定位。继而我们在同心鼓的结构中心点建立平动坐标系。对同心鼓进行受力分析，得到平动加速度，同时可以得到关于平动坐标系的转动力矩及相关参量，分别研究同心鼓的平动运动情况与转动运动情况，通过有限差分法，在每一个时钟，都将各参量进行整合更新迭代。根据仿射变换，研究受力点位置的变化。对倾角的变化给出严格的求解。

2.3 问题三的分析

因实际情景下，人不能精确控制时机和力度，从而使得鼓面产生一定的倾角。本文通过控制变量法，分别基于用力大小、发力时机两个因素对模型进行讨论，得到其对鼓面倾斜角度的影响。

基于上述影响效果，为使其尽可能的达到问题一碰撞时的效果，我们对队员的安排给以一定的优化调整。

2.4 问题四的分析

对球第一次与鼓碰撞的情况进行受力分析，得到二者的速度。旋转第二问的直角坐标系，使得球碰撞后的倾斜方向在水平面的投影为 x 轴正方向，并且对问题二中绳的受力点和施力点进行调整。

分析第一次撞击前后，以及第二次撞击前后，球和鼓的运动规律得到，分析得到第二次撞击点的位置。计算第二次撞击时，人所控制的，鼓面的倾斜角

度，使得鼓面倾斜一定角度使得球碰撞后为竖直状态弹跳。

基于上述模型，为使鼓面倾斜一定角度，通过网格搜索算法，可以给出所有队员的发力时机及力度。

三． 模型假设

1. 忽略空气阻力；
2. 忽略人的心理因素的影响；
3. 假设游戏成员的位置不发生移动；
4. 假设排球与鼓面的碰撞没有能量的损失；
5. 假设同心鼓牛皮为刚体，不发生弹性形变；
6. 假设排球的碰撞后的反射符合光线的反射定律；

四． 符号说明

符号	说明
M	鼓的质量
m	排球的质量
F_i	第 i 根绳子上的拉力
α	为绳子与竖直方向的夹角
l	绳子的长度
n	参与游戏队员的数量
F_{back}	回复力
v_b	理想情况下，球和鼓竖直撞击时的速度
a_i	第 i 个人施力点坐标
b_i	第 i 个拉力在鼓上的作用点
J	转动惯量
w	角速度
K_i	第 i 个时针，中心点的坐标
v_{stant}	排球做斜抛运动的速度
α	鼓绕平动坐标轴转动的角加速度
a	鼓的平动加速度

注：其他符号在下文中给出说明。

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 静力学模型的建立

建立空间直角坐标系如下：我们令所有绳子拉至水平时，同心鼓结构中心的位置为原点，令竖直向上方向为 z 轴的正方向，定义 x 轴的正方向为所有绳子水平时，与编号为 1 号的绳子的方向相同。

同心鼓的结构是鼓面直径为 40cm，鼓身高度为 22cm 的圆柱体，在初始状态下，鼓在自身的重力，绳子拉力的作用，在水平面以下一定高度保持静止状态下，根据弹性力学的知识可知，绳子的拉力始终是沿着绳的方向的。由此，我们可知沿绳拉力的分布是以鼓身的轴线为对称轴，空间对称分布，拉力的作用点位于鼓中心点所在水平面的外圆边线上。

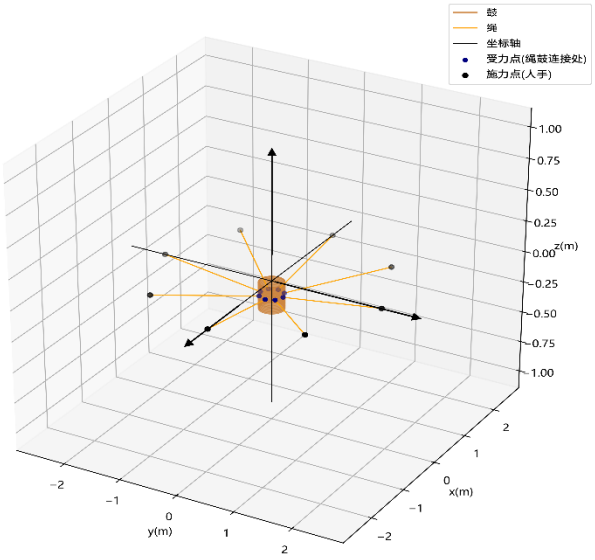

图 1. 鼓在所建空间坐标系中的位置及空间力系分析

因为鼓所受拉力是在中心点水平面外圆线上空间对称分布的，我们假设鼓上总共有 n 条绳子，绳子两两之间是相互对称的，我们取出这一对称力所组成的平面，将空间力系转化成平面力系进行分析，简化了分析的难度，我们继而根据平面汇交力系的相关知识，将各拉力的作用点等效在鼓的几何中心处，如下图所示：

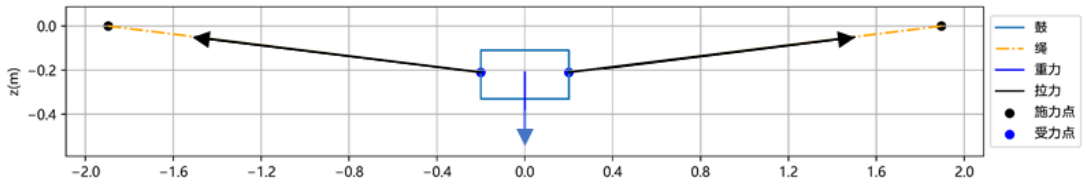

图 2. 作用于鼓身对称力的平面受力分析

在平面正交力系下，水平方向和竖直方向分别列出静力学方程为：

$$\sum F_x = 0 \tag{1}$$

$$\sum F_z = F\cos\beta - \frac{mg}{n} \tag{2}$$

其中，我们特别注意到，由于鼓自身的重力分解到平面汇交力系之后变为 mg/n。

对于排球而言，其理想化的运动为只存在竖直方向上的移动，并且在运动的过程中，排球只受到重力的作用，在非碰撞时刻，排球所受的加速度始终为重力加速度，方向为竖直向下。

5.1.2 运动学模型的建立

1）理想情况假设

因为在理想的情况下，人们可以精准地控制用力的方向、时机和力度，得以实现颠球次数最多的目的，我们假设这样一种理想的情况：

鼓面是始终保持水平的，我们对施力者提出的要求，绳子上始终是有力的，并且力一定是沿着绳子的方向的，根据对称性的原则，我们可以知道成员的人数 n 必须为偶数，团队成员需要控制用力方向使得鼓面保持水平，同时出力以避免鼓沿某个受力方向偏移，每个队员的用力的时机和大小的变化都是同时的，保持着完全相同的一致性。

同心鼓只做在 z 轴方向上的上下移动。同理，排球也只有 z 轴方向上的移动。

我们假设同心鼓和排球的每一次相撞撞击点的位置都是不变的。并且通过分析，我们可以得知，当鼓的结构中心点沿 z 轴平动到所建立坐标系原点是，鼓具有向 z 轴正方向最大的速度，在此位置撞击，可以传递给排球最大的速度，因此可以使排球更加容易地达到题目中规定的 40cm 弹跳高度的要求，可以求得每一次撞击点的位置坐标均为(0,0,0.11)。

根据我们所假设的理想的情况，可以得出结论，只有当同心鼓和排球都做

有规律的周期运动时，并且二者的频率时完全吻合时，才能够保证我们设想理想情况的基本假设。

2）现对鼓的运动情况进行分析：

$$n*\sum F_z = nF\cos\beta - mg \tag{3}$$

其中，$\sum F_z$ 表示由一对对称力组成的平面中，这一对力在 z 轴方向上的合力。

$$F = F_s + F_d \tag{4}$$

公式(4)表示，我们将人施加给绳子的力拆解成了 F_s 和 F_d 了两个力组成，其中，F_s 表示一个大小保持恒定的力，只有它的方向在也是沿着绳子的方向指向施力者，并且随着绳子方向的改变而改变。F_d 表示一个大小和方向都在改变的力，方向同样是沿着绳子的方向指向施力者，其大小的我们给出如下定义。

$$F_d = (+)\frac{mg}{n\cos\beta} / (-)\frac{mg}{n\cos\beta} \tag{5}$$

当鼓的结构中心点位于水平面以下时，取正；反之取负。我们通过把施力者施加的力 F 进行了拆分，使得 F_d 在竖直方向上的分量始终可以与重力平衡。

我们将（3），（4），（5）式联立求解可以得到：

$$n*\sum F_z = nF_s\cos\beta \tag{6}$$

因为 F_d 在竖直方向上的分量可以与重力平衡，因此，对于同心鼓其竖直方向上的合力的大小即为（6）式，方向始终指向坐标原点。

根据简谐运动的定义可知，当某物体进行简谐运动时，物体所受的力跟位移成正比，并且总是指向平衡位置的，简谐运动是一种周期性的运动，并且假若一个物体的运动是简谐运动，那么其位移与时间的关系，速度与时间的关系，均遵从正弦函数的规律。

在前文我们所给出的每个团队成员所施加力的大小和时机的控制方案下，假设同心鼓所做的运动为简谐运动，则定义其回复力为 F_{back}：

$$F_{back} = -n*\sum F_z = -nF_s\cos\beta \tag{7}$$

同心鼓在其运动方向上的位移为：

$$z = l*\cos\beta \tag{8}$$

联立（7），（8）两式可以得到回复力与位移的关系式为：

$$F_{back} = -\frac{nF_s}{l} * z \qquad (9)$$

式（9）表明在我们所给出的力的控制方案的前提下，同心鼓是满足简谐运动的判别条件的，因此，同心鼓在 z 轴上做的运动是具有正弦规律的周期运动。

我们已经完成了对同心鼓运动的分析，并得出结论，在我们所给出的力的控制方案的前提下，同心鼓在做简谐运动。

3）对排球的运动状况进行分析：

排球在向上运动的过程中，在做 $a=-g$ 的匀减速运动，排球在第一次反弹到达最高点后，向下的运动是自由落体运动。因此只要保证，排球与鼓面的每一次的碰撞点位置坐标都是不变的，我们就可以保证，排球向上运动和向下运动这两个运动是互为逆运动的。这一条件我们先前在理想条件的基本假设中已经提到，并且正如我们前文所证明的，同心鼓的运动是周期运动，因此碰撞点的位置坐标保持不变是可以做到的，因此我们可以声明，排球的运动也是周期运动。

在同心鼓与排球相撞时，根据完全弹性碰撞的定义，碰撞的时间极短，碰撞力很大，相比之下，外力是可以忽略不计的。因此在碰撞的过程中是满足动量守恒和能量守恒的，排球的质量为 270g，鼓的质量为 3.6kg，二者的质量之比为 0.075，是一个非常小的比例，于是我们根据能量守恒和动量守恒推导得出，在碰撞前后，同心鼓运动的速度和方向不变。碰撞后，排球的运动速度变为与同心鼓的运动速度相同，v_b 为鼓和球相撞时鼓的速度。

于是我们得到了同心鼓与排球运动的 $v-t$ 图像：

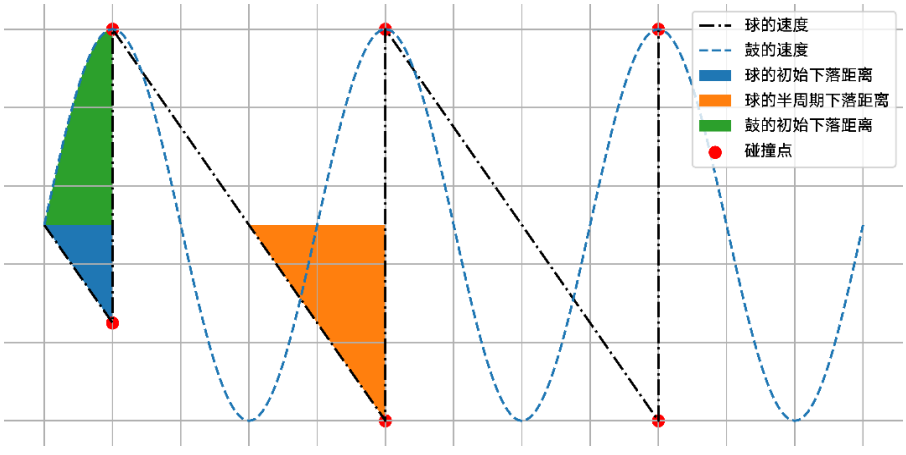

图 3. 排球、同心鼓 *v-t* 运动轨迹图

正如图中所示，蓝色虚线表示鼓的 $v-t$ 关系曲线，黑色点划线表示球的 $v-t$ 关系曲线，如图所示，球的速度在碰撞之后发生突变，变为与鼓的速度相同。球初始下落的距离为 1/4 个周期球下落的距离，在图中用蓝色的阴影部分

表示。橙色阴影表示一般情况下，球的 1/2 个周期下落的距离。记球初始下落的距离为 s_{blue}，球半周期下落的距离为 s_{orange}，同心鼓的初始下落距离为 s_{green}。

$$\begin{cases} \dfrac{s_{blue}}{s_{orange}} = \dfrac{1}{4} \\ \dfrac{s_{blue}}{s_{green}} = \dfrac{\frac{T}{4} * v_b}{\int_0^{\frac{T}{4}} v_b \sin(t)} \\ s_{blue} + s_{green} = 0.4 \end{cases} \tag{10}$$

求解方程组（11）可以得到，s_{orange}=0.45m，s_{orange} 也就是第一问中所要求的理想状况下该策略的颠球高度。

5.2 问题二的模型建立与求解

5.2.1 坐标轴的建立

1）固定坐标系的建立：

当绳子水平时，绳子在鼓上的端点与绳子在人手上的末端处于同一水平面，此时同心鼓的结构中心也位于该水平面内，我们定义此时的同心鼓结构中心点为空间坐标系的坐标原点，该平面定义为 XoY 平面， X 轴正方向指向第一个队员，Z 轴正方向竖直向上。

2）平动坐标系的建立：

相对于固定坐标系而言，我们建立平动坐标系的坐标原点 O' 点始终是固连在同心鼓的结构中心点，轴 Z' 与轴 Z 重合，坐标轴 X' 与 X 平行 Y' 与 Y 平行，建立这样的坐标系方便我们对鼓面的转动进行研究和分析。

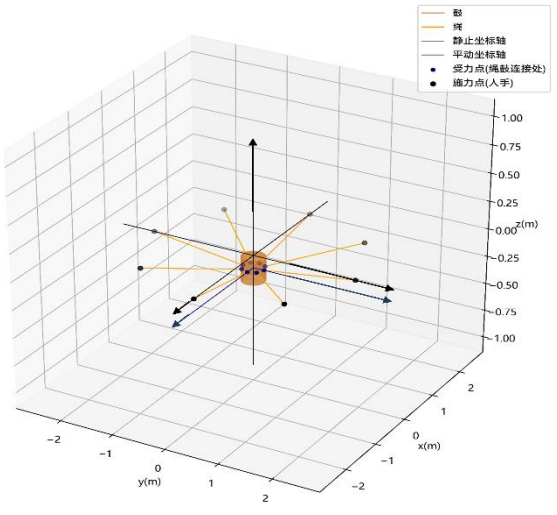

图 5. 固定坐标系和平动坐标系

如上图所示，黑色坐标轴表示固定坐标系，蓝色坐标轴表示平动坐标系，其原点是可以沿着 z 轴上下移动的。

5.2.2 力学模型的建立

1）转动力矩的求解

在固定坐标系下，我们可以得到第 i 个人拉绳子的施力的作用点坐标为：

$$a_i = \left(a_{i1}, a_{i2}, a_{i3}\right) \qquad i = 1, 2, ..., n \tag{11}$$

同理，在固定坐标系下，我们可以得到鼓所受第 i 个人拉力在鼓身的作用点的三维坐标为：

$$b_i = \left(b_{i1}, \mathrm{b}_{i2}, \mathrm{b}_{i3}\right) \qquad i = 1, 2, ..., n \tag{12}$$

我们已经确定人所施加拉力施力点和受力点，力的方向是沿着绳子的方向，绳子上若有拉力，则必然不会产生弯曲。进而，可以得到绳子上拉力的方向矢量：

$$\overrightarrow{c_n} = \frac{\overrightarrow{a_n} - \overrightarrow{b_n}}{\sqrt{|\overrightarrow{a_n}|^2 + |\overrightarrow{b_n}|^2}} \tag{13}$$

题中表格已经给出不同队员所施加力的大小的数值，我们将其记为$|F|$，结合绳子拉力的方向矢量，我们可以得到在该坐标轴下力的矢量为：

$$\overrightarrow{F_n} = \left|F\right| \cdot \overrightarrow{c_n} \tag{14}$$

在固定坐标系的参考下，我们可以求解出同心鼓的结构中心点的位置矢量$\vec{k}$时。同样地，我们可以得知在平动坐标系下，鼓的受力点到鼓中心的矢量$\overrightarrow{d_n}$为：

$$\overrightarrow{d_n} = \overrightarrow{b_n} - \vec{k} \tag{15}$$

根据力矩的定义可知，矢量叉乘运算可以得到编号拉力相对于平动坐标轴O'所产生的力矩：

$$\overrightarrow{M_n} = \overrightarrow{d_n} \times \overrightarrow{F_n} \tag{16}$$

力矩通过与平动坐标轴对应的单位向量的点乘，可以得到一个拉力在对于该坐标轴产生的力矩，我们将所有的力矩分量进行累加，就可以得到所有绳子作用在轴的合力矩分量，相对于X'轴和Y'轴，我们分别记为$M_{\mathrm{x}'}$和$M_{\mathrm{y}'}$：

$$M_{\mathrm{x}'} = \sum \overrightarrow{M_i} \cdot \overrightarrow{e_{\mathrm{x}'}} \qquad i = 1, 2, ..., n \tag{17}$$

$$M_{y'} = \sum \overrightarrow{M_i} \cdot \overrightarrow{e_{y'}} \qquad i = 1, 2, ..., n \tag{18}$$

至此，我们便完成了对鼓相对于平动参考系转动力矩的建模工作，这将用于我们对鼓的转动运动情况进行分析，求解转动加速度，以及角速度和角度。

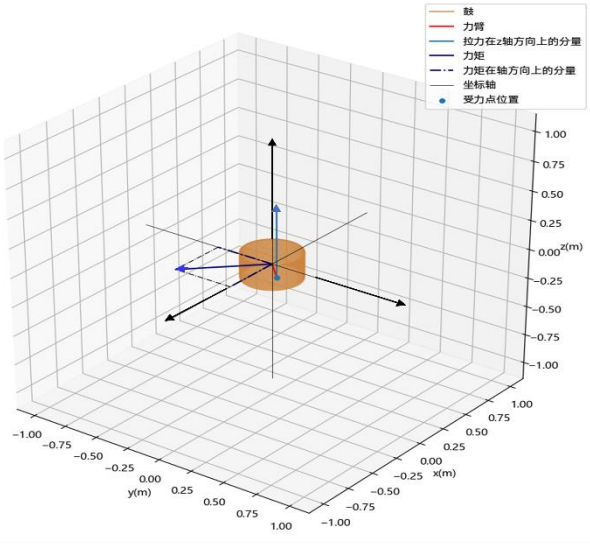

图 6. 力矩合成图

2）转动惯量和转动加速度的求解

对于同心鼓，求解其关于 X' 轴和 Y' 轴的转动惯量，可利用物理学中转动惯量的计算公式表达如下：

$$J_{x'} = J_{y'} = \rho \iiint_V R^2 dV \tag{19}$$

其中 ρ 表示同心鼓的密度，R 表示表示体积微元所在点与转动轴的垂直距离。

查阅文献可知，鼓的外周薄壁圆柱为木材，其密度为 $0.8g/cm^3$，其内部的介质为空气，可近似忽略空气的密度，上下圆面为薄牛皮，其质量相对于木架的质量来讲，也可以忽略不计。因此认为木材的质量为 3.6kg。记鼓外周薄壁木圆柱的厚度为 σ，可以有如下等式：

$$\int_{\sigma}^{0.2} \int_{0}^{2\Pi} \rho r d\theta dr = 3.6kg \tag{20}$$

可以解得 $\sigma = 0.18m$，代入转动惯量的计算公式，可得：

$$J_{x'} = J_{y'} = 2\rho \int_{0}^{0.11} \int_{0.18}^{0.2} \int_{0}^{2\Pi} (r^2 \cos\theta^2 + z^2) r d\theta dr dz \tag{21}$$

通过转动定律，可以求得其在 X' 轴和 Y' 轴的转动加速度：

$$\alpha_{x'} = \frac{M_{x'}}{J_{x'}} \quad \alpha_{y'} = \frac{M_{y'}}{J_{y'}} \tag{22}$$

进而可以求得 X' 轴和 Y' 轴旋转的角度

$$\frac{dw_{x'}}{dt}=\alpha_{x'} \qquad \frac{dw_{y'}}{dt}=\alpha_{y'} \tag{23}$$

$$\frac{d\theta_{x'}}{dt}=w_{x'} \qquad \frac{d\theta_{y'}}{dt}=v_{y'} \tag{24}$$

3）斜面倾角的求解模型的建立

在已知原平面的法向量 $\vec{n_1}$ 和 X' 轴和 Y' 轴理论上旋转的角度下，可以通过平面法向量定理，求得实际上该鼓面旋转的角度 θ：

$$\vec{l}_{x'}=(1,0,\tan\theta_{x'}) \qquad \vec{l_{y'}}=(0,1,\tan\theta_{y'}) \tag{25}$$

其中 $\vec{l}_{x'}$ 表示转动后的鼓面上与 YoZ 平面内的交线的一个向量，同理 $\vec{l}_{y'}$ 表示转动后的鼓面上与 XoZ 平面内的交线的一个向量。

$$\vec{n_2}=\vec{l_{x'}}\times\vec{l_{y'}} \tag{26}$$

求解出的 $\vec{n_2}$ 表示转动后鼓面的法向量，我们将用其来求解转动前后法向量的夹角，两平面法向量夹角的补角就是我们要求的鼓面转动的倾角。

$$\cos\theta=\frac{\left|\vec{n}_1\,\vec{n}_2\right|}{\left|\vec{n_1}\right|\left|\vec{n_2}\right|} \tag{27}$$

5.2.3 基于有限差分算法的力学模型求解

根据上述模型，结合题目所给条件，采用有限差分法对模型求解。算法步骤如下：

Step1：数据初始化：

输入初始时刻绳子的受力点和施力点，鼓的质量和此时鼓中心点的位置；

根据所求精度设计采样率，输入程序。

Step2：更新角速度：

计算在该时刻下绳子作用在 X' 轴和 Y' 轴上的合外力矩；

计算在该时刻下绳子作用在 X' 轴和 Y' 轴上的转动加速度；

因为时间极短，采用差商代替微商，得到该时刻 X' 轴和 Y' 轴的角速度为：

$$w_{x'}(t_2)=w_{x'}(t_1)+\alpha_{x'}(t_2-t_1) \tag{28}$$

$$w_{y'}(\mathrm{t}_2)=w_{y'}(\mathrm{t}_1)+\alpha_{y'}(\mathrm{t}_2-\mathrm{t}_1) \tag{29}$$

Step3：更新平动速度：

计算此时沿鼓面向上的合外力为：

$$N=\sum_{i=1}^{n}(\vec{F}_i\,\vec{e}_z) \tag{30}$$

计算此时的平动加速度：

$$a_z=\frac{N}{m}-g \tag{31}$$

因为时间极短，采用差商代替微商，得到该时刻鼓面的平动速度为:

$$v_z(t_2)=v_z(t_1)+a_z(t_2-t_1) \tag{32}$$

Step4：更新中心点位置 K：

$$K_1(t_2)=K_1(t_1)\qquad K_2(t_2)=K_2(t_1)\qquad K_3(t_2)=K_3(t_1)+v_z(t_2-t_1) \tag{33}$$

Step5：更新绳子的受力点位置：

因为同心鼓倾斜了一定角度，所以受力点的位置将发生变化，为求得其新坐标，在已知原坐标的情况下，采用三维状态下的仿射变换进行求解。

Step6：更新时间参量：

$$t_1=t_1+\triangle t\qquad t_2=t_2+\triangle t \tag{34}$$

Step7：判断：

判断时间是否达到上限，若已达到，进行下一步；若未达到，则返回步骤2。

Step8：输出 0.1s 时的斜面倾斜角度：

根据上述模型和算法，带入题目条件，得到 0.1s 时斜面的倾斜角度如下：

表 1. 不同情况下斜面的倾斜角度

序号	1	2	3	4	5	6	7	8	9
斜面倾斜角度/度	0.494	0.838	0.347	1.527	2.816	1.167	3.343	1.420	0.836

5.3 问题三的分析与求解

由第二问可知，队员不能够精确控制发力时机和力度。在此，我们首先基于问题二的模型，采用控制变量法单独分析发力时机和力度对鼓面倾斜角度的影响，然后基于问题一的模型，给出对一问所给策略的调整。

5.3.1 发力大小的影响

为探究发力大小对鼓面倾斜角度的影响，基于二问模型，令 8 个人于 0s 时同时施力，其中编号为 1-7 的人各施力 80N。在此情况下，以第 8 个人施力大

小为自变量，其力大小从 80N 变化至 190N，得到 0.1s 时鼓面的倾斜角度，如图所示。

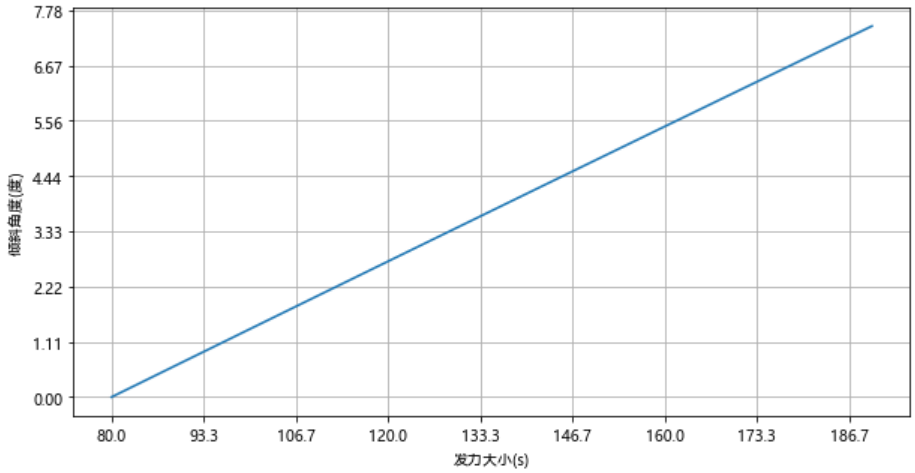

图 7. 发力大小与鼓面倾斜角度的关系

由图可知，鼓面倾斜角度与发力大小呈现出强正相关性，若某队员的发力较大，则易使鼓面倾斜。

5.3.2 发力时机的影响

同理，为探究发力时机对鼓面倾斜角度的影响，令编号为 1-7 的队员于 0s 时同时各施力 80N，编号为 8 的同学施力为 80N，但其施力时机为自变量，范围从 -0.23s 变化 至 0s ，因变量为鼓面倾斜角度，结果如图:

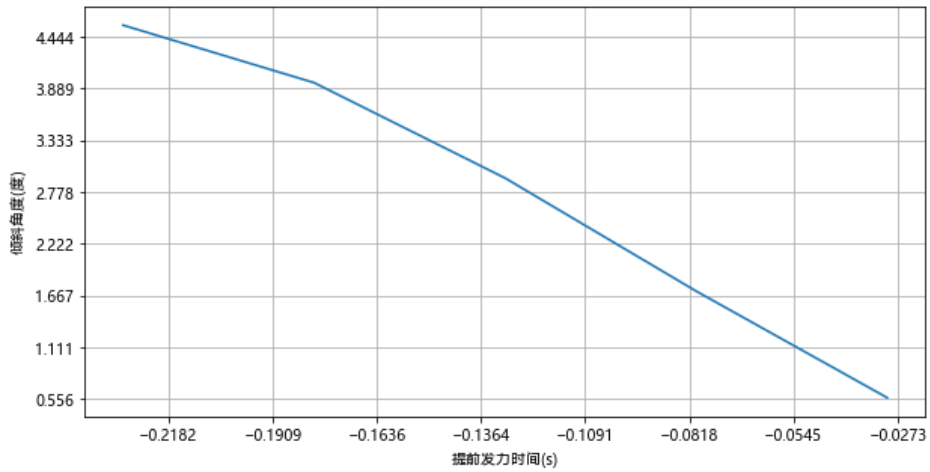

图 8. 发力时机与倾斜角度的关系

由图可知，提前发力时机和鼓面倾斜角度呈现一定的正相关性。

5.3.3 策略的综合调整

实际情况下，往往发力时机和发力大小会同时发生变化。这会使鼓的倾斜角度发生变化，从而影响球与鼓的碰撞，针对这一问题，在下一问中给与解

答。本问中，就如何使鼓面恢复水平进行分析。

根据力矩的合成法则，若 A 队员发力过大，则 A 对面的队员应及时增大力度，且 A 队友旁边的队员及时减小力度，以抵消由队员 A 发力过大而产生的倾斜角度。

同理，若 B 队员提早发力，B 临近的队员发力时机延迟一些，B 对面的队员的发力时机也提前一些。

综合来说，若鼓面已经倾斜，且鼓面最低点指向 A 方向，则 A 方向旁边的队员应减少发力力度，延迟发力时机；A 方向对面的队员增大发力力度，提早发力时机。

5.4 问题四模型的建立与求解

5.4.1 同心鼓与排球第一次碰撞分析

在理想的情况下，每个人都可以精准地控制用力的角度、时机和力度，也就是说，每个成员的动作能够做到完全的协调一致，并且在合理地控制用力角度、时机和力度数值的情况下，能够达到连续颠球次数最多的目的，这也就是我们在第一问中求得的协作策略。

但是在实际的情形下，鼓面并不可能做到始终保持水平的，因为有了鼓面的倾斜，在排球竖直下落时，原本应该与水平的鼓面发生碰撞，但是当鼓面产生了倾斜之后，原本应该竖直反弹的排球，在本题设的情况下实际上相对于竖直方向产生了 1 度的倾斜角度。我们在模型的假设中已经提出，排球与倾斜鼓面相撞后的反射与光的反射定律相同，也即是说排球入射方向与倾斜平面法向量的夹角等于排球的反射方向与法向量的夹角。因此根据光的反射定律，我们容易得出，鼓面倾斜了 0.5 度的角度才可以使得排球与鼓面碰撞反射后，反射方向相对于竖直方向产生了 1 度的倾斜角度。至此，我们分析了球的反射方向与竖直方向产生 1 度的倾斜角度的原因。

我们定义球与鼓面的第一次碰撞的时刻为 0 时刻，第一次碰撞的点为与鼓身相连的绳子达到水平时，中心点所在的位置。我们定义该点为空间直角坐标系的原点，同时我们定义 x 轴的正方向为球碰撞后的倾斜方向在水平面上的投影，同时，我们仍然认为 z 轴的正方向是竖直向上的。至此，我们完成了坐标系的建立工作，与第二问中所建的坐标系相比，新坐标系的变化不大，原点都是位于当绳子水平时的水平面上，z 轴的正方向也都是竖直向上的，区别在于原坐标系的 x 轴的正向是沿着编号为 1 的绳子的方向。二者相比较而言，只是 XoY 坐标平面围绕 z 轴旋转了一定的角度，因此我们在问题二建立的模型在问

题四仍然是适用的。

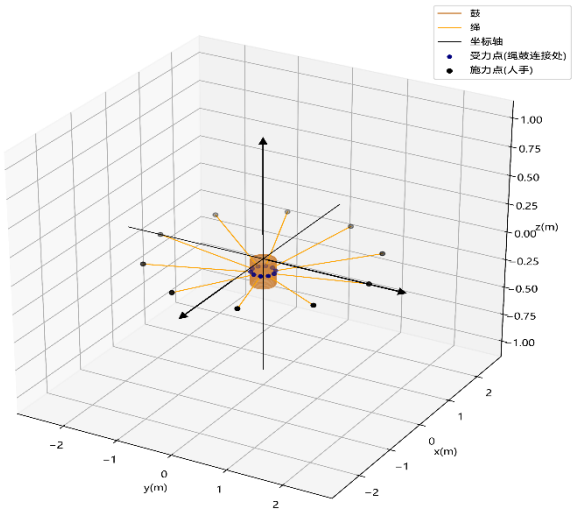

图 9. 以碰撞点为坐标系的坐标轴示意图

在排球与鼓面碰撞之后，分析排球的受力可知，排球在做斜抛运动，排球反射速度的方向与竖直方向成 1 度的角，根据题目的条件可知，排球反弹之后做斜抛运动的最大高度为 60cm.

我们不妨假设排球做斜抛运动的速度为v_{stant}，该速度在竖直方向上的分量为$v_{stant,z}$，水平方向上的分量为$v_{stant,x}$，若在 x，z 下标分量上加上 0，则特指 0 时刻时斜抛速度在 x 轴方向上和 z 轴方向上的分量，定义从反弹开始到排球运行到最高点时所经过的时间为 t，根据斜抛运动的计算公式，我们可以列出以下等式：

$$v_{stant,x_0} = v_{stant,z_0} \cdot \tan 1 \tag{35}$$

$$v_{stant,z_0}{}^2 = 2g \cdot 0.6 \tag{36}$$

$$v_{stant,z_0} = gt \tag{37}$$

可以求得反射后速度在 x 轴方向上的分量为 3.4m/s，水平方向上的速度为 0.05m/s，时间为 0.346s

第一次撞击的示意图如图 10 所示：

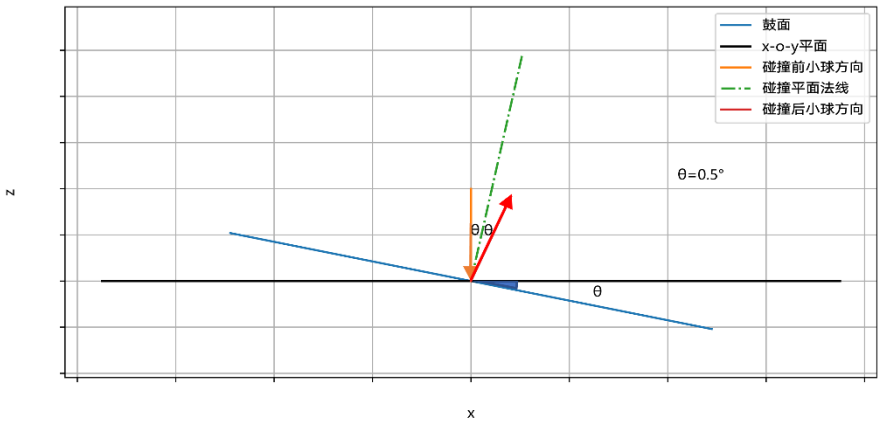

图 10. 第一次碰撞示意图

5.4.2 同心鼓与排球第二次碰撞分析

我们从之前的讨论中已经得知，小球在经历了第一次碰撞之后，所做的运动形式是斜抛运动，同心鼓与排球第二次碰撞的目的，是使小球经过第二次碰撞反射后，速度的方向是竖直向上的。在鼓沿 z 轴正常移动的情况下，我们假设团队的成员们通过控制发力的时机和力度来影响鼓面的倾角。

在本问坐标轴下，排球的斜抛运动的坐标参数只有两个会发生变化，也就是说，我们可以在空间中，寻找出一个经过原点的平面，使得排球的斜抛运动是落在这个平面中的，根据我们所建立的坐标系，斜抛运动的平面即为 XoZ 平面，我们将在这个平面里研究同心鼓与排球发生第二次撞击时的相关运动参数。

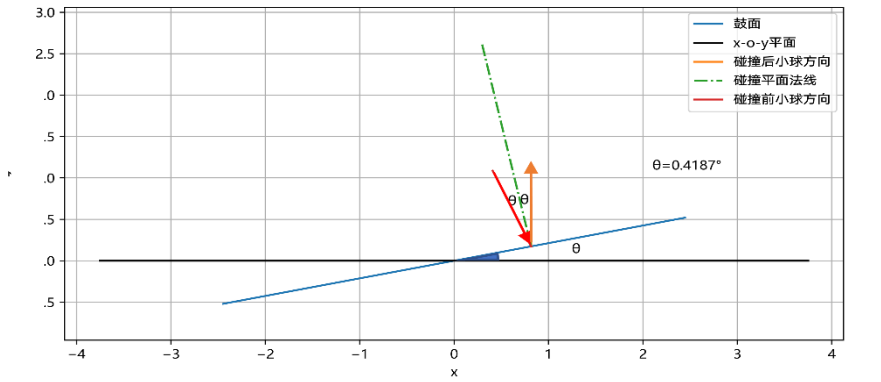

图 11. 第二次碰撞示意图

因此，我们不妨假设，转动后的鼓面与水平面的夹角为 θ 如图（11）中所标示，从第一次碰撞到第二次碰撞所经历的时间为 T，定义该斜抛运动的速度

在第二次撞击的时刻在竖直方向上的分量为$v_{stant,z2}$，水平方向的分量为$v_{stant,x2}$，我们可以得到如下等式：

$$\frac{v_{stant,x2}}{v_{stant,z2}}=\tan 2\theta \tag{38}$$

$$v_{stant,x2}=v_{stant,x0} \tag{39}$$

$$T=\frac{v_{stant,y0}}{g}+\frac{v_{stant,x0}}{g\tan 2\theta} \tag{40}$$

$$\tan\theta=\frac{0.6-\frac{1}{2}gt^2}{T\cdot v_{stant,x0}} \tag{41}$$

将式（38）至式（39）带入式（40）进行化简，发现化简得到的等式较为复杂，求解析解有一定的难度，因此我们利用 *MATLAB* 进行数值求解，最终求解结果$\theta=0.418^{\circ}$，综合第一次碰撞分析和第二次碰撞分析我们可以得到，第一次碰撞后，鼓沿 z 轴上下移动，当鼓达到最低点时，由于重力和拉力的综合作用使得鼓面恢复水平状态，之后，鼓面正常上升，团队成员发力以使得第二次碰撞时间鼓面与水平面形成 0.418° 的夹角。

5.4.3 网格搜索算法的建立

网格搜索算法是通过指定参数进行穷举搜索，通过对估计函数进行交叉验证来得到最优解的求解算法，本模型中，我们采用该算法对扰动值进行求解。

基于第二问的算法，结合网格搜索算法，我们做如下修正：

Step1：根据上述坐标系，重新确定绳子的施力点和受力点的三维坐标

Step2：通过球的上抛高度求得鼓的最大速度 3.4m/s，然后设置参数 h，表示绳子最低点据水平位置的高度，接着通过对鼓以 3.4m/s 速度向下运动情景的仿真，求解出 h 的值。

Step3：设置 10 个队员的初始拉力均为 80N，于 0s 同时施力。第 1，第 5 两个队员施加的拉力各增加 x1 和 x2，共 2 个参数。

Step4：设置扰动参数范围，通过网格搜索算法求得参数结果，使得鼓面在与球第二次碰撞时，鼓面倾斜角度为0.4178°。

Step5：在网格搜索中，算法通过参数 x1,x2 得到鼓面绕 X′ 轴，Y′ 轴旋转的角度 y1,y2。

Step6：已知该坐标系下鼓面应绕 X′，Y′ 轴旋转角度 y3,y4，才能符合题目要求；通过比较损失函数$loss=(y1-y3)^2+(y2-y4)^2$得解。

Step7：对运算结果进行分析，给出最为理想的结果。

5.4.4 模型求解

在网格搜索算法中，我们首先在一定取值范围进行较低精度的仿真，之后，取出最优的几个网格，即图 12 左侧图像中蓝色较深的部分，提高精度等级，再次运行仿真，获得图 12 右侧图像所示的损失函数图，取出最优的几个网格点，得到一组方案。

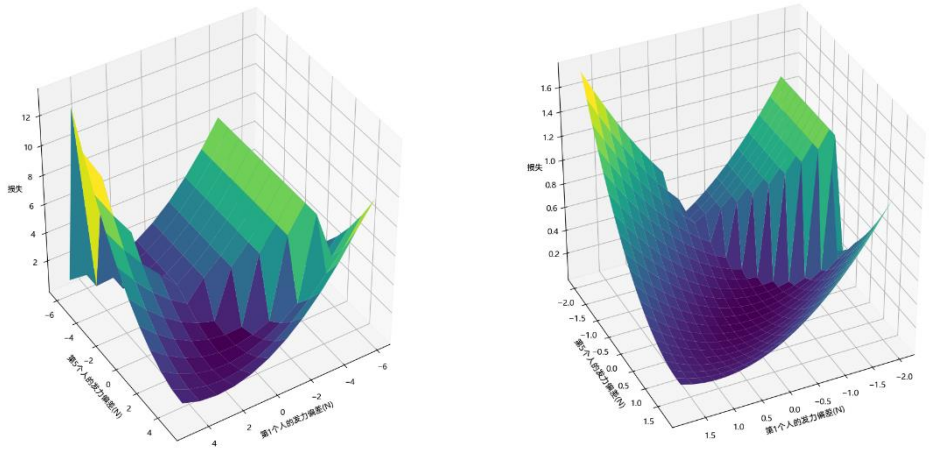

图 12. 损失函数

计算运行后，最优的 10 组方案如下表所示，表中 1 号队员与 x 轴正方向夹角为 12 度，2 号队员于 x 轴正方向夹角为 24 度。为尽量修正小球方向，建议 1 号队员发力 80.4N，5 号队员发力 79.1N，其余队员发力 80N。

表 2. 网格搜索算法最优结果

与 y 轴夹角	与 x 轴夹角	loss	1 号发力力度	5 号发力力度
0.001244	0.393003	0.000602	80.4	79.2
0.015481	0.459999	0.002046	80.4	79
-0.0266	0.455534	0.002154	80.6	79.2
0.043334	0.397462	0.002279	80.2	79
-0.04084	0.388544	0.002506	80.6	79.4
0.057574	0.464464	0.00552	80.2	78.8
-0.06868	0.45107	0.005844	80.8	79.4
0.085431	0.401922	0.007541	80	78.8
-0.08291	0.384087	0.007991	80.8	79.6
0.029095	0.330466	0.008421	80.2	79.2

使用上述搜索到的最优的策略，进行仿真，可以得到它的仿真模拟图。

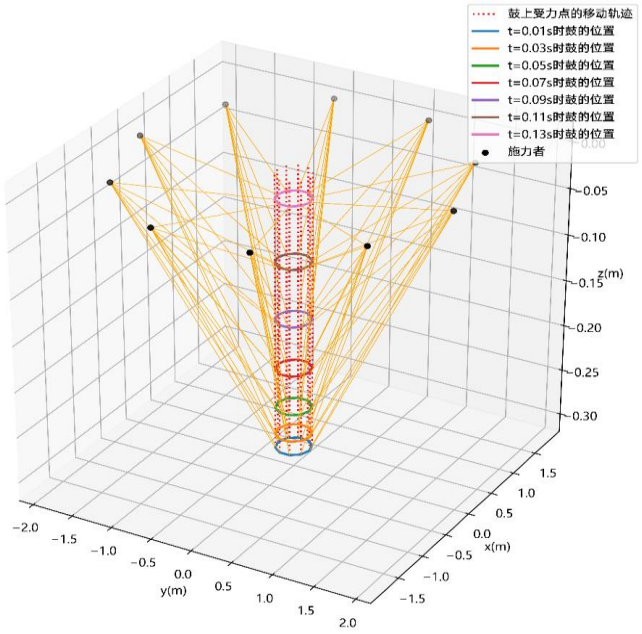

图 13. 最优策略仿真模拟图

在本例中，我们的策略建议 1 号队员增大拉力，5 号队员减小拉力，其他队员不参与角度校正，这在实际情况是难以实现的。在实际情况下，较小的力度控制难以实现，并且由于队员观察视角所限，无法快速精确地确认应该参与角度校正的两名队员，由此，我们对理论情况下的策略进行修正，给出适用于实际情况调整策略为：若发现小球非竖直上抛，所有队员应控制发力，使鼓下降到最低点时由于重力作用而接近水平，在发力使鼓上升时，所有队员都参与角度校正，若发现球的上抛曲线朝向自己，使用较大的力；上抛曲线朝向对面，则应使用较小的力，并且球的上抛曲线倾斜角度越大，则自己的力度调整应该越剧烈。

6.1 模型的优点

1）该模型基于物理实际情况，充分考虑球与同心鼓运动过程中的物理量，通过它们之间的相互关系，得到了一个较为准确的结果

2）该模型采用矢量运算，通过建立坐标系，对坐标进行运算以得到结果，从而避免了对旋转角度的复杂思考与运算，使得模型和算法更为简洁凝练。

3）该模型通过建立在 XoY 平面相对静止，而在 z 轴上平动的垂直轴线 X′ 和 Y′ 轴，基于 X′ 和 Y′ 轴对鼓进行转动分析，从而可以求出鼓面的倾角。

6.2 模型的缺点

1）该模型忽略了队员在发力周期中的拉力变化，实际情况下，队员在发力周期中，其拉力大小不能一直保持恒定值，往往会在该恒定值上波动。

2）该模型忽略了空气阻力，鼓在实际运动中，因受到阻力而损耗能量，队员在施加拉力时需要提供额外拉力以补充这一部分能量。

6.3 模型的改进和推广

1）针对缺点一，在算法设计中，以队员施加的力为基础，增加一个微小的扰动力，使得模型能够更加符合实际情况。

2）针对缺点二，在模型设计中增加空气作用力的影响，使得模型能够推广应用在风阻较大的场合。比如飞机风行时，该模型应用在鸟与机翼的碰撞过程，从而可以尽量避免产生安全事故。

六． 参考文献

[1]庞延理.巧解一维弹性碰撞[J].湖南中学物理,2019,34(06):89-90+96.

[2]刘君,韩芳,夏冰.有限差分法中几何守恒律的机理及算法[J].空气动力学学报,2018,36(06):917-926.

[3]石睿. 基于图像的单视点场景建模[D].中国科学院研究生院（计算技术研究所）,2001.

[4]王好义.反射角等于入射角吗?[J].物理教学,1986(10):28+20.

附录：

MATLAB 2018a

```
g=9.8;
for x=0:0.00001:45
    y=x/180*3.14159265;
    A=((0.05/sin(2*y)).^2-1/400)/(2*g);
    B=0.6-3.4*tan(y)/(20*g)-(0.05*cot(2*y)*tan(y))/(20*g);
    if abs(A-B)<=10^(-4)
        x
    end
end
```

```
########################################################################
# Python3
# -*- coding: utf-8 -*-
# @File：  drum_simulator.py
# @Desc：  同心鼓仿真器-主版本(OI 优化和数据分析版本见附件)
# @Project : MCM-2019

"""
备忘:
 - 时间:      S,以 0 为标准发力时刻
 - 坐标轴：    x 轴方向朝第 1 人,右手系，z=0 位置在绳水平处
 - 力:        F,使用施，受力点控制方向
 - 高度:      M,H 轴以绳子水平位置为 0,向上
 - 角速度:    弧度/s,以右手螺旋法则为正,输出转化为角度制
 - 速度:      m/s,沿 H 轴,向上为正
 - 施力点:    forcer, 人手的位置,三阶 np.array
 - 受力点:    forcee, 鼓连接绳的位置,三阶 np.array

代码结构设计:
 - Situation:  物理环境，控制所有物理变量状态
 - Drum:         鼓，控制鼓本身的状态
 - Force:        力，控制力的状态，
```

```
 - main:            主程序，实例化类并配置以完成仿真

注意:
 - update_forcee :  使用法向量进行 forcee 仿射变换在全周期仿真时存在风险，但在 0.25 周期仿真中会提供比角度逼近更高的精确度
"""

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
from math import *
from tabulate import tabulate

# 全局日志
REPORT = pd.DataFrame(
    index=[
        "角度 X/度",
        "角度 Y/度",
        "高度 H/米",
        "角速度 X/弧度每秒",
        "角速度 Y/弧度每秒",
        "平动速度 H/米每秒",
        "角加速度 X/弧度每二次方秒",
        "角加速度 Y/弧度每二次方秒",
        "平动加速度 H/弧度每二次方秒",
    ]
)


class Situation:
    """
    整个物理情景
    """
```

```
    person_num = 0
    line_length = 0

    drum_radius = 0.2
    drum_start_height = -0.11

    frequency = 100

    def __init__(self, forces_magnitude, forces_time, line_length=None):

        Situation.person_num = len(forces_magnitude)
        if len(forces_magnitude) != len(forces_time):
            print(
                "ERROR:力的输入个数不匹配,力度和开始时间长度分别为：
{}{}".format(
                    len(forces_magnitude), len(forces_time)
                )
            )

        if line_length is not None:
            Situation.line_length = line_length

        self.forces_magnitude = forces_magnitude
        self.forces_time = forces_time

        # 初始化力和鼓
        self.forcer = self.init_forcer()
        self.forcee = self.init_forcee()
        self.drum = Drum(
            self.forcer,
            self.forcee,
            self.forces_magnitude,
            self.forces_time,
```

```
            self.frequency,
            self.drum_start_height,
        )

    @staticmethod
    def init_forcer():
        """
        计算施力点位置
        :return:  施力点位置列表:np.array
        """
        forcer = []
        for i in range(Situation.person_num):
            forcer.append(
                np.array(
                    [
                        (Situation.drum_radius + Situation.line_length)
                        * cos(2 * i * pi / Situation.person_num),
                        (Situation.drum_radius + Situation.line_length)
                        * sin(2 * i * pi / Situation.person_num),
                        0,
                    ]
                )
            )
        return forcer

    @staticmethod
    def init_forcee():
        """
        计算受力点位置
        :return:  受力点位置列表:np.array
        """
        forcee = []
        for i in range(Situation.person_num):
            forcee.append(
```

```
                    np.array(
                        [
                            Situation.drum_radius * cos(2 * i * pi /
Situation.person_num),
                            Situation.drum_radius * sin(2 * i * pi /
Situation.person_num),
                            Situation.drum_start_height,
                        ]
                    )
                )
            return forcee

        def run(self, time_limit=None, height_limit=None):
            global REPORT
            self.drum.run(time_limit, height_limit)
            REPORT.T.to_csv("./brief_report.csv")


class Force:
    """力"""

    forcees = []
    forcers = []

    @classmethod
    def set(cls, forcers, forcees):
        """
        设置力 ee,力 er 的位置，以便力的属性更新
        """
        cls.forcees = forcees
        cls.forcers = forcers

    def __init__(self, id, force, is_started):
        self.force = force
```

```
            self.id = id
            self.forcee = Force.forcees[self.id]
            self.forcer = Force.forcers[self.id]
            self.force_now = 69
            self.update(is_started)

        def update(self, is_started):
            if is_started:
                self.force_now = self.force
            else:
                self.force_now = 69
            self.forcee = Force.forcees[self.id]
            self.forcer = Force.forcers[self.id]


class Drum:
    """鼓"""

    frequency = 1
    console_report = [
        "时间",
        "角度 X",
        "角度 Y",
        "总倾斜角度",
        "高度 H",
        "角速度 X",
        "角速度 Y",
        "平动速度 H",
        "角加速度 X",
        "角加速度 Y",
        "平动加速度 H",
    ]

    def __init__(
```

```
        self, forcer, forcee, forces_magnitude, forces_time, frequency, height
):
        """
        初始化鼓的相关物理量
        :param forcer:  施力点列表
        :param forcee:  受力点列表
        :param forces_magnitude:  力的大小列表
        :param forces_time:  力的开始时间列表
        :param frequency:  采样频率
        :param height:  鼓的初始位置
        """

        #  采样率
        Drum.frequency = frequency

        #  时间
        self.current_time = min(forces_time)

        #  零阶量
        self.forcer = forcer
        self.rota_intertia = 0.0507
        self.weight = 3.6
        self.forces_time = forces_time

        #  负一阶量
        self.angle = {"x": 0, "y": 0}
        self.height = height
        self.forcee = forcee

        #  负二阶量
        self.angel_velocity = {"x": 0, "y": 0}
        self.velocity = 0

        #  力
```

```
            Force.set(self.forcer, self.forcee)
            self.force = [
                Force(i, forces_magnitude[i], forces_time[i] <= self.current_time)
                for i in range(len(forcee))
            ]

    def set_status(self, height=None, angle=None, angel_velocity=None,
velocity=None):
            """
            设置鼓的状态
            :param height:  高度
            :param angle:  角度
            :param angel_velocity:    角速度
            :param velocity:  速度
            :return:
            """
            if height is not None:
                self.height = height
            if angle is not None:
                self.angle = angle
            if angel_velocity is not None:
                self.angel_velocity = angel_velocity
            if velocity is not None:
                self.velocity = velocity

            self.update_forcee()
            Force.set(self.forcer, self.forcee)

    def run(self, time=None, height=None):
            """
            开始仿真
            :param time:停止时间
            :param height:  停止高度
            :return:
```

```
"""

Force.set(self.height, self.angle)

if time is None and height is not None:
        while self.height <= height:
                self.next_moment()

elif time is not None and height is None:
        while self.current_time <= time:
                self.next_moment()
else:
        print("ERROR: Running limit error!")

pd.DataFrame(
        [
                [
                        self.current_time,
                        self.angle["x"],
                        self.angle["y"],
                        self.angel_velocity["x"],
                        self.height,
                        self.angel_velocity["x"],
                        self.angel_velocity["y"],
                        self.velocity,
                        self.angular_acce["x"],
                        self.angular_acce["x"],
                        self.translation_acca,
                ]
        ],
        columns=Drum.console_report,
).to_csv("./res.csv", index=Force)
print(str(self))
```

```
    def next_moment(self):
        """
        更新至下一次采样
        :return:
        """
        # 更新一阶量
        self.angle["x"] += self.angel_velocity["x"] * 0.1 / Drum.frequency
        self.angle["y"] += self.angel_velocity["y"] * 0.1 / Drum.frequency
        self.height += self.velocity * 0.1 / Drum.frequency
        self.update_forcee()

        # 更新二阶量
        self.angel_velocity["x"] += self.angular_acce["x"] * 0.1 / Drum.frequency
        self.angel_velocity["y"] += self.angular_acce["y"] * 0.1 / Drum.frequency
        self.velocity += self.translation_acca * 0.1 / Drum.frequency

        # 更新三阶量
        Force.set(self.forcer, self.forcee)
        for i, f in enumerate(self.force):
            f.update(self.forces_time[i] <= self.current_time)

        # 更新时间
        self.current_time += 0.1 / Drum.frequency

        # 报告
        self.save_report()

    def update_forcee(self):
        """
        根据鼓位置及角度计算受力点
        :return: None
        """

########################################################################
```

```
        # 定义旋转操作符
        rotate_x = lambda r: r[0] * np.array(
            [
                [1, 0, 0, 0],
                [0, cos(r[1]), sin(r[1]), 0],
                [0, -sin(r[1]), cos(r[1]), 0],
                [0, 0, 0, 1],
            ]
        )
        rotate_y = lambda r: r[0] * np.array(
            [
                [1, 0, 0, 0],
                [0, cos(r[1]), sin(r[1]), 0],
                [0, -sin(r[1]), cos(r[1]), 0],
                [0, 0, 0, 1],
            ]
        )
        rotate_z = lambda r: r[0] * np.array(
            [
                [cos(r[1]), sin(r[1]), 0, 0],
                [-sin(r[1]), cos(r[1]), 0, 0],
                [0, 0, 1, 0],
                [0, 0, 0, 1],
            ]
        )

######################################################################
        # 忽略转动
        # for i, e in enumerate(self.forcee):
        #         self.forcee[i] = (e * np.array([1, 1, 0])) + np.array([0, 0, self.height])


######################################################################
        # 使用法向量计算
```

```
        # normal_vector = -np.cross(
        #         np.array([0, 1, tan(self.angle["y"])]),
        #         np.array([1, 0, tan(self.angle["x"])]),
        # )
        # normal_vector = normal_vector / sqrt(sum(normal_vector ** 2))
        #
        # alpha = -(
        #         atan(normal_vector[0] / normal_vector[1])
        #         if round(normal_vector[1], 6) != 0
        #         else 0
        # )
        # beta = (
        #         atan(normal_vector[2] / normal_vector[1])
        #         if round(normal_vector[1], 6) != 0
        #         else 0
        # )
        # s = rotate_x((np.eye(4), alpha))
        # s = rotate_y((s, beta))
        # s = rotate_y((s, -alpha))


########################################################################
        #  直接使用角度计算
        s = rotate_x((np.eye(4), -self.angle["x"]))
        s = rotate_y((s, -self.angle["y"]))


########################################################################
        #  更新所有 forcee
        forcee = []
        for i in range(Situation.person_num):
            forcee.append(
                np.array(
                    [
```

```
                                Situation.drum_radius * cos(2 * i * pi /
Situation.person_num),
                                Situation.drum_radius * sin(2 * i * pi /
Situation.person_num),
                                0,
                            ]
                        )
                    )
            for i, e in enumerate(forcee):
                forcee[i] = (np.append(e, 1) @ s)[:-1] + np.array([0, 0, self.height])
            self.forcee = forcee

    @property
    def angular_acce(self):
        """
        转动加速度计算
        :return: {"x": x 转动加速度, "y": y 转动加速度}
        """
        # X 轴
        x_moment_sum = 0
        for f in self.force:
            f_vector = (
                f.force_now
                * (f.forcer - f.forcee)
                / sqrt(np.sum((f.forcer - f.forcee) ** 2))
            )
            f_arm = f.forcee - np.array([0, 0, self.height])
            f_moment = np.cross(f_vector, f_arm)[0]
            x_moment_sum += f_moment

        # Y 轴
        y_moment_sum = 0
        for f in self.force:
            f_vector = (
```

```
                f.force_now
                * (f.forcer - f.forcee)
                / sqrt(np.sum((f.forcer - f.forcee) ** 2))
            )
            f_arm = f.forcee - np.array([0, 0, self.height])
            f_moment = np.cross(f_vector, f_arm)[1]
            y_moment_sum += f_moment

        return {
            "x": x_moment_sum / self.rota_intertia,
            "y": y_moment_sum / self.rota_intertia,
        }

@property
def translation_acca(self):
        """
        平动加速度计算
        :return:  平动加速度
        """
        h_moment_sum = 0
        for f in self.force:
            f_vector = (
                f.force_now
                * (f.forcer - f.forcee)
                / sqrt(np.sum((f.forcer - f.forcee) ** 2))
            )
            f_moment = f_vector[2]
            h_moment_sum += f_moment
        return h_moment_sum / self.weight

def save_report(self):
        """
        收集鼓状态,输出报告
        :return: None
```

```
        """
        global REPORT
        REPORT[self.current_time] = [
            round(self.angle["x"] / pi * 180, 6),
            round(self.angle["y"] / pi * 180, 6),
            round(self.height, 6),
            round(self.angel_velocity["x"], 6),
            round(self.angel_velocity["y"], 6),
            round(self.velocity, 6),
            round(self.angular_acce["x"], 6),
            round(self.angular_acce["y"], 6),
            round(self.translation_acca, 6),
        ]

    @property
    def angle_all(self):
        """
        组合两个方向上的角度
        :return:  总的偏转角
        """
        normal_vector = np.cross(
            np.array([0, 1, tan(-self.angle["y"])]),
            np.array([1, 0, tan(self.angle["x"])]),
        )
        return acos(abs(normal_vector[2] / sqrt(sum(normal_vector ** 2))))

    def __str__(self):
        """
        美化输出
        :return: Drum.report 里要求的所有项目
        """
        return tabulate(
            pd.DataFrame(
                [
```

```
                        [
                            round(self.current_time, 5),
                            round(self.angle["x"] / pi * 180, 6),
                            round(self.angle["y"] / pi * 180, 6),
                            # 输出为红色，熬夜的时候更容易看清楚主要结果
                            "\033[0;31m%s\033[0m"
                            % str(round(self.angle_all / pi * 180, 6)),
                            round(self.height, 6),
                            round(self.angel_velocity["x"], 6),
                            round(self.angel_velocity["y"], 6),
                            round(self.velocity, 6),
                            round(self.angular_acce["x"], 6),
                            round(self.angular_acce["y"], 6),
                            round(self.translation_acca, 6),
                        ]
                    ],
                    columns=Drum.console_report,
                ),
                tablefmt="pipe",
                headers="keys",
            )


if __name__ == "__main__":

################################################################################
################################################
    # 常规调用仿真
    # 初始化情景
    s = Situation(
        forces_magnitude=np.array([80, 80, 80, 90, 80, 80, 80, 80]),
        forces_time=np.array([0, 0, 0, 0, -0.1, 0, 0, 0]),
        line_length=1.7,
    )
```

```
    # 设置鼓初始状态
    s.drum.set_status(height=-0.11)

    # 开始仿真
    s.run(time_limit=0.1)


################################################################################
################################################
    # 发力大小灵敏度分析-数据采集
    # with open("./force_angle.csv", "w") as f:
    #     f.write("force,angle_y\n")
    #
    # for force in range(80, 200, 10):
    #     # 初始化情景
    #     s = Situation(
    #         forces_magnitude=np.array([force, 80, 80, 80, 80, 80, 80, 80]),
    #         forces_time=np.array([0, 0, 0, 0, 0, 0, 0, 0]),
    #         line_length=1.7,
    #     )
    #
    #     # 设置鼓初始状态
    #     s.drum.set_status(height=-0.11)
    #
    #     # 开始仿真
    #     valuable_data = s.search(time_limit=0.1)[1]
    #     with open("./force_angle.csv", "a") as f:
    #         f.write("{},{}\n".format(force, valuable_data))


################################################################################
################################################
    # 发力时间灵敏度分析-数据采集
```

```
# with open("./time_angle.csv", "w") as f:
#         f.write("time,angle_y\n")
#
# time = -0.23
# while time < 0:
#         # 初始化情景
#         s = Situation(
#                 forces_magnitude=np.array([80, 80, 80, 80, 80, 80, 80, 80]),
#                 forces_time=np.array([time, 0, 0, 0, 0, 0, 0, 0]),
#                 line_length=1.7,
#         )
#
#         # 设置鼓初始状态
#         s.drum.set_status(height=-0.11)
#
#         # 开始仿真
#         valuable_data = s.search(time_limit=0.1)[1]
#         with open("./time_angle.csv", "a") as f:
#                 f.write("{},{}\n".format(time, valuable_data))
#
#         time += 0.05


################################################################################
################################################
# 施力,受力分析-数据采集
# s = Situation(
#         forces_magnitude=np.array([79.6, 80, 80, 80, 80, 80, 80, 80, 80, 79.9]),
#         forces_time=np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0]),
#         line_length=1.7,
# )
#
# s.drum.set_status(height=-0.3)
# s.run(height_limit=0)
```

```
#
# er = ""
# for fr in s.forcer:
#         er += "{},{},{}\n".format(fr[0], fr[1], fr[2])
# with open("./er.csv", "w") as f:
#         f.write(er)


##########################################################################
################################################
#  网格搜索-数据采集
# res = ""
# i = 0
# with open("./log.txt", "w") as f:
#         f.write("x1,x10,y1,y2,loss\n")
# # a,b,x,y
# for x in range(-20, 20):
#         for y in range(-20, 20):
#                 s = Situation(
#                         forces_magnitude=[
#                                 80 - x / 10,
#                                 80,
#                                 80,
#                                 80,
#                                 80,
#                                 80,
#                                 80,
#                                 80,
#                                 80,
#                                 80 - y / 10,
#                         ],
#                         forces_time=[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
#                         line_length=2,
#                 )
```

```
#             i += 1
#             s_x, s_y = s.search(height_limit=0)
#             if i % 20 == 0:
#                 with open("./log.txt", "a") as f:
#                     f.write(res)
#                 res = ""
#                 print(
#                     i,
#                     x,
#                     y,
#                     round(s_x, 2),
#                     round(s_y, 2),
#                     round((s_x ** 2 + (s_y - 0.4175) ** 2), 6),
#                 )
#             res += "{},{},{},{},{}\n".format(
#                 x, y, s_x, s_y, s_x ** 2 + (s_y - 0.4175) ** 2
#             )
# with open("./log.txt", "a") as f:
#     f.write(res)

##############################################################################
# python3
# -*- coding: utf-8 -*-
# @File：  plot.ipynb
# @Desc：  绘图工具
# @Project : MCM-2019

# Q1 v-t 图

plt.figure(figsize=(10, 5))
plt.grid()
x = np.linspace(0, 6 * np.pi, 300)
y = np.sin(x)
d = np.pi / 2
```

```
plt.yticks(np.linspace(-1, 1, 6))
plt.xticks(np.linspace(0, 15 * d, 16))

x1 = [0, d, d, 5 * d, 5 * d, 9 * d, 9 * d]
y1 = [0, -0.5, 1, -1, 1, -1, 1]

x2 = [d, d, 5 * d, 5 * d, 9 * d, 9 * d]
y2 = [-0.5, 1, -1, 1, -1, 1]

plt.fill_between([0, d], [0, -0.5], label="球的初始下落距离")
plt.fill_between([3 * d, 5 * d], [0, -1], label="球的半周期下落距离")
plt.fill_between(
    np.linspace(0, 0.5 * np.pi, 100),
    np.sin(np.linspace(0, 0.5 * np.pi, 100)),
    label="鼓的初始下落距离",
)

plt.plot(x1, y1, color="black", linestyle="-.", label="球的速度")
plt.plot(x, y, linestyle="--", label="鼓的速度")

plt.scatter(x2, y2, color="r", s=50, label="碰撞点")
plt.legend(loc="upper right")

plt.savefig("../res/pic/q1-v-t.png", dpi=720)

# Q3  分析力的大小和发力时间对角度的关系图

f_a = pd.read_csv("../res/csv/force_angle.csv")
plt.figure(figsize=(10, 5))
plt.grid()

plt.yticks(np.linspace(0, 10, 10))
plt.xticks(np.linspace(80, 200, 10))
```

```
plt.xlabel("发力大小(s)")
plt.ylabel("倾斜角度(度)")

plt.plot(f_a["force"], f_a["angle_y"])

plt.savefig("../res/pic/q3-f-a.png", dpi=720)

f_a = pd.read_csv("../res/csv/time_angle.csv")
plt.figure(figsize=(10, 5))
plt.grid()

plt.yticks(np.linspace(0, 5, 10))
plt.xticks(np.linspace(-0.3, 0, 12))

plt.xlabel("提前发力时间(s)")
plt.ylabel("倾斜角度(度)")

plt.plot(f_a["time"], f_a["angle_y"])

plt.savefig("../res/pic/q3-t-a.png", dpi=720)

# Q4  小球运动角度

plt.figure(figsize=(10, 5))
plt.grid()

theta = 30 * 0.4 * pi / 180
l = 2.5
plt.axis("equal")
plt.xlabel("x")
plt.ylabel("z")

plt.plot(
```

```
    [cos(theta) * l, -cos(theta) * l], [-sin(theta) * l, sin(theta) * l], label="鼓面"
)

plt.plot([-l * 1.5, l * 1.5], [0, 0], color="black", label="x-o-y 平面")
plt.plot([0, 0], [0, 1], label="碰撞前小球方向")

plt.plot([0, l * sin(theta)], [0, l * cos(theta)], "-.", label="碰撞平面法线")

plt.plot([0, 1 * sin(2 * theta)], [0, 1 * cos(2 * theta)], label="碰撞后小球方向")
plt.legend(loc="upper right")
plt.annotate(s="θ", xy=(l / 2 * cos(theta / 2 + 0.03), -l / 2 * sin(theta / 2 + 0.03)))
plt.annotate(s="θ", xy=(l / 2 * sin(theta / 2), 1 / 2 * cos(theta / 2)))
plt.annotate(s="θ", xy=(l / 2 * sin(theta / 8 - 0.03), 1 / 2 * cos(theta / 8 - 0.03)))
plt.annotate(s="θ=0.5°", xy=(2.1, 1.1))
plt.savefig("../res/pic/q4-1.png", dpi=720)

plt.figure(figsize=(10, 5))
plt.grid()

theta = -30 * 0.4 * pi / 180
l = 2.5
plt.axis("equal")
plt.xlabel("x")
plt.ylabel("z")

plt.plot(
    [cos(theta) * l, -cos(theta) * l], [-sin(theta) * l, sin(theta) * l], label="鼓面"
)

plt.plot([-l * 1.5, l * 1.5], [0, 0], color="black", label="x-o-y 平面")

x, y = (l / 3 * cos(theta), -l / 3 * sin(theta))
plt.plot([0 + x, 0 + x], [0 + y, 1 + y], label="碰撞后小球方向")
plt.plot([0 + x, l * sin(theta) + x], [0 + y, l * cos(theta) + y], "-.", label="碰撞平面法
```

```
线")
plt.plot(
    [0 + x, 1 * sin(2 * theta) + x], [0 + y, 1 * cos(2 * theta) + y], label="碰撞前小球
方向"
)
plt.legend(loc="upper right")

plt.annotate(s="θ", xy=(l / 2 * cos(theta / 2 + 0.03), -l / 2 * sin(theta / 2 + 0.03)))
plt.annotate(
    s="θ", xy=(l / 2 * sin(theta / 2) + x + 0.015, 1 / 2 * cos(theta / 2) + y + 0.015)
)

plt.annotate(s="θ", xy=(l / 2 * sin(theta / 4) - 0.18 + x, 1 / 2 * cos(theta / 4) + y))
plt.annotate(s="θ=0.4187°", xy=(2.1, 1.1))
plt.savefig("../res/pic/q4-2.png", dpi=720)

# Q2  受力分析图

plt.figure(figsize=(12, 2))
plt.grid()
plt.xlabel("x(m)")
plt.ylabel("z(m)")

plt.plot([0.2, 0.2, -0.2, -0.2, 0.2], [-0.33, -0.11, -0.11, -0.33, -0.33], label="鼓")
l = sqrt(1.7 ** 2 - 0.11 ** 2) + 0.2

plt.xticks(np.linspace(-2, 2, 11))
plt.yticks(np.linspace(-0.6, 0, 4))
plt.axis("equal")
plt.scatter([l, -l], [0, 0], color="black", label="施力点")
plt.scatter([-0.2, 0.2], [-0.21, -0.21], color="b", label="受力点")

plt.plot([l, 0.2], [0, -0.21], "-.", color="orange", label="绳")
plt.plot([-l, -0.2], [0, -0.21], "-.", color="orange")
```

```
plt.plot([0, 0], [-0.21, -0.51], color="b", label="重力")
plt.plot([-0.2, -1.5], [-0.21, -0.05], color="black", label="拉力")
plt.plot([0.2, 1.5], [-0.21, -0.05], color="black")
plt.legend(loc="center left", bbox_to_anchor=(1, 0.5))

plt.savefig("../res/pic/q2-f.png", dpi=720)

# Q2  仿真结果图



er8 = pd.read_csv("../res/csv/er_8.csv", header=None)
ee8 = pd.read_csv("../res/csv/ee_8.csv", header=None)

fig = plt.figure(figsize=(8, 8))
ax = Axes3D(fig)
person_num = 8
for i in range(1, person_num):
    ax.plot(ee8[3 * i][10:], ee8[3 * i + 1][10:], ee8[3 * i + 2][10:], ":", color="red")
ax.plot(ee8[0][10:], ee8[1][10:], ee8[2][10:], ":", color="red", label="鼓上受力点的
移动轨迹")

for j in range(10, len(ee8[1]), 20):
    x = [ee8[3 * i][j] for i in range(person_num)] + [ee8[0][j]]
    y = [ee8[3 * i + 1][j] for i in range(person_num)] + [ee8[1][j]]
    z = [ee8[3 * i + 2][j] for i in range(person_num)] + [ee8[2][j]]
    ax.plot(
        x, y, z, label="t={}s 时鼓的位置".format(round(0.1 * j / 2 / 100, 3)),
linewidth=2
    )

for i in range(person_num):
    for j in range(10, len(ee8[1]), 20):
        x = [ee8[3 * i][j], er8[0][i]]
```

```
            y = [ee8[3 * i + 1][j], er8[1][i]]
            z = [ee8[3 * i + 2][j], er8[2][i]]
            ax.plot(x, y, z, color="orange", linewidth="0.5")

ax.set_xlabel("y(m)")
ax.set_ylabel("x(m)")
ax.set_zlabel("z(m)")

ax.scatter(er8[0], er8[1], er8[2], color="black", label="施力者")
plt.legend(loc="upper right")

plt.savefig("../res/pic/q2-res.png", dpi=720)

# Q4 仿真结果图


er = pd.read_csv("../res/csv/er_10.csv", header=None)
ee = pd.read_csv("../res/csv/ee_10.csv", header=None)

fig = plt.figure(figsize=(8, 8))
ax = Axes3D(fig)
person_num = 10
for i in range(1, person_num):
    ax.plot(ee[3 * i][10:], ee[3 * i + 1][10:], ee[3 * i + 2][10:], ":", color="red")
ax.plot(ee[0][10:], ee[1][10:], ee[2][10:], ":", color="red", label="鼓上受力点的移动
轨迹")

for j in range(10, len(ee[1]), 20):
    x = [ee[3 * i][j] for i in range(person_num)] + [ee[0][j]]
    y = [ee[3 * i + 1][j] for i in range(person_num)] + [ee[1][j]]
    z = [ee[3 * i + 2][j] for i in range(person_num)] + [ee[2][j]]
    ax.plot(
        x,
        y,
```

```
            z,
            label="t={}s 时鼓的位置".format(round(j / len(ee[1]) * 0.139, 3)),
            linewidth=2,
        )

for i in range(person_num):
        for j in range(10, len(ee[1]), 20):
            x = [ee[3 * i][j], er[0][i]]
            y = [ee[3 * i + 1][j], er[1][i]]
            z = [ee[3 * i + 2][j], er[2][i]]
            ax.plot(x, y, z, color="orange", linewidth="0.5")

ax.set_xlabel("y(m)")
ax.set_ylabel("x(m)")
ax.set_zlabel("z(m)")

ax.scatter(er[0], er[1], er[2], color="black", label="施力者")
plt.legend(loc="upper right")

plt.savefig("../res/pic/q4-res.png", dpi=720)

# Q2  情景展示及坐标轴图


fig = plt.figure(figsize=(8, 8))
ax = Axes3D(fig)

# drum
r = 0.2
theta = np.arange(0, 2 * np.pi, 0.1)
x = 0 + r * np.cos(theta)
y = 0 + r * np.sin(theta)
x = np.append(x, x[0])
y = np.append(y, y[0])
```

```
for i in range(-22, 0):
    ax.plot(x, y, i / 100, "peru", alpha=0.8)
ax.plot(x, y, i / 100, "peru", alpha=0.8, label="鼓")

angel = np.arange(0, 2 * np.pi, 2 * pi / 8)
x = 0 + r * np.cos(angel)
y = 0 + r * np.sin(angel)

ax.scatter(x, y, -0.11, color="navy", linewidth=0.1, label="受力点(绳鼓连接处)")

# 人和绳
ax.scatter(er8[0], er8[1], er8[2], color="black", label="施力点(人手)")
for i in range(8):
    ax.plot(
        [er8[0][i], x[i]], [er8[1][i], y[i]], [er8[2][i], -0.11], "orange", linewidth=1
    )
ax.plot(
    [er8[0][i], x[i]],
    [er8[1][i], y[i]],
    [er8[2][i], -0.11],
    "orange",
    linewidth=1,
    label="绳",
)

# 坐标轴
ax.plot([-2.5, 2.5], [0, 0], [0, 0], "black", linewidth=0.8)
ax.plot([0, 0], [-2.5, 2.5], [0, 0], "black", linewidth=0.8)
ax.plot([0, 0], [0, 0], [-1, 1], "black", linewidth=0.8, label="坐标轴")

ax.set_xlabel("y(m)")
ax.set_ylabel("x(m)")
ax.set_zlabel("z(m)")
```

```
plt.legend(loc="upper right")

plt.savefig("../res/pic/q1-situation.png", dpi=720)

# Q4  情景展示图


fig = plt.figure(figsize=(8, 8))
ax = Axes3D(fig)

# drum
r = 0.2
theta = np.arange(0, 2 * np.pi, 0.1)
x = 0 + r * np.cos(theta)
y = 0 + r * np.sin(theta)
x = np.append(x, x[0])
y = np.append(y, y[0])

for i in range(-22, 0):
    ax.plot(x, y, i / 100, "peru", alpha=0.8)
ax.plot(x, y, i / 100, "peru", alpha=0.8, label="鼓")

angel = np.arange(0, 2 * np.pi, 2 * pi / 10)
x = 0 + r * np.cos(angel)
y = 0 + r * np.sin(angel)

ax.scatter(x, y, -0.11, color="navy", linewidth=0.1, label="受力点(绳鼓连接处)")

#  人和绳
ax.scatter(er[0], er[1], er[2], color="black", label="施力点(人手)")
for i in range(10):
    ax.plot(
        [er[0][i], x[i]], [er[1][i], y[i]], [er[2][i], -0.11], "orange", linewidth=1
```

```
        )
ax.plot(
        [er[0][i], x[i]],
        [er[1][i], y[i]],
        [er[2][i], -0.11],
        "orange",
        linewidth=1,
        label="绳",
)

# 坐标轴
ax.plot([-2.5, 2.5], [0, 0], [0, 0], "black", linewidth=0.8)
ax.plot([0, 0], [-2.5, 2.5], [0, 0], "black", linewidth=0.8)
ax.plot([0, 0], [0, 0], [-1, 1], "black", linewidth=0.8, label="坐标轴")

ax.set_xlabel("y(m)")
ax.set_ylabel("x(m)")
ax.set_zlabel("z(m)")

plt.legend(loc="upper right")

plt.savefig("../res/pic/q4-situation.png", dpi=720)

# Q2  力矩分析图



fig = plt.figure(figsize=(8, 8))
ax = Axes3D(fig)

# drum
r = 0.2
theta = np.arange(0, 2 * np.pi, 0.1)
x = 0 + r * np.cos(theta)
y = 0 + r * np.sin(theta)
```

```
x = np.append(x, x[0])
y = np.append(y, y[0])

for i in range(-11, 11):
    ax.plot(x, y, i / 100, "peru", alpha=0.8)
ax.plot(x, y, i / 100, "peru", alpha=0.8, label="鼓")

l = -0.2 / sqrt(2)
ax.scatter([-l], [l], 0, label="受力点位置")
ax.plot([0, -l], [0, l], 0, color="red", label="力臂")
ax.plot([-l, -l], [l, l], [0, 0.6], label="拉力在 z 轴方向上的分量")
ax.plot([0, 3 * l], [0, 3 * l], 0, color="navy", label="力矩")

ax.plot([0, 0], [0, 3 * l], 0, "-.", color="navy", label="力矩在轴方向上的分量")
ax.plot([0, 3 * l], [0, 0], 0, "-.", color="navy")

ax.plot([3 * l, 3 * l], [0, 3 * l], 0, "-.", color="black", linewidth=0.8)
ax.plot([0, 3 * l], [3 * l, 3 * l], 0, "-.", color="black", linewidth=0.8)

#  坐标轴
ax.plot([-1, 1], [0, 0], [0, 0], "black", linewidth=0.5)
ax.plot([0, 0], [-1, 1], [0, 0], "black", linewidth=0.5)
ax.plot([0, 0], [0, 0], [-1, 1], "black", linewidth=0.5, label="坐标轴")

ax.set_xlabel("y(m)")
ax.set_ylabel("x(m)")
ax.set_zlabel("z(m)")

plt.legend(loc="upper right")

# plt.savefig("../res/pic/q2-f-r.png", dpi=720)


#  平动坐标轴显示图
```

```
fig = plt.figure(figsize=(8, 8))
ax = Axes3D(fig)

# drum
r = 0.2
theta = np.arange(0, 2 * np.pi, 0.1)
x = 0 + r * np.cos(theta)
y = 0 + r * np.sin(theta)
x = np.append(x, x[0])
y = np.append(y, y[0])

for i in range(-22, 0):
    ax.plot(x, y, i / 100, "peru", alpha=0.8)
ax.plot(x, y, i / 100, "peru", alpha=0.8, label="鼓")

angel = np.arange(0, 2 * np.pi, 2 * pi / 8)
x = 0 + r * np.cos(angel)
y = 0 + r * np.sin(angel)

ax.scatter(x, y, -0.11, color="navy", linewidth=0.1, label="受力点(绳鼓连接处)")

# 人和绳
ax.scatter(er8[0], er8[1], er8[2], color="black", label="施力点(人手)")
for i in range(8):
    ax.plot(
        [er8[0][i], x[i]], [er8[1][i], y[i]], [er8[2][i], -0.11], "orange", linewidth=1
    )
ax.plot(
    [er8[0][i], x[i]],
    [er8[1][i], y[i]],
    [er8[2][i], -0.11],
    "orange",
    linewidth=1,
```

```
        label="绳",
)

# 坐标轴
ax.plot([-2.5, 2.5], [0, 0], [0, 0], "black", linewidth=0.8)
ax.plot([0, 0], [-2.5, 2.5], [0, 0], "black", linewidth=0.8)
ax.plot([0, 0], [0, 0], [-1, 1], "black", linewidth=0.8, label="静止坐标轴")

ax.plot([0, 2.5], [0, 0], [-0.11, -0.11], "navy", linewidth=0.8)
ax.plot([0, 0], [0, -2.5], [-0.11, -0.11], "navy", linewidth=0.8, label="平动坐标轴")

ax.set_xlabel("y(m)")
ax.set_ylabel("x(m)")
ax.set_zlabel("z(m)")

plt.legend(loc="upper right")

plt.savefig("../res/pic/q1-situation1.png", dpi=720)

##########################################################################
##########
# python3
# -*- coding: utf-8 -*-
# @File：  grid1.ipynb & grid2.ipynb
# @Desc：  网格搜索结果分析工具
# @Project : MCM-2019
log = pd.read_csv("./grid1.txt")
a = log.drop_duplicates().sort_values(by="loss").reset_index()
delta = 1

x = np.arange(-6.0, 6.0, delta)
y = np.arange(-6.0, 6.0, delta)
l = len(x)
Z = np.zeros(shape=(l, l))
```

```
X, Y = np.meshgrid(x, y)
for i in range(l):
    Z[i] = a[a["x10"] == -6 + i].sort_values(by="x1")["loss"]

fig = plt.figure(figsize=(8, 8))
ax = Axes3D(fig)
ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=plt.get_cmap("rainbow"))

log = pd.read_csv("./grid2.txt")
a = log.drop_duplicates().sort_values(by="loss").reset_index()

delta = 0.1

x = np.arange(-2.0, 2.0, delta)
y = np.arange(-2.0, 2.0, delta)
l = len(x)
Z = np.zeros(shape=(l, l))
X, Y = np.meshgrid(x, y)

for i in range(l):
    Z[i] = a[a["x10"] == -20 + i].sort_values(by="x1")["loss"]

fig = plt.figure(figsize=(8, 8))
ax = Axes3D(fig)
ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=plt.get_cmap("rainbow"))
```