“同心协力”策略研究

摘　要

如何通过调整队员发力方向、大小和时机能使连续颠球次数尽可能多是颠球活动中需要解决的主要问题。本文建立刚体绕定轴转动模型，在理想状态下和实际情况下模拟，对比后进行调整，得到最优的人员分布、发力大小和时机。

关于问题一 在理想状态下，忽略能量损失，将球与鼓的碰撞看作是完全弹性碰撞，即前后颠球高度始终保持一致，鉴于人数、发力大小和绳与鼓面夹角不确定，可以建立定积分方程求解出这三个变量的关系，在一定的限制条件下得到最优解。

关于问题二 现实情况下，每个人的用力时机，力度和方向不能保证相同，因此就会出现鼓面倾斜现象。通过对题目中已知条件分析，就可以得出初始状态下绳子与水平面的夹角和每个人需提供的力度。然后建立起牛皮鼓定轴转动的模型，利用迭代法，求出每一个时刻的偏转角度，逐次迭代得出最后所求的偏转角 θ 。

关于问题三 在问题二的前提下修改问题一中的最佳策略，其中问题一为理想情况，问题二则为现实情况。问题一中策略在现实情况下，力度和时机的偏差不可避免，那么我们只能从原系统入手，尽可能减小时机和力度不合适所形成的偏斜角。因此，这个误差属于系统误差，若要减小此误差，我们可以从绳长，人数，鼓的初始位置三个方面进行优化，来达到缩小夹角的目的。在问题一策略前提下，假设人数和绳长是不可更改的，是一个定值，那么我们只能从鼓的初始位置建立非线性规划的模型。

关于问题四 与问题二相比，本问已知鼓面倾斜角度，根据问题二建立的模型，给出随机的发力不同人员和发力大小，在 Python 程序中模拟最接近倾斜角度时的各个队员发力大小的最优解。

关键字：刚体绕定轴转动模型、系统误差、偏转角、迭代方法、非线性规划

1. 问题重述

1.1 问题背景

“同心鼓”是一项团队协作能力拓展项目,可培养队员们同心协作的能力。该项目规则是团队成员每人牵拉一根绳子，使鼓面保持水平。项目开始时，球从鼓面中心上方竖直落下，队员一起将球颠起，使其在鼓面上跳动。在题目中我们对队员的发力时机、力度和方向进行模拟和计算，降低队员们颠球需要消耗的体力，可以使颠球次数更多。这类问题研究的是一个物体经过自由落体后，对落地时和落地之后弹起的力学和能量分析，通过研究这类问题，我们可以控制部分变量来实现弹跳次数最多或最少，以及阻碍自由落体状态的物体下落的能量最少等目标，这对于高空坠人救援，航空航天中的飞船降落，军事武器打击等研究提供了简化的思路，因此研究此类问题是极具有现实意义的。

1.2 问题描述

团队在进行“同心鼓”活动时，颠球过程中队员只能抓握绳子的末端，不能接触鼓或绳子的其他位置。活动人数不得少于 8 人，队员之间的距离不得小于 60 cm。活动开始时，球从鼓面中心上方 40 cm 处落下，球被颠起的高度应高于鼓面 40 cm 以上，如果低于 40cm，则项目失败。项目的目标是使得连续颠球的次数尽可能多。

根据上述条件，要研究以下四个问题：

- 在理想状态下，给定每个人的用力方向、时机和力度，建立数学模型得到最省力的方法，使颠球数目最多，并求出此时的颠球高度 ；

- 现实状态下，用力方向，时机和力度不可能做到完全一致，在已知站在某一位置的人出现时机和力度误差的情况下，求解角度与用力方向和时机之间的关系，并计算出在特定情况下的倾斜角度；

- 在现实状态下，每个人都有可能出现时机或者力度的偏差，而导致一定的倾斜角，那么在给定人数，绳长的条件下，如何通过调整鼓所处的初始高度，

夹角使**系统误差**最小化，达到即使在不清楚哪个位置会出现问题，以及几个人出现问题的情况下，对整个游戏的影响最小的目的

- 在已知队员用力方向，时机，力度出现问题而导致的影响的前提下，我们需要根据二三问的模型推出究竟哪个位置的队员出现问题及出现的问题具体是什么，通过此过程，我们可以得到类似于问题二中表格所示内容。然后我们针对已经发生的问题去调整下一时刻的发力方向，时机和力度，使其恢复为理想状态下竖直弹跳，达到快速调整，颠球次数尽可能多的目的

2. 模型假设

a. 不考虑空气阻力带来的能量损失；
b. 球与鼓面碰撞时的碰撞过程视为弹性碰撞；
c. 球与鼓面的碰撞时间极短，因此可忽略球和鼓面所产生的形变以及因为形变所产生的的作用力，即视为理想鼓和理想球
d. 理想状态下，设人的体型、高度、手拉绳的运动方式保持一致，并假设每个人的身高体重恒定，用力相等；
e. 模型将绳子视为理想状态的绳子，牵拉绳子而使绳子产生的形变对整个系统的影响可以忽略不计；

3. 题设限定

（1） 队员人数不少于 8 人；
（2） 队员之间的最小距离不得小于 60 cm；
（3） 项目开始时，球从鼓面中心上方 40 cm 处竖直下落做自由落体运动；
（4） 球被颠起的高度应离开鼓面 40 cm 以上；
（5） 如果球被颠起的高度低于 40cm，则项目停止；
（6） 初始状态下鼓面保持水平；
（7） 鼓身中间固定多根绳子且绳子在鼓深周围定点沿圆周呈均匀分布；

（8） 每根绳子的长度相同.。

4. 符号说明

“表 1”列出了本文需要的符号，文中出现的其它符号将在出现时进行解释。

符号	符号描述	单位
M	鼓的质量	Kg
m	球的质量	Kg
m_1	鼓皮质量	Kg
m_2	鼓身质量	Kg
v_1	鼓碰撞前的速度	m/s
v_2	球碰撞前的速度	m/s
$v_1^{'}$	鼓碰撞后的速度	m/s
$v_2^{'}$	球碰撞后的速度	m/s
g	重力加速度	$m/s\wedge 2$
t	球从下落开始至碰撞的时间	s
h_1	球与鼓面碰撞前下落的距离	m
h_2	鼓面上升的距离	m
h_3	球的反弹高度	m
θ_1	绳子与初始时鼓面位置的夹角	°
θ_2	绳子与鼓面位置最高时的夹角	°
F	每个人用力力度	N
n	参与活动人数	个
Δx	队员之间的距离	m
l	绳长	m

表 1 符号说明

5. 问题分析与求解

5.1 问题一

5.1.1 问题分析

考虑到理想状态下，由于没有能量损失，球与鼓的碰撞应为完全弹性碰撞，否则颠球的高度没有上限。在保证颠球的高度不低于 40 cm 的情况下，为了使得队员们颠球数量更多且更省力，应使颠球高度尽量低，所以我们假定颠球上升的恒定高度为 40 cm。由于题目给出能够精确控制用力方向、力度和时机，因而假定每个人的用力方向和力度一致，同一时机拉绳，但竖直方向的分力值随绳与鼓面夹角变化而变化，因此建立微分方程得到力度与球速度的关系。

5.1.2 模型建立

牛皮鼓与球的碰撞为**完全弹性碰撞**（如图 1），利用动量定理和动能定理建立方程：

(式中：球和鼓碰撞后速度大小均与碰撞前速度大小相等，方向相反)

$$\begin{cases} Mv_1 + mv_2 = Mv_1' + mv_2' \\ \frac{1}{2}Mv_1^2 + \frac{1}{2}mv_2^2 = \frac{1}{2}Mv_1'^2 + \frac{1}{2}mv_2'^2 \\ v_1 = -v'_1 \\ v_2 = -v'_2 \end{cases} \quad (1)$$

在进行活动的过程中，在参与人员手的高度相同的情况下，将此高度视为一个平面，则鼓的质心在运动时可有如图的三种情况——**低于手的平面**、**和手处于同一平面**、**高于手的平面**，即图 1 中红色方块①②③分别表示颠球过程中鼓面上升的三个位置。

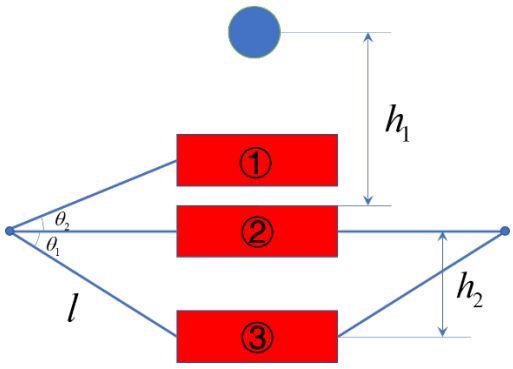

图 1 颠球过程图

由问题分析可知，h_1=40 cm，小球接触鼓面前一直做自由落体运动，可以得到：

$$\begin{cases} v_2 = gt \\ mgh_1 = \dfrac{1}{2}mv_2^2 \end{cases} \tag{2}$$

队员在做颠球活动时，力在竖直方向的分力随θ 变化而变化（如图 1），对鼓的运动进行分析，鼓在上升过程中做**变加速运动，**运用动能定理建立方程：

$$\int_{\theta_1}^{\theta_2} nFsin\theta \cdot \frac{1}{2} \cdot \frac{nFsin\theta}{M} \int_0^t t^2 d\theta dt - Mgh_2 = \frac{1}{2}Mv_1^2 \tag{3}$$

鼓在竖直方向上的分力一直变化，即鼓在上升过程中做**变加速运动**，令

$$F' = n\int_{\theta_1}^{\theta_2} Fsin\theta d\theta = nF(cos\theta_1 - cos\theta_2)$$

则满足（4）式方程：

$$\int_0^t \frac{F'}{M} tdt = h_2 \tag{4}$$

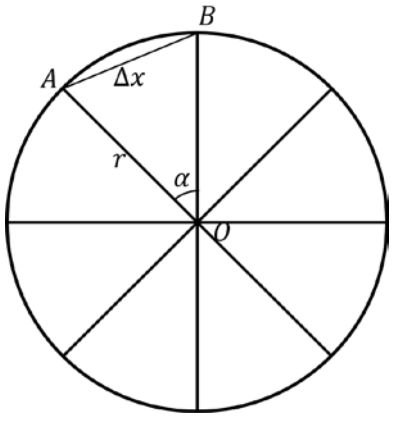

图 2 队员站位俯视图

题目给定队员之间的距离Δx不得小于 60 cm，图 2 表示队员围成圆圈后的俯视图，易看出$\widehat{AB} > \Delta x$。r为队员所围圆的半径。

$$\begin{cases} \boldsymbol{r = lcos\theta} \\ \boldsymbol{\frac{2\pi r}{n} > \Delta x} \end{cases} \quad (5)$$

由三角形余弦定理得：

$$\begin{cases} \boldsymbol{\alpha = \frac{2\pi}{n}} \\ \boldsymbol{cos\alpha = \frac{r^2 + r^2 - \Delta x^2}{2r \cdot r}} \end{cases} \quad (6)$$

5.1.3 问题求解

由方程组（1）可得：

$$\boldsymbol{Mv_1 + mv_2 = 0} \quad (7)$$

由方程组（2）解得：

$$\boldsymbol{v_2 = \sqrt{2gh_1} = 2.8m/s} \quad (8)$$

$$\boldsymbol{t = \frac{v_2}{g}} \quad (9)$$

通过实践和查看有关视频，在颠球过程中鼓面的最高位置与绳子基本保持水平，所以我们假定$\theta_2 = 0^{\circ}$。由方程（3）和（4）得到：

$$\boldsymbol{\frac{n^2F^2}{3M} \cdot \left(\sqrt{\frac{2h_1}{g}}\right)^3 \cdot \left(\frac{\pi}{4} - \frac{\theta_1}{2} + \frac{1}{4}sin2\theta_1\right) = 2h_1nFcos\theta_1 + Mv_1^2} \quad (10)$$

题目给定队员人数不少于 8 人，队员间距不得小于 60 cm，我们假定队员人数为 n，队员间距为 60 cm，则此时可由方程组（6）计算出半径 r 的表达式：

$$\boldsymbol{r = \sqrt{\frac{0.18}{1 - cos\left(\frac{2\pi}{n}\right)}}} \quad (11)$$

问题 2 中设绳长为 1.7m，在问题 3 中需要对比调整问题 1，因此**我们在问题 1 中也将绳长设为 1.7m**。由方程组（5）可得：

$$\frac{2\pi}{n}\cdot\sqrt{\frac{0.18}{1-\cos\left(\frac{2\pi}{n}\right)}}>\Delta x\geq 0.6 \tag{12}$$

将式（12）化简为：

$$n\sqrt{1-\cos\left(\frac{2\pi}{n}\right)}\leq 4.44 \tag{13}$$

令 $f(n)=n\sqrt{1-\cos\left(\frac{2\pi}{n}\right)}$，对 $f(n)$ 进行求导，可知其为增函数，求解得到 n=30 满足条件。由方程组（5）得：

$$\frac{\cos\theta}{n}\geq 0.56 \tag{14}$$

令 $8\leq n\leq 30$，使用由 python 代码编写的程序（见附录）求解式（**14**），得到了满足模型的 n 的上限值为 17，即得到 n 的范围应该为 $8\leq n\leq 17$，其中 n 与 θ 的对应关系见表 2。

符号	数值									
n	8	9	10	11	12	13	14	15	16	17
θ / °	63.38	59.74	55.94	51.98	47.78	43.28	38.37	32.86	26.36	17.82

表 2 n 与θ的对应关系

由表 1 可知当人数越多时θ值越小，即队员在颠球过程中鼓的运动幅度越小，队员越省力，所以我们的**最佳策略为参与活动人数 17 人，出力方向与水平面夹角 17.82°**。将已知数值代入式（10）中解得 F=13 N，即**每个队员出力力度为 13N，队员同时出力，颠球高度恒为 40 cm。**

5.2 问题二

5.2.1 问题分析

在现实状态下，我们不能保证每个人的用力时机，力度和方向是相同的，因此就会出现鼓面倾斜现象。对题目中已知条件分析可以得知，牛皮鼓的初始状态是鼓面与绳的末端处于同一水平面上，根据这一条件，我们就可以得出初始状态下绳子与水平面的夹角和每个人需提供的力度，从而进行下面的计算。

我们将牛皮鼓看做一个刚体，对其进行受力分析，当牛皮鼓各个方向受力不均衡时，会发生倾斜现象，究其原因，当合力方向为竖直向上时力与力矢成直角，合力矩为零，不会使其发生转动；而在此题中，合力方向不再为竖直向上，出现偏转，力与力矢之间的夹角不再为 90°，合力矩不为 0，这时物体就要围绕某一点或某一轴产生转动。

在以上条件的基础上，把鼓看做由一个薄壁圆筒和两个薄圆盘所组成的物体，此时牛皮鼓绕鼓的质心与筒面垂直的转轴进行转动，同时，竖直方向上的分力会使牛皮鼓产生向上的平动位移。因此，对于这道题来说，我们最终把它简化为**刚体定轴转动和竖直方向的平动位移问题**进行分析和求解。

5.2.2 模型建立与求解

首先，我们对已知条件进行分析。在初始状态时，鼓面初始时刻是水平静止的，初始位置较绳子水平时下降 11 cm，因此我们对绳与鼓的连接点之一进行受力分析（如图 3），受力分析如下：

$$\begin{cases} nT\sin\alpha = Mg \\ l\sin\alpha = h_3 \end{cases} \tag{15}$$

题目中已给出$n = 8$，$l = 1.7$，$h_3 = 0.11$的条件，故可解得：

$$\alpha = 3.71°$$

$$T = 69.55N$$

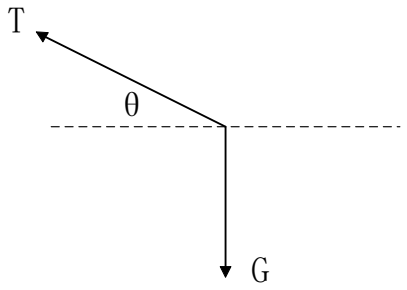

图 3 绳与鼓连接点的受力分析

经过上面的分析，牛皮鼓会进行定轴转动，那么一定满足转动定律，即刚体绕定轴转动时，刚体的角加速度与它所受合外力矩成正比，与刚体的转动惯量成反比。所以，我们需要算出牛皮鼓的转动惯量。经过查阅资料和咨询商家可知，制鼓所用牛皮的质量密度约为$\rho = 0.25kg/m^2$，厚约 $2mm$，鼓身则看为薄壁圆筒，忽略其厚度，进行转动惯量的求解。求解过程如下：

首先对鼓面分析，鼓面可看做圆盘，那么转轴沿直径的圆盘的转动惯量为：

$$J_1 = \int_0^R \rho\pi r^3 dr = \frac{\pi\rho R^4}{4} = m_1 R^2/4 \tag{16}$$

根据平行轴定理，可得鼓面对质心的转动惯量为：

$$J_1' = J_1 + m_1 h_3^2 \tag{17}$$

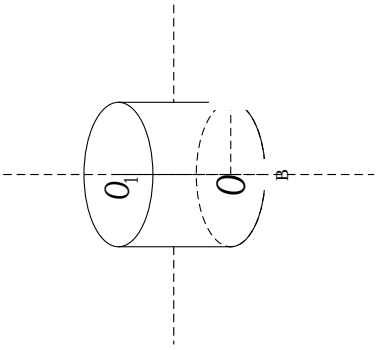

图 4 鼓的转动惯量模型图

那么对于鼓身来说，薄壁圆筒即为空心圆柱体，转轴通过质心与几何轴垂直，其转动惯量为

$$J_2 = \frac{m_2 R^2}{4} + \frac{4m_2 {h_3}^2}{12} \tag{18}$$

则总转动惯量为：

$$J = 2J_1' + J_2 \tag{19}$$

解得：

$$J = 0.05455 kg \cdot m^2$$

对于定轴转动的牛皮鼓，满足角动量定理，同样的，在竖直方向上满足动量定理，故可得以下式子：

$$\begin{cases} \int_{t_0}^{t_0+\Delta t} \vec{M} dt = \int_{t_0}^{t_0+\Delta t} J\vec{\omega}(t) dt \\ \int_{t_0}^{t_0+\Delta t} \vec{F} dt = \int_{t_0}^{t_0+\Delta t} m\vec{v}(t) dt \end{cases} \tag{20}$$

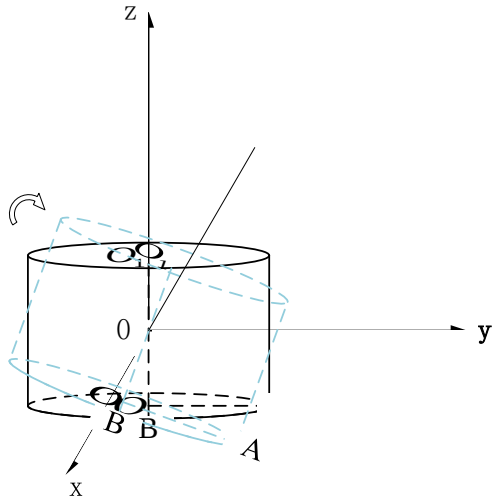

图 5 鼓的转动状态图

若想对上述两式求解，则需求出力矩$\vec{M}$，$\vec{F}$，$\vec{\omega}$（t），$\vec{v}$（t），那么我们首先设转动轴的转动角为$\vec{\phi}=$（x，y，z），且物体竖直向上平动的位移$\vec{s}=$（s_x，s_y，s_z），一共有 8 位队员，我们可以得到每位队员的位置坐标，用$\vec{e_i}$（i ∈ (1,8) 且 i ∈ Z^+）表示

$$\vec{M}=\vec{F}\times\vec{\tau} \tag{21}$$

因为在 0.1 的范围内，每时每刻力矢$\vec{\tau}$都在发生变化，是一个动态的过程，所以我们需要在每个时刻力矢的变化，因此，我们可以求出每个人在$t+\Delta t$时间时的力矢：

$$\vec{\tau_i}=\vec{\phi}\times r\vec{e_i}+\ r\vec{e_i} \tag{22}$$

同样的，对于力的计算，要先计算出每个人在$t+\Delta t$时竖直方向上的分力，因为随着鼓的转动，质心向上发生平移，因此需要通过人施力时的位置与绳和鼓身节点位置来确定$\vec{F_{is}}$：

人施力处位置：

$$\begin{cases} \overrightarrow{\boldsymbol{Q_i}} = \overrightarrow{\boldsymbol{\tau_i}} + \vec{\boldsymbol{s}} \\ \overrightarrow{\boldsymbol{P_i}} = \sqrt{\boldsymbol{l^2 - h_3^2}} * \overrightarrow{\boldsymbol{e_i}} + (\boldsymbol{0，\ 0，\ h_3}) \\ \overrightarrow{\boldsymbol{F_{is}}} = \dfrac{\overrightarrow{\boldsymbol{F_i}} - \overrightarrow{\boldsymbol{F_i}}}{\left|\overrightarrow{\boldsymbol{F_i}} - \overrightarrow{\boldsymbol{F_i}}\right|} * \overrightarrow{\boldsymbol{F_i}} \end{cases} \tag{23}$$

然后我们运用力矩公式求出每个方向的力矩，最后进行力矩的合成

$$\mathbf{MT} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{8}} \overrightarrow{\boldsymbol{F_{is}}} \times \overrightarrow{\boldsymbol{\tau_i}} \tag{24}$$

最后，根据$\vec{\omega} = \frac{d\vec{\emptyset}}{\mathrm{dt}}$和$\vec{v} = \frac{d\vec{s}}{\mathrm{dt}}$可得

$$\begin{cases} \vec{\emptyset}(\boldsymbol{t + \Delta t}) = \vec{\emptyset}(\boldsymbol{t}) + \int_{\boldsymbol{t_0}}^{\boldsymbol{t_0+\Delta t}} \boldsymbol{\omega\ （t）\ dt} \\ \vec{\boldsymbol{s}}(\boldsymbol{t + \Delta t}) = \vec{\boldsymbol{s}}(\boldsymbol{t}) + \int_{\boldsymbol{t_0}}^{\boldsymbol{t_0+\Delta t}} \boldsymbol{v\ （t）\ dt} \end{cases} \tag{25}$$

模型求解：我们利用迭代式算法，把 0.1s 等分为 n 份，计算每一个Δt s后，即$\mathrm{t} + \Delta\mathrm{t}$时刻的转矩，建立出关于$\vec{\emptyset}$的关系，依次迭代，得到最后的偏转角 θ 。计算结果如下：

序号	用力参数	1	2	3	4	5	6	7	8	鼓面倾角（度）
1	发力时机	0	0	0	0	0	0	0	0	0.183
	用力大小	**90**	80	80	80	80	80	80	80	
2	发力时机	0	0	0	0	0	0	0	0	0.314
	用力大小	**90**	**90**	80	80	80	80	80	80	
3	发力时机	0	0	0	0	0	0	0	0	0.146
	用力大小	**90**	80	80	**90**	80	80	80	80	
4	发力时机	**-0.1**	0	0	0	0	0	0	0	0.056
	用力大小	80	80	80	80	80	80	80	80	
5	发力时机	**-0.1**	**-0.1**	0	0	0	0	0	0	0.184
	用力大小	80	80	80	80	80	80	80	80	
6	发力时机	**-0.1**	0	0	**-0.1**	0	0	0	0	0.036
	用力大小	80	80	80	80	80	80	80	80	
7	发力时机	**-0.1**	0	0	0	0	0	0	0	0.202
	用力大小	**90**	80	80	80	80	80	80	80	
8	发力时机	0	**-0.1**	0	0	**-0.1**	0	0	0	0.036
	用力大小	**90**	80	80	**90**	80	80	80	80	
9	发力时机	0	0	0	0	**-0.1**	0	0	**-0.1**	0.036
	用力大小	**90**	80	80	**90**	80	80	80	80	

5.3 问题三

5.3.1 问题分析

问题一为理想状态下的最佳策略，问题二为现实情况下人会出现力度和时机上的偏差，也就导致了倾斜现象。那么问题一在现实情况下，力度和时机的偏差不可避免，那么我们只能从原系统入手，尽可能减小时机和力度不合适所形成的偏斜角。因此，若要减小系统误差，我们可以从绳长，人数，鼓的初始位置三个方面进行优化，来达到缩小夹角的目的。我们认为在问题一策略前提下，人数和绳长是不可更改的，是一个定值，那么我们只能从鼓的初始位置和鼓相对于球开始运动的时刻进行优化。

5.3.2 模型建立

根据问题二的模型，我们只需改变问题二模型中人数以及鼓的初始高度即可。人数为定值，鼓的初始高度是我们所需要优化的变量，因此我们对问题二中的模型进行了更新。我们选择以高度为优化量，偏斜角为目标函数，得到了新模型，来求解高度的最合适值。

首先设鼓的初始位置距施力的水平面距离为 x。

鼓的转动惯量，角动量定理和竖直方向上的动量定理同题目二中模型，故可得以下式子：

$$\boldsymbol{J = 0.05455kg \cdot m^2}$$

若想对方程组（21）求解，则需求出力矩$\overrightarrow{\boldsymbol{M}}$，$\vec{\boldsymbol{F}}$，$\vec{\boldsymbol{\omega}}$（$\boldsymbol{t}$），$\vec{\boldsymbol{v}}$（$\boldsymbol{t}$），那么我们首先设转动轴的转动角为$\vec{\boldsymbol{\emptyset}} =$（$\boldsymbol{x}$，$\boldsymbol{y}$，$\boldsymbol{z}$），且物体竖直向上平动的位移$\vec{\boldsymbol{s}} =$（$\boldsymbol{s_x}$，$\boldsymbol{s_y}$，$\boldsymbol{s_z}$），一共有 8 位队员，我们可以得到每位队员的位置坐标，用$\overrightarrow{e_i}$（i ∈ (1,8)且 i ∈ Z^+）表示

$$\overrightarrow{\boldsymbol{M}} = \vec{\boldsymbol{F}} \times \vec{\boldsymbol{\tau}} \qquad (26)$$

因为在 0.1 的范围内，每时每刻力矢$\vec{\tau}$都在发生变化，是一个动态的过程，所以我们需要在每个时刻力矢的变化，因此可以求出每个人在$\boldsymbol{t + \Delta t}$时间时的力矢：

$$\overrightarrow{\boldsymbol{\tau_i}} = \vec{\emptyset} \times \mathbf{r}\overrightarrow{\boldsymbol{e_i}} + \mathbf{r}\overrightarrow{\boldsymbol{e_i}} \qquad (27)$$

同样的，对于力的计算，要先计算出每个人在$\boldsymbol{t + \Delta t}$时竖直方向上的分力，因为随着鼓的转动，质心向上发生平移，因此需要通过人施力时的位置与绳和鼓身节点位置来确定$\overrightarrow{F_{is}}$：

人施力处位置：

$$\begin{cases} \overrightarrow{\boldsymbol{Q_i}} = \overrightarrow{\boldsymbol{\tau_i}} + \vec{\boldsymbol{s}} \\ \overrightarrow{\mathbf{P_1}} = \sqrt{\mathbf{l^2 - x^2}} * \overrightarrow{\mathbf{e_1}} + (\mathbf{0},\ \mathbf{0},\ \mathbf{x}) \\ \overrightarrow{\boldsymbol{F_{is}}} = \dfrac{\overrightarrow{\boldsymbol{P_i}} - \overrightarrow{\boldsymbol{Q_i}}}{\left|\overrightarrow{\boldsymbol{P_i}} - \overrightarrow{\boldsymbol{Q_i}}\right|} * \overrightarrow{\boldsymbol{F_i}} \end{cases} \qquad (28)$$

然后我们运用力矩公式求出每个方向的力矩，最后进行力矩的合成

$$\mathbf{MT} = \sum_{\boldsymbol{i=1}}^{\mathbf{8}} \overrightarrow{\boldsymbol{F_{is}}} \times \overrightarrow{\boldsymbol{\tau_i}} \qquad (29)$$

最后，根据$\vec{\omega} = \frac{d\vec{\emptyset}}{\mathrm{dt}}$和$\vec{v} = \frac{d\vec{s}}{\mathrm{dt}}$可得

$$\begin{cases} \vec{\emptyset}(t+\Delta t)=\vec{\emptyset}(t)+\int_{t_0}^{t_0+\Delta t}\omega\ (t)\ dt \\ \vec{s}(t+\Delta t)=\vec{s}(t)+\int_{t_0}^{t_0+\Delta t}v\ (t)\ dt \end{cases} \tag{30}$$

我们把偏离角看做目标函数，使其最小化，即：

$$\min=\min\{\vec{\theta}\}$$

5.3.3 模型求解

根据上述条件，本题就变成了我们所熟知的使目标函数最小的非线性规划问题，利用 Python 中的非线性规划的 scipy.optimize.minimize 算法对鼓的初始高度进行了优化。并使用 O 梅森旋转法使每个人随机发生用力时机和力度的偏差进行验证，验证结果如下（代码见附录）：

人数 / 组别	1	2	3	4	5	6	7	8	9	10
1	13	13	13	13	13	13	13	13	13	13
2	13	25	13	11	13	28	15	13	17	26
3	13	13	13	13	15	13	13	13	17	13
4	13	13	13	22	13	24	13	13	13	24
5	13	13	13	23	27	13	13	13	15	10
6	28	19	13	13	13	13	13	13	13	13
7	13	14	13	26	13	28	13	13	13	21
8	27	13	13	23	13	15	13	13	13	23
9	13	13	13	12	23	13	13	13	13	15
10	26	13	13	13	17	13	13	13	25	13
11	13	13	13	10	22	24	13	28	13	13
12	13	13	13	11	13	18	13	13	19	24
13	13	14	13	13	13	13	13	13	18	24
14	22	15	13	13	24	23	13	13	22	13
15	27	17	13	20	23	27	13	19	13	20
16	13	12	13	13	13	16	13	23	13	13
17	13	19	13	13	13	23	11	13	13	22
优化前θ	0.421037	0.037702	1.87E-07	0.157363	0.055377	0.24996	1.39E-06	0.079234	0.092939	0.008189
优化后θ	0.001341	0.000221	1.39E-06	0.002685	0.012089	0.078481	7.23E-08	4.46E-09	0.019127	0.005737
策略	0.154037	0.081334	1	0.127518	0.103515	0.132502	1	0.087937	0.088973	0.108576

表 3 检验算法表

从表中可以看出，当我们改变牛皮鼓的初始位置，偏转角 θ 有明显程度的变小，证明优化结果是极其有效的，所以我们对牛皮鼓初始位置的优化是极其成功的。

5.4 问题四

5.4.1 问题分析

颠球过程中鼓面发生倾斜，说明队员们的发力大小不同。鼓面倾斜方向在水平面的投影指向某两队员之间，即这两名队员与其他队员发力大小或时机不同。投影方向与两位队员的夹角比为 1：2，可以建立这两名队员发力力矢的关系。为使球调整为竖直方向弹跳，此时将鼓面作为反射面，得到鼓转动的角度。此问题与问题二相似，只是力的大小不确定，已知鼓的倾斜角，我们可以再次利用问题二的方法，假定一部分力的大小大于另一部力的大小，计算出发力大小。

5.4.2 模型建立与求解

球的反弹高度为 60 cm，此时我们可计算得到小球接触鼓面需要的时间为：

$$t = \sqrt{\frac{2h_3}{g}} = 0.35s$$

由问题二题设可知，0.35s 足够一次将鼓面调成水平。

小球反弹后再落下做的是抛物线运动，由抛物线运动规律得，反弹前和反弹后与竖直方向夹角相同（如图 6），即$\beta_1 = \beta_2 = 1°$。为了将球调整为数值状态弹跳，应调节鼓面倾斜的角度，如图 7，鼓面倾斜角度为$\beta = 0.5°$。

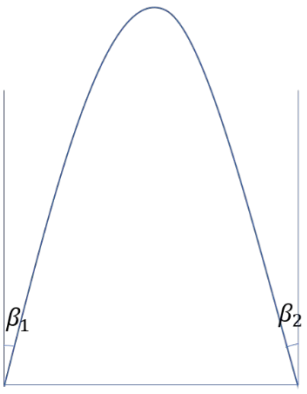

图 6 球的运动曲线图

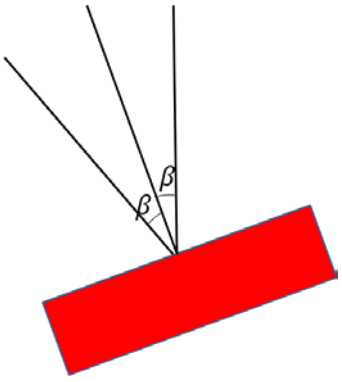

图 7 鼓面调整图

题目给定可精确控制队员的发力时机和力度，在本题中我们假设发力时机相同，所以鼓发生倾斜一定是队员的发力大小不同。套用问题二的模型，我们随机给出各个队员发力大小。已知 n=10，$l=2$，$h_3=0.11$，由方程组（15）解得在鼓静止时，各个队员的牵拉力为 65.455 N。运行 Python 程序时，我们给出的随机牵拉力为65.455 ± 25 N，同时队员中具体哪个人与其他人的发力大小不同我们不清楚，所以先随机是哪个队员出了错误，其次给这些出错的队员随机的力，在程序中模拟得到与θ值最接近的一组发力大小解。此时$\theta=0.49°$。

符号	数值									
n	1	2	3	4	5	6	7	8	9	10
F/N	65.45	71.45	65.45	60.45	54.45	70.45	49.45	62.45	65.45	65.45

表 4 每个队员的发力大小

参考文献

[1]程守洙，江之永.《普通物理学 上》，高等教育出版社，第七版

[2]张启仁.《理论物理学-经典力学》-科学出版社-2001

[3][美]Н.С.巴赫瓦洛夫 /Н.П.热依德科夫，《数值方法》，高等教育出版社

[4]王子文《变截面椭圆形独塔斜拉桥塔_梁结合段模型试验研究》中外公路 39-4, 2019.08

[5]李哲，高君《单足机器人垂直跳跃落地碰撞冲击力分析》哈尔滨工业大学学报 49-7,2017.07

[6]吴萍，冯卓宏《定轴转动刚体的碰撞动力学实验研究》实验室科学 22-2，2019.04

[7][M] 吴军《数学之美》，人民邮电出版社，2012.06

[8]https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html

附录

Answer_Q1.py —求解问题一程序

此程序段代码用用以求解式 $\frac{cos\theta}{n} \geq 0.56$中 θ 与人数 n 的关系，并最终得到了满足模型的 n 的上限值。

```
import math
import numpy as np
PI = 3.1415926

# 函数声明
def GetMaxNum_of_People():
    n = 8       # 至少8人
    List1 = []
    List2 = []
    while True:
        # np.arccos方法在遇到无法求解的情况时值为NaN
        a = np.arccos(0.056 * n)
        a = a * (180/PI)
        List1.append([a,n])

        try:
            # math.acos方法在遇到无法求解的情况时会报错
            b = math.acos(0.056 * n)
            b = b * (180/PI)
            List2.append([b,n])
        except:
            pass
```

```
            n += 1
            if n > 20:
                break
        print(List1)
    print(List2)

    # 运行程序
    GetMaxNum_of_People()
```

Answer_Q2.py —— 求解问题二的程序

此段程序用以求解模型中的公式，最终得到不同情况下的鼓在 0.1 秒时产生的相对于水平面的倾斜角度。

```
    import numpy as np
    import math

    # F0 = 所有人同时发力时的发力列表
    # t = 每个人的发力时机
    # n = 参加的人数

    def GetAnswer_2(F0,t,n):
        # 默认值
        PI = 3.1415926
        g = 9.8    # 重力加速度

        # 鼓的参数
        r = 0.2   # 鼓的半径
        M = 3.6   # 鼓的质量
        l = 1.7   # 绳长
```

```
J = 0.05455 # 转动惯量（自定义的量） 0.0885

# BF为基础拉力 0.11 为鼓身高度的半
BF = l * M *10 / (0.11*n)

# 获取每个人站位的单位向量
e = []
for i in range(1,n+1):
    e.append(np.mat([math.cos(i*2*PI/n),math.sin(i*2*PI/n),0]))

# 角度的初始值及变化后的值
Q = Q_t = np.mat([0,0,0])

# 初始时刻的转动惯量
wt0 = wt0_t = 0

# 初始状态下鼓倾斜的上升距离
s = s_t = 0

# 初始状态下鼓倾斜时的上升速度
v = v_t = 0

# 表示人手的位置
Hand = []
for k in range(0,n):
    Hand.append((r + l) * e[k] + np.mat([0,0,0.11]))

FList = []
# 初始状态下所有人拉住鼓的基础拉力的列表
```

```
        for i in range(0,n):
            FList.append(BF)

    # 时间片划分的标注，在循环中
    # 值为100时表示将0~0.1的时间进行等间隔的100次剖分
        TimeSlice = 100
        for i in range(0,n):
            # 若有人提前发力，则剖分的时间片为 0~0.2
            if t[i] != 0:
                TimeSlice = 200
                # 若有人提前发力则将对应人的拉力更改为他发力时的拉力
                FList[i] = F0[i]

    # 若有人提前发力（TimeSlice != 100）
    # 则将更新后的基础拉力列表提供给变量 F
        # 若无人提前发力，则将同时发力时发力列表提供给变量 F
        if TimeSlice != 100:
            F = FList
        else:
            F = F0

        # 划分 0 ~ 0.1 时间为100份
    # 若有人提前施加拉力，则将施加拉力的时刻定为0
    # 并将总时间 0~0.2 部分划分200份
        dt = 0.001

        for t0 in range(0, TimeSlice):
            # 若TimeSlice == 100时所有人开始发力，则将拉力情况更换为F0
            if TimeSlice == 100:
```

```
            F = F0
        # 绳到质心的矢量
        sl = []
        for k in range(0,n):
            sl.append(r * np.cross(Q,e[k]) + r * e[k] +
np.mat([0,0,np.linalg.norm(s)]))

        Ft = np.mat([0,0,0])
        for k in range(0,n):
            Ft = Ft + ((Hand[k] - sl[k]) / np.linalg.norm(Hand[k] - sl[k])) *
F[k]
        Ft = Ft + M * g * np.mat([0,0,-1])
        v_t = dt * Ft / M + v
        s_t = s + v * dt

        Mt = np.mat([0,0,0])
        for k in range(0,n):
            Mt = Mt + np.cross((r * np.cross(Q,e[k]) + r * e[k]) , ((Hand[k] -
sl[k]) / np.linalg.norm(Hand[k] - sl[k])) * F[k])

        wt0_t = wt0 + dt*Mt/J
        Q_t = Q + dt*wt0

        # 下一时刻将计算出来的值作为已知量
        s = s_t
        v = v_t
        Q = Q_t
        wt0 = wt0_t
```

```
    return np.linalg.norm(Q) * 180 / PI


# 设置表1中的数据
InputList = [[90,80,80,80,80,80,80,80],[0,0,0,0,0,0,0,0],
                [90,90,80,80,80,80,80,80],[0,0,0,0,0,0,0,0],
                [90,80,80,90,80,80,80,80],[0,0,0,0,0,0,0,0],
                [80,80,80,80,80,80,80,80],[-0,1,0,0,0,0,0,0],
                [80,80,80,80,80,80,80,80],[-0.1,-0.1,0,0,0,0,0,0],
                [80,80,80,80,80,80,80,80],[-0.1,0,0,-0.1,0,0,0,0],
                [90,80,80,80,80,80,80,80],[-0.1,0,0,0,0,0,0,0],
                [90,80,80,90,80,80,80,80],[0,-0.1,0,0,-0.1,0,0,0],
                [90,80,80,90,80,80,80,80],[0,0,0,0,-0.1,0,0,-0.1]]

# 循环读入表1中的数据并输出结果
for i in range(0,18,2):
print(GetAnswer_2(InputList[i],InputList[i + 1],8))
```

Answer_Q3.py —— 求解问题三

此段代码采用随机算法随机给出问题一会出现的误差值，之后使用问题二给出的模型对偏差进行计算，最后使用优化算法，优化问题一的模型，给出优化的方案以及优化后与优化前的结果对比。

```
import numpy as np
import math
from scipy.optimize import minimize
from time import time
```

```
#var
index = 624
MT = [0]*index
# MT[0] ->seed

def inter(t):
    return(0xFFFFFFFF & t) #取最后32位->t

def twister():
    global index
    for i in range(624):
        y = inter((MT[i] & 0x80000000) +(MT[(i + 1) % 624] & 0x7fffffff))
        MT[i] = MT[(i + 397) % 624] ^ y >> 1
        if y % 2 != 0:
            MT[i] = MT[i] ^ 0x9908b0df
    index = 0

def exnum():
    global index
    if index >= 624:
        twister()
    y = MT[index]
    y = y ^ y >> 11
    y = y ^ y << 7 & 2636928640
    y = y ^ y << 15 & 4022730752
    y = y ^ y >> 18
    index = index + 1
    return inter(y)
```

```
def mainset(seed):
    MT[0] = seed        #seed
    for i in range(1,624):
        MT[i] = inter(1812433253 * (MT[i - 1] ^ MT[i - 1] >> 30) + i)
    return exnum()

def GetRandom(minnum,maxnum):
    return minnum + int((maxnum-minnum)*(mainset(int(time())) / (2**32-1)))

# 产生随机的误差数据
def RandomData():
    F0 = [13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13]
    t = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
    count = GetRandom(0,17)
    for i in range(count):
        F0[GetRandom(1,17)] = GetRandom(10,30)

    for i in range(count):
        if GetRandom(1,17)%2 == 0:
            t[GetRandom(1,17)] = -0.1

    return F0,t

# 获得鼓与水平面可能产生的倾角
def GetQ(F0,t,hs):

    n = 17
    PI = 3.1415926
    g = 9.8     # 重力加速度
```

```
# 鼓的参数
r = 0.2    # 鼓的半径
M = 3.6    # 鼓的质量
l = 1.7    # 绳长
J = 0.05455 # 转动惯量（自定义的量） 0.0885

# 获取每个人站位的单位向量
e = []
for i in range(1,n+1):
    e.append(np.mat([math.cos(i*2*PI/n),math.sin(i*2*PI/n),0]))

# 角度的初始值及变化后的值
Q = Q_t = np.mat([0,0,0])

# 初始时刻的转动惯量
wt0 = wt0_t = 0

# 初始状态下鼓倾斜的上升距离
s = s_t = 0

# 初始状态下鼓倾斜时的上升速度
v = v_t = 0

# 表示人手的位置
Hand = []
for k in range(0,n):
    Hand.append(math.sqrt(l * l + hs * hs) * e[k] + np.mat([0,0,hs]))
```

```
FList = []
# 初始状态下所有人拉住鼓的基础拉力的列表
for i in range(0,n):
    FList.append(13)

# 时间片划分的标注，在循环中，值为100时表示将0~0.1的时间进行等间隔的100次剖分
TimeSlice = 100
for i in range(0,n):
    # 若有人提前发力，则剖分的时间片为 0~0.2
    if t[i] != 0:
        TimeSlice = 200
        # 若有人提前发力则将对应人的拉力更改为他发力时的拉力
        FList[i] = F0[i]

# 若有人提前发力（TimeSlice != 100）则将更新后的基础拉力列表提供给变量 F
# 若无人提前发力，则将同时发力时发力列表提供给变量 F
if TimeSlice != 100:
    F = FList
else:
    F = F0

# 划分 0 ~ 0.1 时间为100份
# 若有人提前施加拉力，则将施加拉力的时刻定为0，并将总时间 0~0.2 部分划分200份
dt = 0.001

for t0 in range(0, TimeSlice):
```

```
            # 若TimeSlice == 100时所有人开始发力，此时则将拉力情况更换为F0
            if TimeSlice == 100:
                F = F0
            # 绳到质心的矢量
            sl = []
            for k in range(0,n):
                sl.append(r * np.cross(Q,e[k]) + r * e[k] + np.mat([0,0,np.linalg.norm(s)]))

            Ft = np.mat([0,0,0])
            for k in range(0,n):
                Ft = Ft + ((Hand[k] - sl[k]) / np.linalg.norm(Hand[k] - sl[k])) * F[k]

            Ft = Ft + M * g * np.mat([0,0,-1])
            v_t = dt * Ft / M + v
            s_t = s + v * dt

            Mt = np.mat([0,0,0])
            for k in range(0,n):
                Mt = Mt + np.cross((r * np.cross(Q,e[k]) + r * e[k]) , ((Hand[k] - sl[k]) / np.linalg.norm(Hand[k] - sl[k])) * F[k])

            wt0_t = wt0 + dt*Mt/J
            Q_t = Q + dt*wt0

            # 下一时刻将计算出来的值作为已知量
            s = s_t
```

```
            v = v_t
            Q = Q_t
            wt0 = wt0_t

        return np.linalg.norm(Q) * 180 / PI

    def Func(hs):
        # 随机获得带有误差的数据
        F0,t = RandomData()
        print("F0 = ",F0)
        print("t = ",t)
        return F0,t,GetQ(F0,t,hs)

    def TestQ(args):
        F0,t = args
        Q = lambda hs:GetQ(F0,t,hs)
        return Q

    file_name = 'Question_3_data.txt'
    with open(file_name,'w') as file_obj:
        for i in range(29,51):
            print("第",i+1,"组测试数据————————————————————————————————\n")
            file_obj.write("第"+str(i+1)+"组测试数据————————————————————————————————————————\n")
            F0,t,Q = Func(0.11)
            file_obj.write(str(F0)+"\n")
            print("优化前因误差产生的倾斜角度为："+str(Q))
            file_obj.write("优化前因误差产生的倾斜角度为："+str(Q))
```

```
        args = (F0,t)


        print("优化开始！")
        file_obj.write("优化开始！\n")
        v = TestQ
        x0 = np.asarray((10))
        cons = ({'type': 'ineq', 'fun': lambda s: 1 - s},{'type': 'ineq', 'fun': lambda
s: s})
        res = minimize(v(args), x0, method='SLSQP',constraints=cons)
        print("是否完成优化：",res.success)
        if res.success == True:
            file_obj.write("是否完成优化：True\n")
        else:
            file_obj.write("是否完成优化：False\n")
        print("改变策略的方法：令鼓的初始位置升高到",res.x)
        file_obj.write("改变策略的方法：令鼓的初始位置升高到
"+str(res.x)+"\n")
        Q = GetQ(F0,t,float(res.x))
        print("优化后会产生的倾斜角度为：",Q)
        file_obj.write("优化后会产生的倾斜角度为："+str(Q)+"\n")
        print("————————————————————————————————————
————————————————————————\n")
        file_obj.write("————————————————————————————————————
————————————————————————\n")
```

Answer_Q4.py ——— 求解问题四

此段代码通过随机数算法对题目中假设的数据进行处理，之后给出最终的结果。

```
import numpy as np
import math
from scipy.optimize import minimize
from time import time

#var
index = 624
MT = [0]*index
# MT[0] ->seed

def inter(t):
    return(0xFFFFFFFF & t) #取最后32位->t

def twister():
    global index
    for i in range(624):
        y = inter((MT[i] & 0x80000000) +(MT[(i + 1) % 624] & 0x7fffffff))
        MT[i] = MT[(i + 397) % 624] ^ y >> 1
        if y % 2 != 0:
            MT[i] = MT[i] ^ 0x9908b0df
    index = 0

def exnum():
    global index
    if index >= 624:
        twister()
    y = MT[index]
```

```
        y = y ^ y >> 11
        y = y ^ y << 7 & 2636928640
        y = y ^ y << 15 & 4022730752
        y = y ^ y >> 18
        index = index + 1
        return inter(y)

    def mainset(seed):
        MT[0] = seed        #seed
        for i in range(1,624):
            MT[i] = inter(1812433253 * (MT[i - 1] ^ MT[i - 1] >> 30) + i)
        return exnum()

    def GetRandom(minnum,maxnum):
        return minnum + int((maxnum-minnum)*(mainset(int(time())) / (2**32-1)))

    def RandomData():
        n = 10
        M = 3.6   # 鼓的质量
        l = 2   # 绳长
        # BF为基础拉力 0.11 为鼓身高度的半
        BF = l * M *10 / (0.11*n)
        F0 = [BF,BF,BF,BF,BF,BF,BF,BF,BF,BF]
        t = [0,0,0,0,0,0,0,0,0,0]
        count = GetRandom(0,10)
        for i in range(count):
            F0[GetRandom(1,10)] = GetRandom(BF-
GetRandom(1,25),BF+GetRandom(1,25))
```

```
    return F0,t


def GetQ(F0,t,hs):

    n = 10
    PI = 3.1415926
    g = 9.8     # 重力加速度

    # 鼓的参数
    r = 0.2    # 鼓的半径
    M = 3.6    # 鼓的质量
    l = 1.7    # 绳长
    J = 0.0885 # 转动惯量（自定义的量） 0.0885

    BF = l * M *10 / (0.11*n)

    # 获取每个人站位的单位向量
    e = []
    for i in range(1,n+1):
        e.append(np.mat([math.cos(i*2*PI/n),math.sin(i*2*PI/n),0]))

    # 角度的初始值及变化后的值
    Q = Q_t = np.mat([0,0,0])

    # 初始时刻的转动惯量
    wt0 = wt0_t = 0

    # 初始状态下鼓倾斜的上升距离
```

```
        s = s_t = 0

        # 初始状态下鼓倾斜时的上升速度
        v = v_t = 0

        # 表示人手的位置
        Hand = []
        for k in range(0,n):
            Hand.append(math.sqrt(l * l + hs * hs) * e[k] + np.mat([0,0,hs]))

        FList = []
        # 初始状态下所有人拉住鼓的基础拉力的列表
        for i in range(0,n):
            FList.append(BF)

        # 时间片划分的标注，在循环中，值为100时表示将0~0.1的时间进行等间隔的100次剖分
        TimeSlice = 100
        for i in range(0,n):
            # 若有人提前发力，则剖分的时间片为 0~0.2
            if t[i] != 0:
                TimeSlice = 200
                # 若有人提前发力则将对应人的拉力更改为他发力时的拉力
                FList[i] = F0[i]

        # 若有人提前发力（TimeSlice != 100）则将更新后的基础拉力列表提供给变量 F
        # 若无人提前发力，则将同时发力时发力列表提供给变量 F
        if TimeSlice != 100:
```

```
            F = FList
        else:
            F = F0

        # 划分 0 ~ 0.1 时间为100份
        # 若有人提前施加拉力，则将施加拉力的时刻定为0，并将总时间
0~0.2 部分划分200份
        dt = 0.001

        for t0 in range(0, TimeSlice):
            # 若TimeSlice == 100时所有人开始发力，此时则将拉力情况更换
为F0
            if TimeSlice == 100:
                F = F0
            # 绳到质心的矢量
            sl = []
            for k in range(0,n):
                sl.append(r * np.cross(Q,e[k]) + r * e[k] +
np.mat([0,0,np.linalg.norm(s)]))

            Ft = np.mat([0,0,0])
            for k in range(0,n):
                Ft = Ft + ((Hand[k] - sl[k]) / np.linalg.norm(Hand[k] - sl[k])) *
F[k]

            Ft = Ft + M * g * np.mat([0,0,-1])
            v_t = dt * Ft / M + v
            s_t = s + v * dt
```

```
            Mt = np.mat([0,0,0])
            for k in range(0,n):
                Mt = Mt + np.cross((r * np.cross(Q,e[k]) + r * e[k]) , ((Hand[k] -
sl[k]) / np.linalg.norm(Hand[k] - sl[k])) * F[k])

            wt0_t = wt0 + dt*Mt/J
            Q_t = Q + dt*wt0

            # 下一时刻将计算出来的值作为已知量
            s = s_t
            v = v_t
            Q = Q_t
            wt0 = wt0_t

        return np.linalg.norm(Q) * 180 / PI

    def Func(hs):
        # 随机获得带有误差的数据
        F0,t = RandomData()
        return F0,t,GetQ(F0,t,hs)

    count = 1
    while True:
        F0,t,Q = Func(0.11)
        if Q >= 0.49 and Q <=0.51:
            print(F0)
            print(t)
            print(Q)
            break
```

```
print("执行完第",count,"次循环~")
count += 1
```