“同心协力”任务中的策略研究

摘　要

本文介绍了在“同心协力”项目背景下，针对多种不同情况的团队合作策略和鼓面修正方法。本文主要分为五个部分，第一、二部分对问题进行重述和简要分析，以确定主要研究思路和方法；第三部分给出一些基本假设，将同心鼓项目抽象化，便于后续的力学分析和数学建模；第四部分是具体的分析和建模过程，以及策略选取方法；第五部分是对于模型的整体评价。

针对问题一，由于假定所有人都可以精确的控制用力，故可以让所有人一直处于用力状态，同时每次击球点和击球高度均相同，这样可以形成稳定的周期循环。根据刚体碰撞规律，第一次碰撞选取合适的击球点可以保证球的撞击高度大于 40cm。后续只要保证鼓的加速度不变，且球和鼓撞击时动量和为 0 即可达到每次击球稳定。

针对问题二，将鼓抽象为一个柱面并置于坐标系进行受力分析，计算出各个受力阶段的力矩后，通过角加速度和角速度计算转角。其中由于每个阶段时间较短，角位移较小，故可以利用矢量公式计算一个阶段内转角。整体转角可以通过两次旋转法向量得到。

针对问题三，通过问题二结果中的数据对比可以发现，每个参与者之间的发力关系，包括发力大小，发力时机差距，不同发力者在圆形同心鼓一圈的角度分布可能会使本身因为一个发力者发力出现偏差而导致的鼓面偏移程度减小。因此在第三问中我们将条件限制的更为简单后分别进行数据的计算得出在一位参与者出错后其余参与者如何“最低成本”（(发力时间不宜过长，发力大小不宜过大) 条件下将同心鼓鼓面的偏移误差降低来达到策略优化，即保证第一问中整个让球弹起落下的过程顺利进行。

针对问题四，我们主要分析鼓在竖直方向上的合力，我们希望在力系合成之后对于系统的作用仅剩下能够使得鼓面按照特定的角度转动从而达到使球竖直的效果。通过搜索我们能找到十个人同时发力的调整方案。同时考虑现实生活中的可行性，也可以让其中某几个人进行调整，比如偏移投影方向的队员额外发力的方式达到效果。

关键词 同心鼓、刚体转动、力矩分析、力系合成

1 问题重述

1.1 问题背景

“同心协力”，或称同心鼓，是一项在公司团建，外出游玩时常见的培养团队协作能力的娱乐拓展项目。该项目由于可以同时让多人参与到活动中来，并且有着一定的难度与趣味性，因此受到很多人的喜爱。

同心鼓需要多人各持一根绳子，通过多人对绳的配合使用，来让具有弹性牛皮鼓面的同心鼓上下运动，进而使球在垂直方向弹起落下，并在多次弹起中保持一定的高度和稳定性来获得团队的胜利。

想要在同心鼓比赛中取得好的成绩，就需要团队中的每个成员根据鼓和绳具体的尺寸、以及期望的弹起高度进行当前碰撞时的发力情况的规划。同时在实际情况中，因为每个人的发力时间和真实大小会有误差，可能与预想的情况有一定出入，因此需要每一个人实时地根据当前情况进行模型分析，为下一次的预期碰撞位置和发力提供策略，来避免失败。

1.2 相关信息

典型的同心鼓的侧面图如下图所示，同心鼓是一个上下为等大圆面的圆柱体。同心鼓为了发出声音，其内部为空心结构，上下两面为了提供弹性由牛皮鼓面包裹，质量很轻，可以将其质量视为均匀分布在圆柱的一圈侧面上。每个人手持的长绳一端固定在鼓侧面的中间高度处，分析受力可知，绳子与鼓侧面应该保持一个不垂直的角度，来提供力使鼓在悬空时处于平衡的状态。每个参与者手持绳子末端，发力方向应为沿绳方向最为省力，因为其他方向的力也只有沿绳方向可以传导至鼓边沿使鼓受力，垂直于绳方向的力会被摩擦力及压力抵消。

经过对有关人员的问询我们了解到，每个同心鼓比赛的参与者与鼓的相对位置是不可以改变的，同时为了满足手只能握绳末端的要求，因此可以移动的应该是手臂而非人的整个身体。在理想的情况下，每个人的发力时机和发力大小可以保持一致，此时可以让绳和鼓侧面的夹角保持不变，同时鼓面保持水平，绳和鼓作为一个固定的整体向上或向下以一个给定的加速度上下垂直运动。

图 1: 同心鼓侧面简图

1.3 需解决的问题

本文将该题目的问题分为三个部分，并试图建立相关的数学模型进行研究。

问题一：理想状态下的团队协作策略。

该问题中，假设每个人可以精确控制用力的大小、方向及发力的时间。讨论使得连续颠球次数尽可能多的合作策略，并给出颠球高度。

问题二与问题三：发力时机和力度存在误差的实际情况。

该问题需要根据队员发力时机和大小的数据，建立数学模型确立每一种情况下鼓面的倾斜角度，并讨论现实情境中问题一中策略调整的问题。

问题四：倾斜情况下的调整方案。

该问题需要根据所给数据给出在球跳动方向不在垂直情况下的拉绳策略，包括发力时机及力度，并分析其效果。

2 问题分析

本问题主要研究同心鼓游戏中调整发力使得颠球次数最多的协作方案。首先研究理想状况下的最佳发力策略；接着考虑现实中发力时机和大小不能保证完全精准控制的情况，利用数学模型分析其影响并提出改进方案；最后针对一种倾斜的情况制定发力的调整方案及评估实施效果。

2.1 问题一

通过分析问题一，在使得颠球次数尽可能多的目标下，可以建立理想状况下的运动学模型。首先通过运动学的分析，得出能够使得游戏持续进行的小球与鼓的运动状态。接着根据鼓的运动特征确定团队成员的发力时机、大小等。最后通过计算得出该策略下的颠球高度。

2.2 问题二

考虑到题目给出的数据，每个人的发力时间为 0 时刻或者是 -0.1 时刻，时间间隔即为 0.1 秒，同时题目要求我们求 0.1 时刻的鼓的偏角情况，因此很自然地将时间间隔定为 0.1 秒来分析同心鼓的运动情况。这里使用的主要数学模型是力矩使同心鼓产生相应方向的角加速度，通过角加速度、角速度的合成得出 0.1 时刻的偏移情况。

2.3 问题三

由第二问进行联想，根据第二问的结果可以发现多位参与者进行合适的时机和发力大小进行发力可以做到让鼓面的倾斜角度小于本来只有某一个参与者发力失误时的倾斜角度。以此为出发点，进行控制变量法数据计算分析，得出互相发力之间的“修正”作用，以此得出在出现偏差时候的优化策略来修正我们在第一问时理想状况下给出的策略。

2.4 问题四

对于问题四，首先确定出在球有一定的偏斜时，能够使得球的轨迹恢复垂直的鼓的摆放位置。接着分析调整过程中鼓的整体方位的变化，并根据数学模型得到此过程中队员发力时机及力度，最后对于模型应用的实际效果进行分析。

3 基本假设

3.1 模型假设

- 球与鼓面的撞击为弹性碰撞，碰撞时系统没有能量损失。
- 在全过程中，不计空气阻力和一切摩擦的影响。

- 我们认为人提供的力都为沿绳当前方向的力，因此鼓与绳接触处鼓也受这个方向和大小的力。
- 第二问中，考虑到题目设定某些时刻内所有参与者施加力大小为 0，我们认为此时参与者是有施加与同心鼓重力平衡而使同心鼓处于静止平衡状态的力的，所以第二问题目数据的 $0, 90, 80$ 都是为了使同心鼓运动而多施加的力，故计算中未考虑重力。

3.2 符号说明

符号	符号说明	单位
m_1	鼓的质量	kg
m_2	球的质量	kg
d	鼓面直径	m
b	鼓面厚度	m
L	绳长	m
v_1	撞击之前鼓的速度	m/s
v_2	撞击之前球的速度	m/s
v_1'	撞击之后鼓的速度	m/s
v_2'	撞击之后球的速度	m/s
h_1	鼓下落最低点和撞击点的垂直距离	m
h_2	球上升最高点和撞击点的垂直距离	m
g	重力加速度	m/s^2
a	撞击后鼓运动的加速度	m/s^2

表 1: 记号

4 模型建立与求解

4.1 问题一的求解

问题一中，我们假设绳子与鼓面的夹角保持不变，同时假设碰撞均为弹性碰撞，则在碰撞的过程中没有能量损失。我们希望能够使得连续颠球次数尽可能多。因此，可以设定一个碰撞点，使得每次碰撞发生在同一个位置，同时每次撞击中球和鼓的速度均相同。队员需要在每个周期中控制力度和时间，使得鼓能够在每次撞击前达到相同的位置并具有相同的速度，从而达到周期性的平稳状态。

根据动量守恒定律，设垂直向上为正方向：

$$m_1 v_1 + m_2 v_2 = m_1 v_1' + m_2 v_2'$$

其中，二者的质量比为 $m_1 : m_2 = 3.6 : 0.27 = 40 : 3$，由于每次的撞击点在同一个位置，球在空中运动时没有损失，因此有

$$v_2' = -v_2$$

即在碰撞之后，球的运动速度大小与碰撞前相同，方向相反。代入动能守恒公式：

$$\frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 = \frac{1}{2} m_1 {v_1'}^2 + \frac{1}{2} m_2 {v_2'}^2$$

同时参考动量守恒，可知必有：

$$v_1' = -v_1$$

易知此时系统动量和必为 0。且由能量守恒公式 $\frac{1}{2}mv_2^2 = mgh_2$ 及题目数据要求有：

$$h_2 = \frac{v_2^2}{2g}$$

由于需要确保球和鼓在同一时间回到撞击地点，因此假定每个人能控制自己的力度和发力时间，使得鼓能够以恒定的加速度运动，并且满足

$$\frac{m_2}{m_1} = \frac{v_1}{v_2} = \frac{a}{g}$$

可以解得：$a = \frac{3}{40}g$。根据运动的对称性，在鼓运动到最低点的时候，球正好达到最高点，根据题目要求，需要满足条件:$h_1 + h_2 \geq 0.4$。由于开始时鼓面距离球正好为 40cm，假设队员可以控制使得鼓每一次回落的最低点与开始时位置相同，而球上升最高点高于初始位置。因此在第一次到达击球点之前，队员应该让鼓达到一个更大比平稳周期中 v_1 更大的速度 v_{10}，使得球和鼓在运动过程中最大间隔距离大于 40cm。之后只需要保证鼓在撞击时的速度达到 v_1 即可。在用力的同时需要保证每位队员的发力时间和力度大小均相同以保证鼓面水平，从而使得每次球能垂直弹起落下。

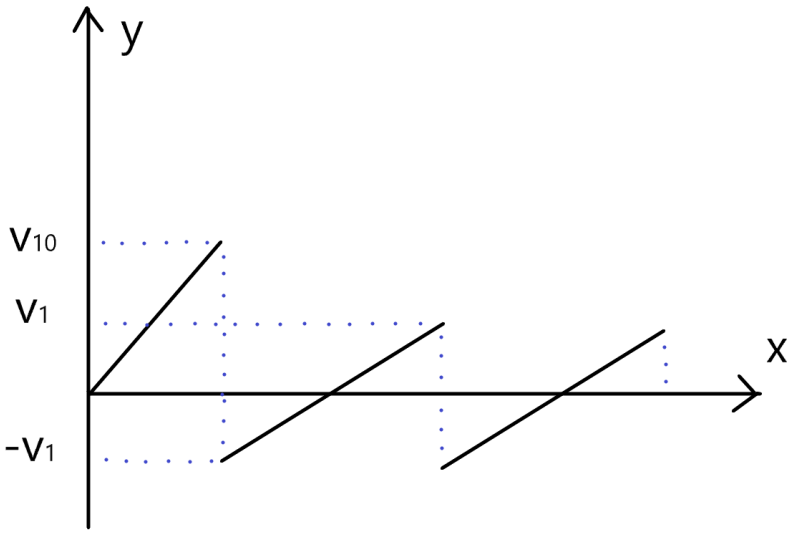


图 2: 鼓的运动状态

根据运动学公式，有：

$$v_1^2 = 2ah_1$$

$$v_2^2 = 2gh_2$$

因此可得：$\frac{h_1}{h_2} = 3 : 40$，从而颠球高度为:$H = h_1 + h_2 = \frac{43}{3}h_1$。据此可知，若要保证颠球高度大于 40cm，则第一次撞击点的设置需要满足 $h_1 > \frac{120}{43}$, 即鼓面至少要上升 $\frac{120}{43}cm$ 再与球进行碰撞。

4.2 问题二的求解

题目已经给出了此时长绳和同心鼓侧面的角度，相当于给出了沿绳方向的力在同心鼓侧面的垂直和水平方向的受力分解情况。根据受力分析，如果将绳上沿绳方向的力设为 F，应有如下的受力分解关系：

$$F_{\text{垂直}} = \frac{11}{170} \times F, \qquad F_{\text{水平}} = \frac{169.6}{170} \times F$$

在这里我们由于水平方向的力会通过转动中心，因此只会产生平移作用，故我们在这一问中不考虑水平方向的力，而是只考虑第 i 名参与者沿绳方向施加的力的垂直方向分力 F_i。

4.2.1 鼓模型的简化及转动惯量的计算

根据文章之前的分析，同心鼓为空心结构，且上下两面为牛皮面材质，质量很小，因此可以简化为一个空心并且上下不封口的圆柱形结构，或者说是一个圆环沿垂直方向平移后的结构，其质量主要分布在圆柱体的一圈侧表面上。

对于一个圆环，如图 3所示，如果转动轴定为穿过圆心的直径所在直线，那么可以很方便地通过垂直圆心转动轴时转动惯量为 $J = m \times R^2$，再由垂直轴定理分解，得到此时绕穿过圆心直径方向转动轴的转动惯量为 $J = \frac{1}{2} \times m \times R^2$。

之后，假如将该轴沿着垂直于转轴方向平移一个距离 s，如图 3右半部分所示，由于原本的转动轴是通过圆环质心的，所以可以应用转动惯量的平行轴定理。那么根据转动惯量的平行轴定理，此时一个圆环绕这样一个转动轴的转动惯量为

$$J_0 = \frac{1}{2} \times m_0 \times R^2 + m_0 \times s^2$$

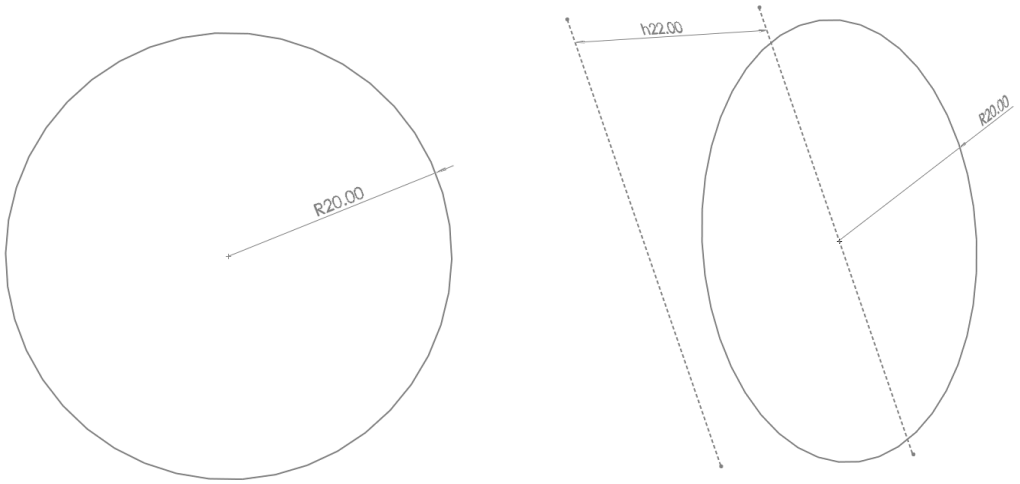


图 3: 圆环的转动惯量及转动轴平移后的情况

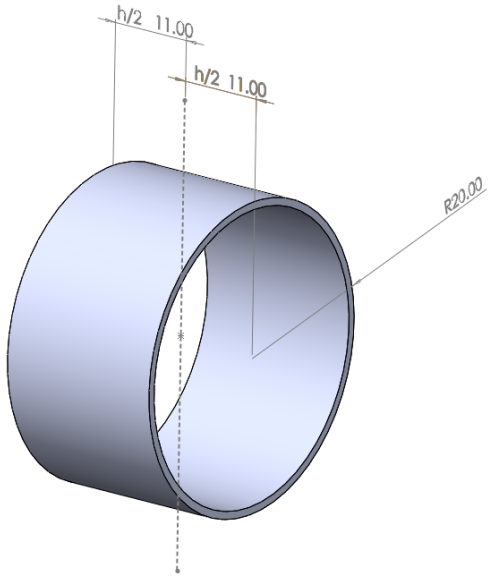


图 4: 简化同心鼓模型转动惯量计算

当考虑此时我们简化的上下不封口空心圆柱体模型时，如图 4所示，由于同心鼓受力点都在高度一半位置的平面上，因此将转动轴定为此平面上一条通过圆心的直线。之后整个同心鼓可以在高度方向上看作许多个圆环的叠加，每个圆环的质量为 $\frac{m}{h}$，把中间高度定义为 0，即是从 $-\frac{h}{2}$ 向 $\frac{h}{2}$ 积分，因此整体的转动惯量为

$$J = \int J_0 \, ds = \int_{-\frac{h}{2}}^{\frac{h}{2}} \left(\frac{1}{2} \times \frac{m}{h} \times R^2 + \frac{m}{h} \times s^2 \right) ds$$

进行积分后可以得出，在中间高度平面上，同心鼓对于转动轴的转动惯量为

$$J = \frac{1}{2} \times m \times R^2 + \frac{1}{12} \times m \times h^2$$

4.2.2 转动过程分解及偏转角的计算

为了便于分析，可以建立三维坐标系 $Oxyz$，并将鼓抽象为圆柱置于 xOy 平面上，中心位于坐标系原点。现在考虑鼓的转动，由于其受力状态有两个阶段，故要进行分阶段的分析。取每个阶段时间为 $t = 0.1s$，在每个阶段内，计算每个人对于原点的力矩 $\tau_i = \vec{r_i} \times \vec{F_i}$，其中 $\vec{r_i}$ 为矢径。由外积运算规律可知，τ_i 的方向必在 xOy 平面上，且垂直于矢径 $\vec{r_i}$。

将每个人产生的力矩累加可以得到和力矩为 $\tau = \sum_i^8 \tau_i$

根据得到力矩后通过刚体定轴转动定律可以得到相应的角加速度:

$$\tau = J \cdot \vec{\alpha}$$

其中 $\vec{\alpha}$ 的方向由右手定则确定。设每个阶段的初始角速度为 $\vec{\omega}$，其方向垂直于转动方向。由于转动时间较短，角位移较小，可以近似使用矢量运算公式：

$$\vec{\theta} = \vec{\omega} t + \frac{1}{2} \vec{\alpha} t^2$$

来计算出各个阶段内的角位移。

而由于每个阶段的角位移较大，不能将其按矢量计算出整体旋转角，故此处采取对法向量进行分阶段旋转的方式来求解整体的旋转角。根据坐标系，可取初始状态鼓的法向量 $\vec{n} = (0, 0, 1)$。若一个阶段的

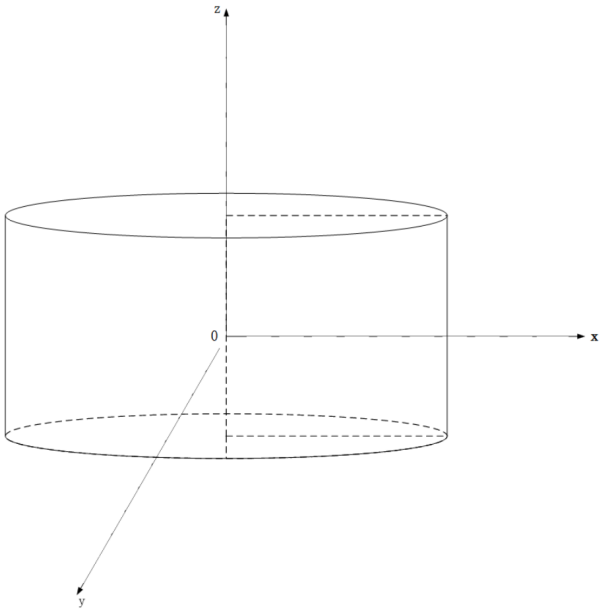


图 5: 对鼓建立坐标系

角位移为 $\vec{\theta}$，则根据 Rodrigues 旋转公式，可得此阶段后的鼓的法向量为:

$$\vec{n'} = \vec{n} \cdot cos|\vec{\theta}| + \frac{1}{|\vec{\theta}|^2}(1 - cos|\vec{\theta}|)(\vec{n} \cdot \vec{\theta})\vec{\theta} + \frac{sin|\vec{\theta}|}{|\vec{\theta}|}(\vec{\theta} \times \vec{n})$$

经过两个阶段的旋转可以得到最终的鼓的法向量 n_f，此时使用夹角公式:

$$\beta = arccos(\vec{n} \cdot \vec{n_f})$$

便可到最终的转动倾角 β。

4.2.3 最终结果汇总

根据上面建立的模型代入具体数据进行计算，得到的鼓面倾角在不同情况下的结果如表 2中所示。

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	0.4285
	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	0.7918
	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	0.3280
	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	10.2840
	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	19.0023
	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	7.7810
	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	11.5695
	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	8.1062
	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	7.5431
	用力大小	90	80	80	90	80	80	80	80	

表 2: 各种情况下倾角记录

观察上面表中的结果，可以发现当力大小不一样时，两个不一样的力的施加者相距的远近；当施力时机不一样时，提前发力的两个参与者相距的远近；以及分别不同参与者发力大小和发力时机不一样时，都会有不同的效果，相互对比可以发现，不同的“偏差”情况可能会对鼓面的倾角产生抵消或增强作用，可以作为第三问的一个解题思路。

4.3 问题三的求解

通过观察分析问题二的九组数据结果，可以从中归纳出现实情况中可能出现的四种主要误差情况：

- 同时在规定时刻发力，但有参与者力度大小不同
- 所有参与者发力大小一致，但是有参与者提前发力
- 某位参与者既发力大小没有控制好，同时发力时机不对
- 既有发力大小未控制好的参与者，同时也有其他参与者发力时机未控制好

在问题一中，我们的最优策略是基于参与者可以按照事先确定好的发力大小和时机来发力的前提下确定的，因此结合第二问的情景我们发现，应该针对上面的四种主要误差情况，给出我们的优化策略方法。

在这一问中，参照第二问的情景，我们的优化策略方法是基于 8 位参与者的情况下给出的。

4.3.1 同时发力但发力大小不一致的策略优化处理

对于所有参与者都可以在规定时刻发力，但是有些参与者无法准确施加需要大小的力的情况，为了观察普适性的结果，我们首先设定任意两位参与者发力大小不是规定大小的力 ($80N$)。假若将 8 位参与者按照同心鼓一圈顺时针标号为 $1 \sim 8$ 号，要观察两位不同发力参与者的情况，我们只要选择 1 2 1 3 1 4 1 5

这四种组合即可，其他情况都与这四种组合中某种一致。这里将 1 号参与者施加的力恒定设为 $90N$，另一位参与者施加的力在 $60 \sim 120N$ 中变化，得到各种情况下同心鼓的角度偏移结果如图 6。

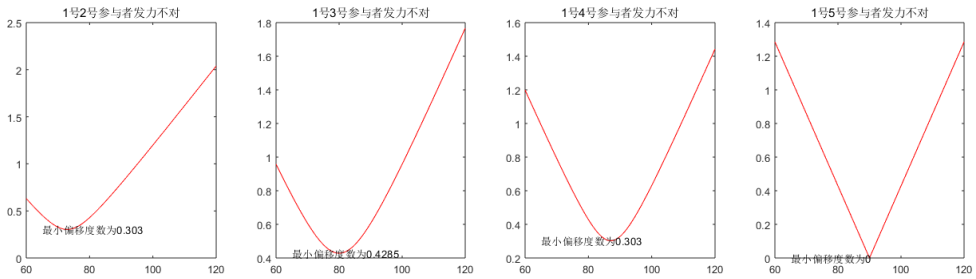


图 6: 两位参与者发力不对造成鼓面倾角变化

可以很明显地从上面的图中得出如下结论：

- 两位参与者间连线经过同心鼓圆心时，即使一人发力不是设定好的发力大小，另一人也可以通过施加相同大小的力来抵消这个影响。
- 两位参与者与鼓心连线垂直时，还是可以通过控制力度大小来将鼓偏移误差降到很低。
- 另外两种情况下，无论怎样控制用力都会使鼓面产生比较大影响的偏转。
- 相比于只有 1 号参与者发力的情况，即问题 2 中的第一种情况，1 号 2 号、1 号 4 号、1 号 5 号都可以起到修正的作用。

虽然将两两互相影响的发力情况归结为以上四种，但是我们可以发现一个问题: 只有 1 号和 5 号，即两个参与者处于正对面的方向时，可能会将一个人失误发力的结果降低为 0，这一点其实比较好理解，因为两个人的合力矩互相抵消了，但是之所以可能互相抵消，就是因为两人一开始的方向导致他们分别的力矩方向一定是共线的。

因此很自然产生这样的想法：假如以上面 1 号和 2 号的情况为例，让跟 2 号对称处于 1 号另一侧的 8 号参与者也同时施加和 2 号一样大小的力。从物理角度分析，这样一来 2 号和 8 号参与者的力矩互相抵消，产生的合力和 1 号参与者的力的连线经过圆心，应该会对偏角有着更强烈的修正作用。

上述想法实际计算后结果如图 7所示。

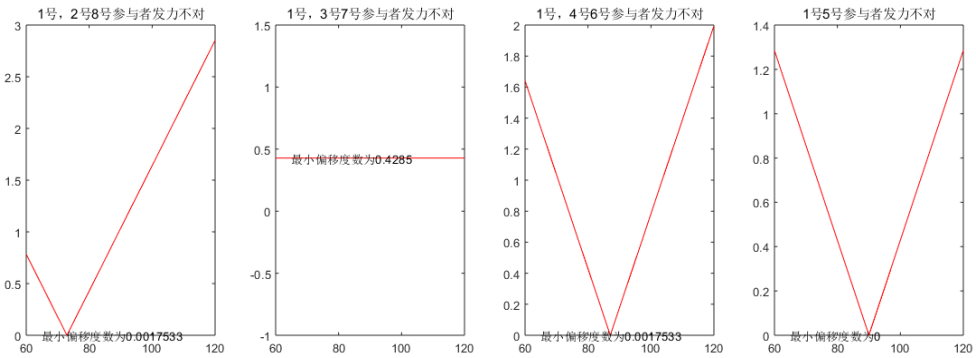


图 7: 用对称的两位参与者修正 1 号参与者的结果

4.3.2 一名参与者提前发力的情况下策略优化处理

现在考虑第二种可能出现的偏差情况：在所有参与者都可以保证准确发力，即 $80N$ 的力的情况下，1 号参与者提前 0.1 秒用力的情况下，我们像上一部分一样，考虑 2 号、3 号、4 号、5 号分别在 0 时刻时

施加什么样的力可以达到修正的作用。之后像上一部分一样，考虑对称的两个参与者一起发力能否有更好的修正作用。

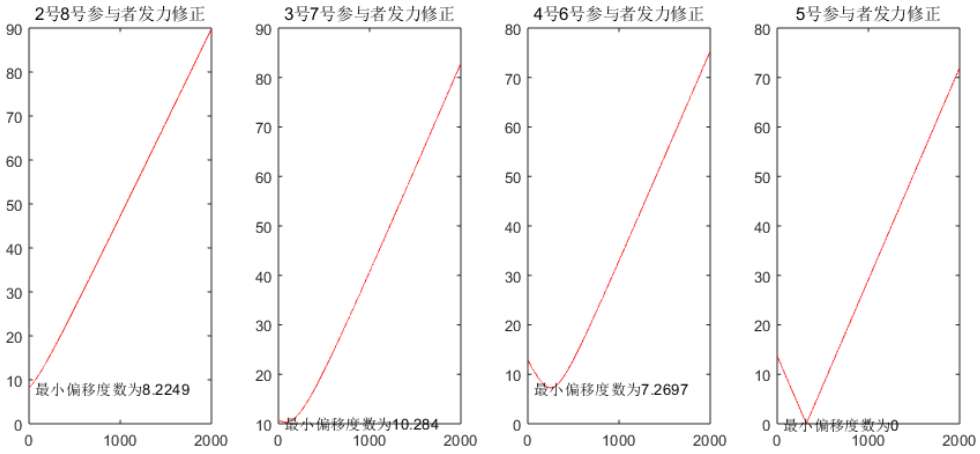


图 8: 提前发力后的修正作用

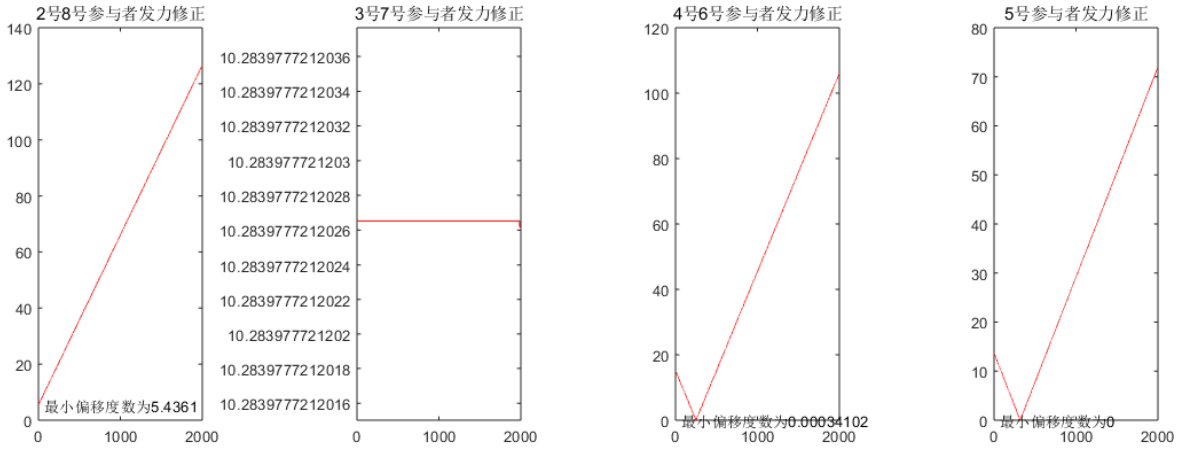


图 9: 对称两名参与者对提前发力的修正作用

首先可以发现，想在延迟 0.1 秒后由其他参与者做出动作来修正 1 号参与者提前发力造成的结果，即便可以做到，也会比两人同时发力要难得多。分析只有除 1 号外一名参与者发力修正的情况，令人吃惊的是，即便存在这 0.1 秒的误差，仍然可以通过在同心鼓对面的 5 号参与者施加一定大小的力来修正影响，使鼓面保持水平。这里 1 号参与者提前 0.1 秒发力大小为 $80N$，经程序计算此时 5 号参与者在 0 时刻发力 $320N$ 既可以修正鼓面的倾斜。而且从图上数值很容易看出，此时另外三名参与者无论施加怎样的力，都不可能使偏移角度保持在一个比较小乃至可以忽略的范围内。

若是有左右两个对称的参与者一起发力修正，可以发现在对面的 5 号参与者两边的 4 号和 6 号参与者同时施加相同的力也可以将偏移角度降低到近似于 0，此时程序输出的两名参与者发力大小分别为 $249.7N$。

4.3.3 发力修正中的叠加作用

分析上面两部分的结果还可以发现，相互之间比较相似机理的发力作用是可以互相抵消的，考虑到我们的模型中将鼓面的偏移、即角位移视为可以矢量运算的矢量，也就是说将角度默认为小角度，所以在

我们关心的角度数比较小的地方，这种互相叠加抵消作用是可以预见的，但是偏移角度已经达到甚至直角或者钝角时，显然不能单纯的使用角度的叠加，不过此时我们也不再关心曲线在那个区段间的数据，所以没有太大的关系。

4.3.4 建立一对一的发力修正关系

从上面的分析中可以看出，出现差错的参与者正对面的那个参与者，是可以将角度偏差修正到 0 度的，那么目前我们就只考虑这两个参与者。将时间间隔先按照之前的设定，设为 0.1 秒。即 1 号参与者提前 0.1 秒发出一个为 F 的力，求解在 0 时刻对面的 5 号参与者应该将发力大小控制为多大的 F' 可以保证在 0.1 秒后将偏转角度控制在 0 度。

通过直接运动规律反解得出 F 与 F' 之间应该满足的关系是 $F' = 4 \times F$，通过 Matlab 程序计算得到的两者关系曲线如图 10所示，也符合这个代数关系。

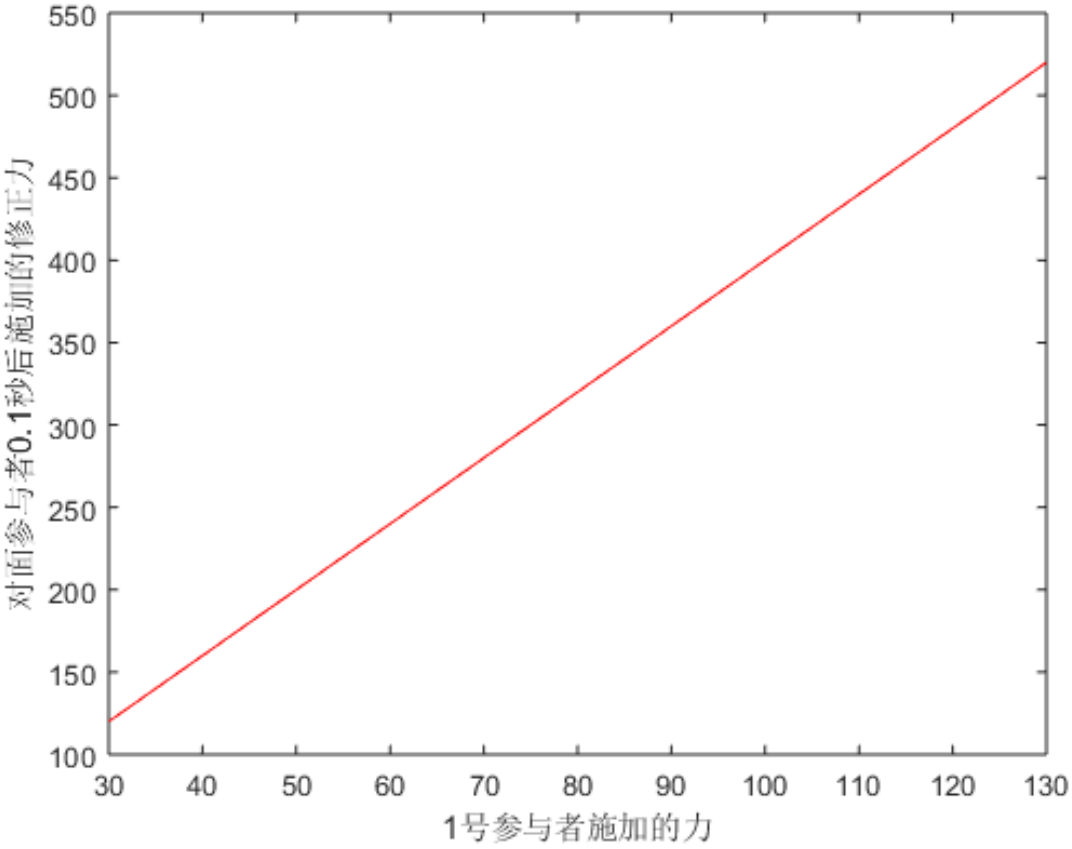


图 10: 正对面修正力的关系

4.3.5 建立二对一的发力修正关系

虽然可以通过 5 号参与者施加一个四倍于提前发力大小数值的修正力可以达到修正圆心鼓偏转的效果，不过假如在前 0.1 秒内，除了 1 号参与者之外的参与者都没有发力，只有 1 号参与者施加了 $80N$ 的力，那么为了在 0.1 秒内达到修正的效果，按照之前的理论，可以通过 4 号和 6 号参与者施加一个相同的力来达到抵消的效果。

在这里的程序实验中需要注意的是，我们需要把两个 0.1 秒内除了 1 号，4 号和 6 号外其他参与者发力大小都设为 0。前一种情况不需要考虑是因为 1 号和 5 号间相互影响，其余参与者只有发力大小一样就会两两相对抵消影响，但这里因为相对两参与者发力不再相同，所以把无关参与者发力都设置为 0。

求得在其他参与者发力大小都为 0 时，4 号和 6 号参与者应该提供的修正力和 1 号参与者错误的力的大小如图 11所示。经过拟合发现 4 号和 6 号参与者分别提供的力 F'' 与此时的 F 之间的关系约为 $F'' = 2.83 \times F$。

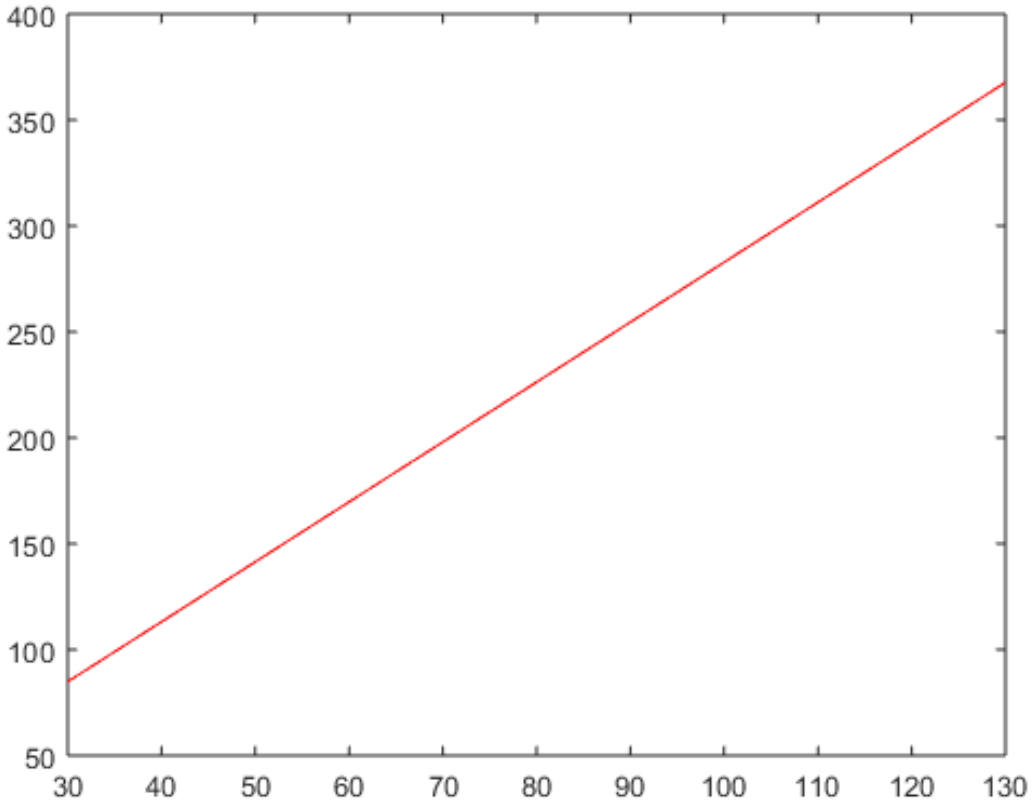


图 11: 二对一的发力修正关系

4.3.6 修正优化策略汇总

我们问题 1 中的策略是，每个参与者发力大小相同使得鼓面水平、同心鼓按照需要的运动规律上升。根据人的反应时间为 0.1 秒，因此基于问题 2 和之前的优化方法探究，我们认为假如需要所有参与者同时施加 F_0 的力，那么如果 1 号参与者错误使用了大小为 F 的力，那么他会保持这个大小的力 0.2 秒，其中前 0.1 秒是反应时间，后 0.1 秒内对面的参与者和此参与者两侧的三名参与者会施加不同于 F_0 的力来使 0.2 秒的时间周期结束后，鼓恢复到水平无倾斜且没有转动速度。

根据一对一的发力修正关系，如果只考虑对面的 5 号参与者，他应该在 0.1 秒中提供的力为

$$F_5 = 4F$$

根据二对一的发力修正关系，如果只考虑对面的 4 号和 6 号参与者，他们应该在 0.1 秒中提供的力为

$$F_4 = F_6 = 2.83\,(F - F_0)$$

为了使单个参与者提供的力不会过大，考虑到前文中的叠加性质，当只有一个参与者，即 1 号参与者出错时，应该改变 4 号，5 号和 6 号参与者的发力情况，具体发力不同如下：

$$F_5 = 2F, \quad F_4 = F_6 = 2.83\left(\frac{1}{2}F - F_0\right)$$

当有多个参与者出现差错时，分别要求每个出现差错的参与者对面的三个参与者施加这样的力，之后再在同一个参与者上遵循叠加原则进行叠加即可。

4.3.7 修正时间长短的影响

上面的优化策略是基于在 0.1 秒内将转动偏角修正为 0 度的考虑下，实际中我们还可以将这个时间间隔调整得更大一些。下面图中左侧为误差后的修正反应时间仍为 0.1 秒但花费 0.2 秒时间来将转动偏角误差降低为 0；右侧为误差后的修正反应时间和用于修正的时间都为 0.2 秒。两张图都是 1 号参与者发力错误的发力大小和 5 号参与者的修正发力的发力大小之间的对应关系。

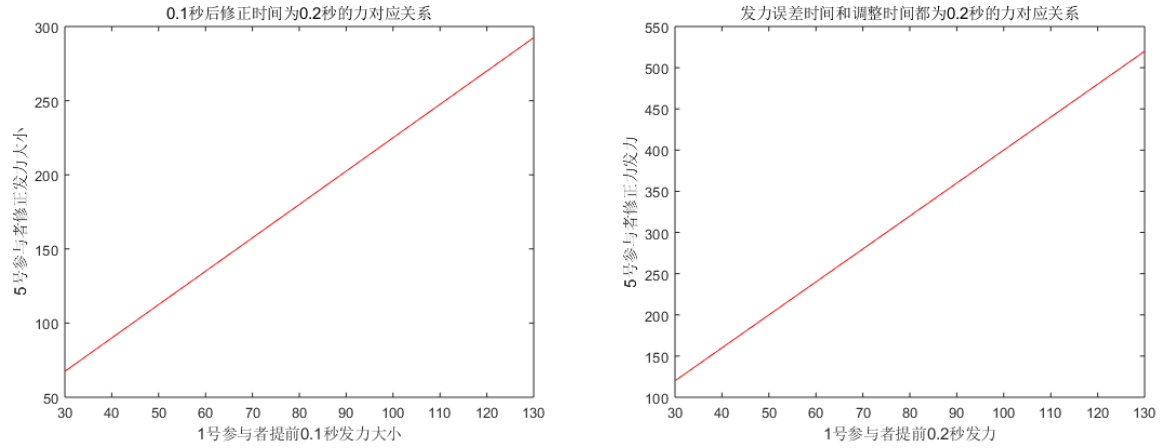


图 12: 不同的误差时间和修正时间

很明显可以看出，如果只是同时加大误差发生后的反应时间和修正时间，力的对应关系仍与原来相同。但是如果将力的修正时间拉长，倍率关系确实会发生明显的下降，因此可以在修正优化策略时考虑在修正时间不至于长到影响结果的情况下尽可能增加修正时间来使修正的发力者可以施加比较小的力来达到修正的结果。

4.4 问题四的求解

和问题二的考虑相同，由于鼓面水平方向的只会产生平移作用，故只考虑第 i 名参与者沿绳方向施加的力的垂直方向分力 F_i。

4.4.1 调整时鼓面倾斜度分析

这一问中，一开始由于鼓面倾斜而对颠球产生了一定的偏差：球的反弹相对竖直高度产生 1 度的偏离。因此，若想通过调整鼓面使得球重新竖直，需要队员配合将鼓面调整为另一个恰当的倾斜度。根据前两问的结论，在调整的时候，对于最终鼓面倾斜的程度影响较大的是发力时间，而在力的大小相差不是特别悬殊的时候，同时发力而力的大小稍有不同对于最终结果的影响较小。并且从容易控制的角度出发，我们假设所有成员发力时间均相同而力度有所不同，基于此建立相关数学模型。

如图，球在距离鼓面竖直距离为 0.6m（由鼓面倾斜造成的高度偏差忽略不计），球在竖直方向上偏差角度 α 与鼓面偏斜角度 β 的关系为：

$$\alpha = 2\beta$$

因此，若想要在球第二次下落时将其运动方向调整为垂直向上，则鼓面应该调整的角度为 $2\beta = 1$ 度。

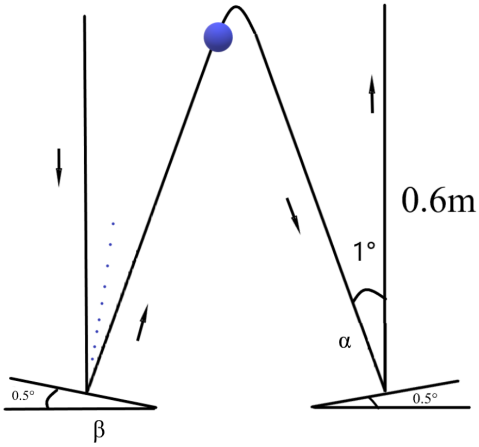


图 13: 使得颠球方向竖直的调整方案

十个人参与的情况下，所有队员从球一开始下落时便同时发出恒定的拉力，这里只考虑这些力垂直于鼓面的分量，根据力矩的合成原理，我们希望这些恒力最后的作用效果能够使得鼓面与球的下落方向垂直。

4.4.2 空间直角坐标系的建立

与问题二相同，我们以鼓的中心为原点建立空间直角坐标系，其中鼓面和 xOy 平面平行，本题中我们需要考略十个人的情况，此时鼓面已经产生偏斜，球在空中偏斜方向的投影指向两个队员之间，夹角比为 1:2。

如图，在坐标系中 10 位队员的站立方位如图所示，10 位队员均匀分布在一个圆周上，间隔 36 度。方便起见，我们设这两名队员为 1 号和 2 号，其中投影线与 1 号队员夹角为 12 度，与 2 号队员夹角为 24 度，球在空中偏斜方向的投影与 y 轴共线。

鼓的半径为 0.2m，鼓外围在 xOy 平面的方程为：

$$x^2 + y^2 = 0.04$$

考虑鼓实际收到的十个竖直向上的合力，我们希望寻找一组恰当的取值，使得其根据力系合成规则进行合成后最后剩下一个沿 x 轴正方向的力矩，使得鼓面能够以 x 轴为旋转轴进行旋转，从而进行鼓面倾斜度的调节。

为了能够使得球与鼓面撞击的时候鼓刚好有合适的倾斜度，首先计算球下落时间：

$$t = \sqrt{\frac{2h}{g}}$$

其中 $h = 0.6m$，取 $g = 9.8m/s^2$，在 t 时刻之内，鼓需要达到一个合适的位置。具体为：围绕 x 轴旋转 $\theta = 2\beta = \pi/180$。假设发力前鼓是稳定状态，角速度为 0，设角加速度为 $\vec{\alpha}$，则有：

$$\theta = \frac{1}{2}\vec{\alpha}t^2$$

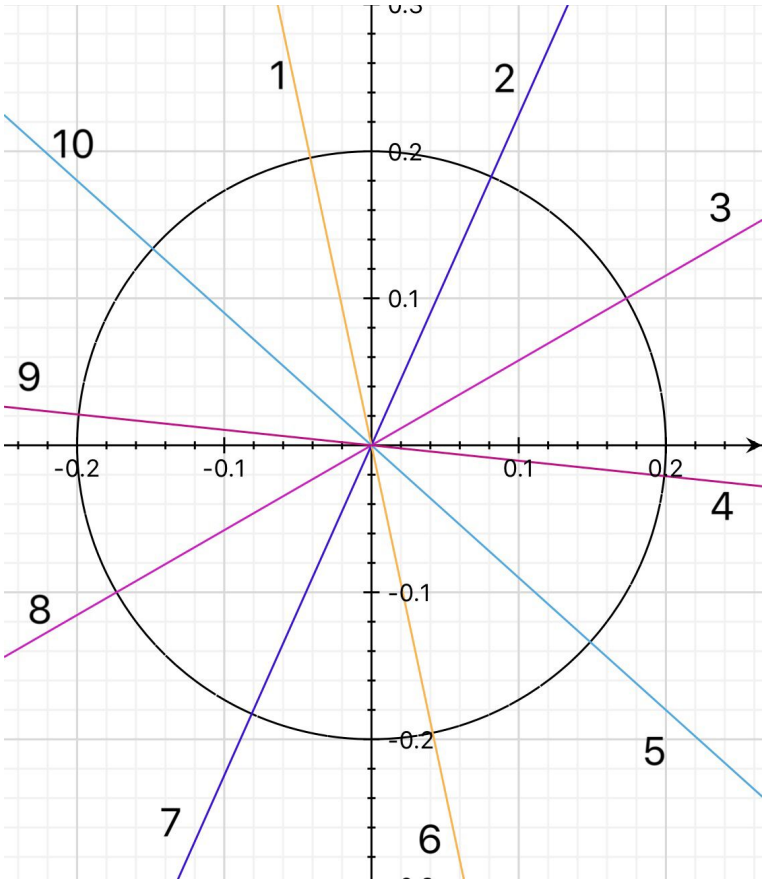


图 14: 队员站立方位在 xOy 平面上的表示

在空间直角坐标系中，要保证鼓仅围绕 x 轴发生正向转动，则在力系合成后关于原点 O 的力矩可以表示为 $\tau = (x, 0, 0)$（$x > 0$）。根据第二问同心鼓沿轴转动惯量的结论 $J = \frac{1}{2}mR^2 + \frac{1}{12}mh^2$ 及 $\vec{\alpha} = \frac{\tau}{J}$ 可得，若使得球与鼓面碰撞时角度合适，则有：

$$x = 0.1442$$

4.4.3 力系合成

考虑鼓在竖直方向上的受到绳子的力为 $F_1, F_2, ..., F_{10}$，每个人克服鼓的重力后合力的大小为 $f_1, f_2, ..., f_{10}$，根据受力分析：

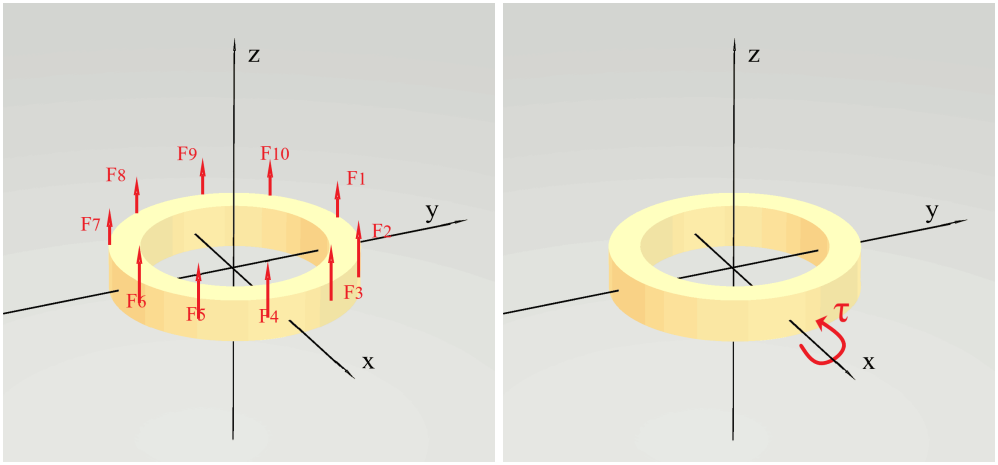
$$\sum_{i=1}^{10} F_i - mg = \sum_{i=1}^{10} f_i$$

假设开始调整前鼓处于静止状态，即 $\sum_{i=1}^{10} F_i = mg$，$m = 3.6kg$ 为鼓的质量而 g 为重力加速度，合力大小 f_i 在数值上等于队员 i 调整时需要增加的力的大小（竖直方向）。根据力系分解原则，有：

$$\tau = \vec{r_1} \times f_1 + \vec{r_2} \times f_2 + ... + \vec{r_{10}} \times f_{10}$$

将合力与方向向量代入上述公式中计算（如在此坐标系中 $\vec{r_3} = (0.1732, 0.1000, 0)$, $\vec{f_3} = (0, 0, f_3)$），我们希望寻找一组力度 $\vec{F} = (f_1, ..., f_{10})$，使得最终力系合成的结果 τ 尽量接近于 $(x, y, z) = (0.1442, 0, 0)$。

忽略队员发力对鼓在竖直方向上的位移影响，同时根据题目精确控制发力大小的条件，不妨限制所有垂直方向上合力 f_i 的大小在 1N 以内。设任意一组力合成的结果为 (x_t, y_t, z_t)，定义误差为：



(a) 真实受力情况　　(b) 期望的力系合成结果

图 15: 鼓的受力分析

$$loss = (x - x_t)^2 + (y - y_t)^2 + (z - z_t)^2$$

通过随机搜索的方式找到能够使得与 $\tau = (0.1442, 0, 0)$ 误差最小的解（代码如附录所示）：十位队员在竖直方向上分力的大小为

$$f_1 = 0.8416N, f_2 = 0.9589N, f_3 = 0.0036N, f_4 = 0.5526N, f_5 = 0.8132N$$

$$f_6 = 0.0321N, f_7 = 0.6068N, f_8 = 0.4849N, f_9 = 0.2413N, f_{10} = 0.6350N$$

在这种力度下力系合成的结果为：$\tau_0 = (0.144208, -0.000023, 0.000000)$。因此，按照这种方式出力，能够使得球再次弹起时其垂直方向运动轨迹无偏斜。

4.4.4 调整情形的实施效果

使得 O 点合力矩 τ 最终的结果接近 $(0.1442, 0, 0)$ 的力的组合并不唯一，限制误差在一定的范围内（这里设置 $loss < 5 \times 10^{-8}$），在 10^9 次随机搜索中，算法找到了几十组满足要求的解（见附表）。

现实中，同心鼓的诀窍通常都是大家尽可能同时发力，然而当球产生偏斜，想要进行调整从而保证球能够竖直回弹，难以做到十个人按照精确的方式发力以达到恰当好处的角度。通常是偏斜一方的几个人进行发力，而其他人保持恒力不变动，这种方式将会更为可行，并且容易操控。

比如在本例中，当球偏斜的投影指向在 1 号和 2 号成员之间时，可以让其他成员维持原有状态（即刚好克服重力而合力为零），1 号和 2 号成员进行调整（即 $f_1 > 0$，$f_2 > 0$）。此时可以直接解得一组解：

$$f_1 = 0.498N, f_2 = 0.255N, f_i = 0, i = 3, 4, ..., 10$$

同样，只需要两个人进行发力以达到所需效果还有其他方案，具体如下表：

调整者	力度大小
$1,2$	$f_1 = 0.498N, f_2 = 0.255N$
$1,3$	$f_1 = 0.656N, f_3 = 0.157N$
$2,9$	$f_2 = 0.753N, f_9 = 0.308N$
$2,10$	$f_2 = 0.563N, f_{10} = 0.308N$
$3,10$	$f_3 = 0.563N, f_{10} = 0.656N$

表 3: 双人发力的调整方案

另外也可以让两个以上的成员同时发力，并且通过计算可以发现，力量调整涉及到两个人以上时，其调整方案并不唯一。

现实中，两个人或多个人同时调整要比所有人调整更为可行。实际游戏往往采用这样的策略：看到球倾斜时，倾斜投影方向两侧的两个队员进行调整，或者是倾斜方向一半的人员进行力量调整，从而使得鼓面到达合适的位置，以达到让球垂直方向运动回到竖直的效果。

5 模型评价

5.1 第一题运动学模型评价

第一题针对理想情况，我们假定所有人能够同时发力并且力度相同，从而将问题转化为一个运动学的问题进行求解，通过人为设定撞击位点，可以计算出相应的高度。在这种情形下，只要给鼓在垂直方向上有一个足够的距离进行加速，即可达到颠球所需的效果。

当然，现实中这种理想的模型并不存在，除了空气阻力、碰撞时的能量损失等因素外，大家力度控制难免会有偏差，这个时候问题就不能简化为一个简单的运动学模型。在颠球效果出现偏差时，需要及时根据偏差进行调整。

5.2 二三题中受力转动模型评价

在第二题中，将同心鼓简化为一个中空、上下不封口的圆柱形结构来计算质量分布以及转动惯量。这一步是有必要且有实际条件支撑的。因为需要计算转动惯量，所以我们需要一个简化的质量分布模型，同时经过实际考察，确定了同心鼓确实符合我们模型中建立的质量分布，内部中空的结构和上下两面的质量极低的牛皮鼓面都是为了增加鼓的弹性，才能正常地进行如“同心协力”游戏这样的活动。事实上，正是因为同心鼓有着这样的质量分布特征，才会很容易因为边上一圈参与者给的力不平衡而导致转动。

之后，我们的模型最大的特点就是将角度的位移确定为矢量进行计算。在物理学中我们知道，角度是不能一概当作矢量进行叠加的，最明显的就是它们的相加不满足交换律。这里我们由力矩引起的角加速度和角速度是分解为坐标系三个坐标方向的。在前一时间段内的角加速度引起的角速度和后一时间段另一个角加速度的方向显然是不共线的。这里如果是速度位移的速度和加速度，不同方向 (存在夹角) 是可以分别计算再矢量求和的。这里我们的模型将角度也视为这样的矢量，因为物理学中其实这样的矢量叠加在角度偏小时候是完全可以成立的，很多理论力学中的定理也由此推出，因此这样的模型建立是有依据并且可行的。

但是这样的叠加总会有一定的误差存在，因此如果可以将原本的时间间隔 0.1 秒拆分成更小的时间段进行迭代计算，可以将误差控制在更小的范围之内。

5.3 第四题中的力系合成

对于一个已经产生的偏差，需要进行调整，从现实的角度来说，由于不同发力时间比力度不同对于倾斜的结果更为敏感，我们希望控制大家使用不同的力度而保证相同的发力时间进行调整。

根据力系合成的思想，我们希望最终的结果是使得鼓面在一定时间内发生一定角度的偏转。模型首先通过搜索计算出十个人同时调整时每个人的出力方案，另外讨论现实中更为可行的方案：即不需要所有人员调整而只涉及其中几位队员的力度调整，这样的模型将会更为可行。从直观上来讲，当鼓面产生倾斜时，倾斜方一侧需要使出额外的力气使得鼓获得一个角加速度，从而达到调整角度的效果，进过我们模型的验证，这种调整的方案会更为合理。

6 参考文献

参考文献

[1] 姜启源. 数学建模 [M]. 高等教育出版社, 2011-1.

[2] 哈尔滨工业大学理论力学教研室. 理论力学 [M]. 高等教育出版社, 2010-11

[3] 程稼夫. 奥林匹克物理竞赛教材：力学篇.2013.

附录

第二题使用的计算程序 rotation_final.m

```
clear all;
clc;
N = 8; % 人数
m = 3.6; % 鼓的质量
h = 0.22; % 鼓的厚度
R = 0.2; % 鼓的半径
angle = 2 * pi/N * (0:(N-1));
f = [0 0 0 0 0 0 0 0;90 80 80 80 80 80 80 80]; % 设定不同的力得到不同的结果
% 构建三维数组Force，Force的每一页为题目给定的一种发力情况
Force(:,:,1) = f;
Force(:,:,2) = [0 0 0 0 0 0 0 0;90 90 80 80 80 80 80 80];
Force(:,:,3) = [0 0 0 0 0 0 0 0;90 80 80 90 80 80 80 80];
Force(:,:,4) = [80 0 0 0 0 0 0 0;80 80 80 80 80 80 80 80];
Force(:,:,5) = [80 80 0 0 0 0 0 0;80 80 80 80 80 80 80 80];
Force(:,:,6) = [80 0 0 80 0 0 0 0;80 80 80 80 80 80 80 80];
Force(:,:,7) = [90 0 0 0 0 0 0 0;90 80 80 80 80 80 80 80];
Force(:,:,8) = [0 80 0 0 80 0 0 0;90 80 80 90 80 80 80 80];
Force(:,:,9) = [0 0 0 0 80 0 0 80;90 80 80 90 80 80 80 80];
Time_sec = [1 2];
lapse = 0.1; % 时间间隔
result = [];

for k = 1:9
    % 初始化每次计算时需要的参数
    omiga = 0; % 角速度
    J = 1/2 * m * R^2 + 1/12 * m * h^2; % 转动惯量
    theta = 0;
    Direction_vector = [0 0 1]; % 鼓面法向量
    b = [0,0,1];
    for t = Time_sec
        moment = 0; % 力矩
        if norm(Force(t,:,k)) == 0
            continue
        end
        for i = 1:N % 计算N个人的力相对于原点产生的总力矩
            radius = R * [cos(angle(i)),sin(angle(i)),0];
            F = [0,0,11/170*Force(t,i,k)];
            moment = moment + cross(radius,F);
        end
        a = moment/J; % 加速度
        rotate = 1/2 * a * lapse^2 + omiga * lapse;
        axis = rotate/norm(rotate); % 转轴
        theta = norm(rotate); % 旋转角度
        omiga = a * lapse + omiga;
        %以下用Rodrigues旋转公式计算旋转后的法向量
        Direction_vector = Direction_vector*cos(theta) + ...
            (1-cos(theta))*dot(Direction_vector,axis)*axis + ...
            sin(theta)*cross(axis,Direction_vector);
    end
    result_now = acosd(dot(Direction_vector,b));
    result = [result result_now];
```

```
50      end
51      disp(result);
```

将第二题程序简化为函数的 rotation_func.m

```
1       function result = rotation_func(F)
2       % 传入F为发力矩阵，2行N列，第一行为前0.1秒内发力，第二行为0秒时发力
3       N = length(F(1,:)); % 人数
4       m = 3.6; % 鼓的质量
5       h = 0.22; % 鼓的厚度
6       R = 0.2; % 鼓的半径
7       angle = 2 * pi/N * (0:(N-1));
8       Time_sec = [1  2];
9       lapse = 0.1; % 时间间隔
10      omiga = 0; % 角速度
11      J = 1/2 * m * R^2 + 1/12 * m * h^2; % 转动惯量
12      theta = 0;
13      Direction_vector = [0 0 1]; % 鼓面法向量
14      b = [0,0,1];
15      for t = Time_sec
16          moment = 0; % 力矩
17          if norm(F(t,:))==0
18              continue
19          end
20          for i = 1:N % 计算N个人的力相对于原点产生的总力矩
21              radius = R * [cos(angle(i)),sin(angle(i)),0];
22              f = [0,0,11/170*F(t,i)];
23              moment = moment + cross(radius,f);
24          end
25          a = moment/J; % 加速度
26          rotate = 1/2 * a * lapse^2 + omiga * lapse;
27          axis = rotate/norm(rotate); % 转轴
28          theta = norm(rotate); % 旋转角度
29          omiga = a * lapse + omiga;
30          % 以下用Rodrigues旋转公式计算旋转后的法向量
31          Direction_vector = Direction_vector*cos(theta) + ...
                (1-cos(theta))*dot(Direction_vector,axis)*axis + ...
                sin(theta)*cross(axis,Direction_vector);
32      end
33      result = acosd(dot(Direction_vector,b));
34      end
```

第三题使用的画图比对程序 draw_2_different_force.m

```
1       clear all;clc;
2       F = [0 0 0 0 0 0 0 0;
3       90 80 80 80 80 80 80 80];
4       Force_list = 60:0.1:120;
5       %%
6       %第1号和第2号参与者发力大小不当
7       F = [0 0 0 0 0 0 0 0;
```

```
    90 80 80 80 80 80 80 80];
    result_1_2 = [];
    min_1_2 = 5;
    for k = Force_list
        F(2,2)=k;
        F(2,8)=k;
        temp = rotation_func(F);
        if temp ≤ min_1_2
            min_1_2 = temp;
        end
        result_1_2 = [result_1_2 temp];
    end
    %%
    %第1号和第3号参与者发力大小不当
    F = [0 0 0 0 0 0 0 0;
    90 80 80 80 80 80 80 80];
    result_1_3 = [];
    min_1_3 = 5;
    for k = Force_list
        F(2,3)=k;
        F(2,7)=k;
        temp = rotation_func(F);
        if temp ≤ min_1_3
            min_1_3 = temp;
        end
        result_1_3 = [result_1_3 temp];
    end
    %%
    %第1号和第4号参与者发力大小不当
    F = [0 0 0 0 0 0 0 0;
    90 80 80 80 80 80 80 80];
    result_1_4 = [];
    min_1_4 = 5;
    for k = Force_list
        F(2,4)=k;
        F(2,6)=k;
        temp = rotation_func(F);
        if temp ≤ min_1_4
            min_1_4 = temp;
        end
        result_1_4 = [result_1_4 temp];
    end
    %%
    %第1号和第5号参与者发力大小不当
    F = [0 0 0 0 0 0 0 0;
    90 80 80 80 80 80 80 80];
    result_1_5 = [];
    min_1_5 = 5;
    for k = Force_list
        F(2,5)=k;
        temp = rotation_func(F);
        if temp ≤ min_1_5
            min_1_5 = temp;
        end
        result_1_5 = [result_1_5 temp];
    end
    %%
```

```
figure(1);
subplot(1,4,1);
plot(Force_list, result_1_2,'r-');
text(65,min_1_2,['最小偏移度数为',num2str(min_1_2)]);
title('1号，2号8号参与者发力不对')
subplot(1,4,2);
plot(Force_list, result_1_3,'r-');
text(65,min_1_3,['最小偏移度数为',num2str(min_1_3)]);
title('1号，3号7号参与者发力不对')
subplot(1,4,3);
plot(Force_list, result_1_4,'r-');
text(65,min_1_4,['最小偏移度数为',num2str(min_1_4)]);
title('1号，4号6号参与者发力不对')
subplot(1,4,4);
plot(Force_list, result_1_5,'r-');
text(65,min_1_5,['最小偏移度数为',num2str(min_1_5)]);
title('1号5号参与者发力不对')
```

第三题使用的画图对比延迟效果的 delay_correction.m

```
clear all;clc;
F = [80 0 0 0 0 0 0 0;
80 80 80 80 80 80 80 80];
Force_list = 0:0.1:2000;
%%
%用2号参与者发力修正结果
F = [80 0 0 0 0 0 0 0;
80 80 80 80 80 80 80 80];
result_1_2 = [];
min_1_2 = 10;
for k = Force_list
    F(2,2)=k;
    F(2,8)=k;
    temp = rotation_func(F);
    if temp ≤ min_1_2
        min_1_2 = temp;
    end
    result_1_2 = [result_1_2 temp];
end
%%
%用3号参与者发力修正结果
F = [80 0 0 0 0 0 0 0;
80 80 80 80 80 80 80 80];
result_1_3 = [];
min_1_3 = 100;
for k = Force_list
    F(2,3)=k;
    F(2,7)=k;
    temp = rotation_func(F);
    if temp ≤ min_1_3
        min_1_3 = temp;
    end
    result_1_3 = [result_1_3 temp];
end
```

```
%%
%用4号参与者发力修正结果
F = [80 0 0 0 0 0 0 0;
80 80 80 80 80 80 80 80];
result_1_4 = [];
min_1_4 = 10;
for k = Force_list
    F(2,4)=k;
    F(2,6)=k;
    temp = rotation_func(F);
    if temp ≤ 0.0004
        disp(k);
    end
    if temp ≤ min_1_4
        min_1_4 = temp;
    end
    result_1_4 = [result_1_4 temp];
end
%%
%用5号参与者发力修正结果
F = [31 0 0 0 0 0 0 0;
31 80 80 80 80 80 80 80];
result_1_5 = [];
min_1_5 = 10;
for k = Force_list
    F(2,5)=k;
    temp = rotation_func(F);
    if temp == 0
        disp(k)
    end
    if temp ≤ min_1_5
        min_1_5 = temp;
    end
    result_1_5 = [result_1_5 temp];
end
%%
figure(1);
subplot(1,4,1);
plot(Force_list, result_1_2,'r-');
text(65,min_1_2,['最小偏移度数为',num2str(min_1_2)]);
title('2号8号参与者发力修正')
subplot(1,4,2);
plot(Force_list, result_1_3,'r-');
text(65,min_1_3,['最小偏移度数为',num2str(min_1_3)]);
title('3号7号参与者发力修正')
subplot(1,4,3);
plot(Force_list, result_1_4,'r-');
text(65,min_1_4,['最小偏移度数为',num2str(min_1_4)]);
title('4号6号参与者发力修正')
subplot(1,4,4);
plot(Force_list, result_1_5,'r-');
text(65,min_1_5,['最小偏移度数为',num2str(min_1_5)]);
title('5号参与者发力修正')
```

得出一对一修正规律的 search_force.m

```
clear all;clc;
F_1 = 30:0.1:130;
F_5 = 0:0.01:2000;
correspond_force = [];
F = [80 0 0 0 0 0 0 0;
80 80 80 80 80 80 80 80];
for f1 = F_1
    for f5 = F_5
        F(1,1) = f1;
        F(2,1) = f1;
        F(2,5) = f5;
        result = rotation_func(F);
        if result ≤ 0.000001
            two_force = [f1;f5];
            correspond_force = [correspond_force two_force];
            break;
        end
    end
    disp(f1);
end
plot(correspond_force(1,:), correspond_force(2,:), 'r-');
```

得出二对一修正规律的 search_2_force.m

```
clear all;clc;
F_1 = 30:0.1:130;
F_5 = 0:0.01:2000;
correspond_2_force = [];
F = [80 0 0 0 0 0 0 0;
80 0 0 0 0 0 0 0];
% 这里将其他人的施加力都设为0来防止误差
% 因为此时如果4和6是变力，就没有人平衡2和8的力，会产生偏转
for f1 = F_1
    for f5 = F_5
        F(1,1) = f1;
        F(2,1) = f1;
        F(2,4) = f5;
        F(2,6) = f5;

        result = rotation_func(F);
        if result ≤ 0.001
            two_force = [f1;f5];
            correspond_2_force = [correspond_2_force two_force];
            break;
        end
    end
    disp(f1);
end
plot(correspond_2_force(1,:), correspond_2_force(2,:), 'r-');
```

第四题使用的计算程序 Countcro.m

```
function [output] = Countcro(F) %输入一个len=10的向量
d=0.2; %圆的半径
a=[102,66,30,354,318,282,246,210,174,138]*2*pi/360; %极坐标转换为直角坐标

%计算空间向量F与r的值
r1=[cos(a(1)),sin(a(1)),0]*d;
r2=[cos(a(2)),sin(a(2)),0]*d;
r3=[cos(a(3)),sin(a(3)),0]*d;
r4=[cos(a(4)),sin(a(4)),0]*d;
r5=[cos(a(5)),sin(a(5)),0]*d;
r6=[cos(a(6)),sin(a(6)),0]*d;
r7=[cos(a(7)),sin(a(7)),0]*d;
r8=[cos(a(8)),sin(a(8)),0]*d;
r9=[cos(a(9)),sin(a(9)),0]*d;
r10=[cos(a(10)),sin(a(10)),0]*d;

F1=[0,0,F(1)];
F2=[0,0,F(2)];
F3=[0,0,F(3)];
F4=[0,0,F(4)];
F5=[0,0,F(5)];
F6=[0,0,F(6)];
F7=[0,0,F(7)];
F8=[0,0,F(8)];
F9=[0,0,F(9)];
F10=[0,0,F(10)];

%计算力系合成的结果
M=cross(r1,F1)+cross(r2,F2)
 +cross(r3,F3)+cross(r4,F4)
 +cross(r5,F5)+cross(r6,F6)
 +cross(r7,F7)+cross(r8,F8)
 +cross(r9,F9)+cross(r10,F10);

output=M;
end
```

第四题搜索最优解程序 Gridsearch.m

```
function Gridsearch(times)
Radom=10000; %生成等距随机数
R1=unidrnd(Radom,times,1)/Radom;
R2=unidrnd(Radom,times,1)/Radom;
R3=unidrnd(Radom,times,1)/Radom;
R10=unidrnd(Radom,times,1)/Radom;
R9=unidrnd(Radom,times,1)/Radom;
R8=unidrnd(Radom,times,1)/Radom;
R7=unidrnd(Radom,times,1)/Radom;
R6=unidrnd(Radom,times,1)/Radom;
R5=unidrnd(Radom,times,1)/Radom;
R4=unidrnd(Radom,times,1)/Radom;
LOSS=100;
for i=1:times
```

```
        F=[R1(i),R2(i),R3(i),R4(i),R5(i),R6(i),R7(i),R8(i),R9(i),R10(i)];
        result=Countcro(F);
        % 寻找与期望力矩差距最小的结果
        loss=power(result(1)-0.1442,2)+power(result(2),2)+power(result(3),2);
        if loss<LOSS % 每次输出最优结果
            LOSS=loss;
            form='force=(%.4f,%.4f,%.4f,%.4f,%.4f,%.4f,%.4f,%.4f,%.4f,%.4f), ...
                vector=(%f,%f,%f), loss=%f, times=%d\n';
            fprintf(form,R1(i),R2(i),R3(i),R4(i),R5(i),R6(i),R7(i),R8(i),R9(i),R10(i),
            result(1),result(2),result(3),loss,i);
        end
    end
```

程序 Gridsearch.m 的执行结果

```
>> Gridsearch(100000000)
force=(0.5842,0.2943,0.0428,0.0039,0.8311,0.4702,0.3205,0.5344,0.8430,0.4721), vector=(-0.062147,0.205562,0.000000), loss=0.084835, times=1
force=(0.8095,0.3144,0.4056,0.3811,0.9073,0.8950,0.0366,0.9183,0.8613,0.5418), vector=(-0.056114,0.103839,0.000000), loss=0.050908, times=2
force=(0.0930,0.7492,0.5564,0.1410,0.0089,0.0293,0.2091,0.8221,0.2241,0.8929), vector=(0.204612,0.152651,0.000000), loss=0.026952, times=3
force=(0.7652,0.2263,0.9543,0.1395,0.5421,0.5208,0.4602,0.6556,0.1415,0.9625), vector=(0.091249,0.040335,0.000000), loss=0.004431, times=10
force=(0.7543,0.6758,0.8493,0.7182,0.8194,0.2450,0.6397,0.5915,0.6606,0.9853), vector=(0.153007,-0.013210,0.000000), loss=0.000252, times=16
force=(0.7407,0.1193,0.9996,0.3026,0.6621,0.4258,0.0222,0.5186,0.6439,0.8094), vector=(0.154292,0.011663,0.000000), loss=0.000238, times=667
force=(0.1084,0.4704,0.6840,0.3820,0.0546,0.3818,0.0842,0.1204,0.7310,0.5995), vector=(0.153655,0.010002,0.000000), loss=0.000189, times=721
force=(0.1965,0.7146,0.3740,0.7663,0.0595,0.1935,0.0189,0.9957,0.0190,0.7037), vector=(0.136116,-0.001681,0.000000), loss=0.000068, times=1340
force=(0.4215,0.7918,0.0888,0.4400,0.5777,0.1240,0.4485,0.1008,0.3424,0.7812), vector=(0.144917,-0.002644,0.000000), loss=0.000008, times=2341
force=(0.1548,0.9566,0.4373,0.4450,0.2943,0.3002,0.4919,0.1989,0.5076,0.7693), vector=(0.145176,-0.002090,0.000000), loss=0.000005, times=4791
force=(0.7725,0.6832,0.8023,0.6958,0.1869,0.8058,0.6831,0.2262,0.6830,0.8939), vector=(0.145461,0.001358,0.000000), loss=0.000003, times=21399
force=(0.6276,0.9196,0.2617,0.0962,0.3133,0.5651,0.5503,0.1635,0.0136,0.7250), vector=(0.142891,0.000310,0.000000), loss=0.000002, times=42002
force=(0.7508,0.7038,0.2484,0.8362,0.4591,0.6611,0.0659,0.3693,0.8609,0.6163), vector=(0.143562,0.001056,0.000000), loss=0.000002, times=46242
force=(0.5649,0.6336,0.6527,0.6754,0.1946,0.4061,0.3340,0.3467,0.9070,0.3621), vector=(0.143663,0.000192,0.000000), loss=0.000000, times=49215
force=(0.9896,0.0162,0.8525,0.4036,0.9939,0.0334,0.6372,0.0632,0.7208,0.8829), vector=(0.144305,0.000162,0.000000), loss=0.000000, times=70513
force=(0.5583,0.6191,0.4134,0.8369,0.6663,0.0763,0.3626,0.5051,0.6739,0.7832), vector=(0.144225,0.000013,0.000000), loss=0.000000, times=1102970
force=(0.8416,0.9589,0.0036,0.5526,0.8132,0.0321,0.6068,0.4849,0.2413,0.6350), vector=(0.144208,-0.000023,0.000000), loss=0.000000, times=54504553
```

图 16: 程序输出结果

搜索所有平方损失在 5×10^{-8} 以内的解

```
for i=1:times
    F=[R1(i),R2(i),R3(i),R4(i),R5(i),R6(i),R7(i),R8(i),R9(i),R10(i)];
    result=Countcro(F);
    loss=power(result(1)-0.1442,2)+power(result(2),2)+power(result(3),2);
    if loss<0.00000005
        form='%.4f & %.4f & %.4f & %.4f & %.4f & %.4f & %.4f & %.4f & %.4f & ...
            (%.4f,%.4f,%.4f) & %d \n';
        fprintf(form,R1(i),R2(i),R3(i),R4(i),R5(i),R6(i),R7(i),R8(i),R9(i),R10(i),
        result(1),result(2),result(3),i);
    end
end
```

可以达到效果的近似解

f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	τ	id
0.8865	0.8706	0.0505	0.5023	0.6427	0.6212	0.1718	0.3277	0.5377	0.5808	(0.1443,0.0000,0.0000)	5226759
0.4092	0.6124	0.7307	0.2323	0.1562	0.2464	0.4272	0.1802	0.6727	0.2637	(0.1443,-0.0001,0.0000)	5618542

0.4335	0.4952	0.1598	0.2958	0.7175	0.2711	0.0741	0.1491	0.2458	0.9821	(0.1441,0.0000,0.0000)	6068945
0.7419	0.6807	0.7418	0.6861	0.7108	0.6129	0.2120	0.5579	0.9605	0.7792	(0.1442,0.0001,0.0000)	7016748
0.5436	0.7089	0.3031	0.4272	0.6627	0.3877	0.0400	0.6129	0.2531	0.8584	(0.1443,0.0002,0.0000)	8433823
0.7845	0.8652	0.7674	0.3506	0.1249	0.3795	0.8059	0.2779	0.7326	0.1032	(0.1441,-0.0000,0.0000)	9909855
0.4644	0.8955	0.2837	0.6794	0.1460	0.2672	0.1153	0.6421	0.6464	0.1439	(0.1443,-0.0001,0.0000)	11879622
0.3655	0.4723	0.6358	0.8430	0.5431	0.1163	0.1163	0.6726	0.6989	0.8195	(0.1441,0.0002,0.0000)	14067425
0.9440	0.1841	0.2719	0.7951	0.2972	0.6475	0.0126	0.4756	0.1288	0.9619	(0.1440,-0.0001,0.0000)	19490692
0.7028	0.8115	0.7792	0.4894	0.2631	0.0383	0.6532	0.8146	0.4430	0.1855	(0.1440,0.0001,0.0000)	19811065
0.6069	0.6926	0.3682	0.4125	0.3410	0.0677	0.4986	0.5420	0.0720	0.5482	(0.1442,-0.0002,0.0000)	20093666
0.7228	0.5765	0.8909	0.0027	0.8264	0.0923	0.8845	0.1646	0.3978	0.7986	(0.1442,-0.0001,0.0000)	20338172
0.4006	0.1054	0.9087	0.0919	0.1587	0.1743	0.1066	0.1019	0.7114	0.2058	(0.1440,-0.0000,0.0000)	20700059
0.5251	0.3960	0.8892	0.3345	0.6750	0.1016	0.1756	0.4160	0.9693	0.3789	(0.1441,-0.0000,0.0000)	22232689
0.8660	0.5192	0.4839	0.1935	0.8613	0.1256	0.1026	0.8203	0.1527	0.5441	(0.1440,-0.0001,0.0000)	26151771
0.9897	0.3484	0.4229	0.8132	0.5377	0.0934	0.6124	0.4912	0.2614	0.8028	(0.1442,0.0002,0.0000)	26658220
0.8150	0.6618	0.5998	0.8521	0.9369	0.0827	0.4943	0.7209	0.7492	0.8209	(0.1441,0.0001,0.0000)	27280069
0.3562	0.9991	0.1069	0.2166	0.3059	0.3926	0.1060	0.1954	0.5704	0.2292	(0.1443,0.0001,0.0000)	28279550
0.8931	0.6182	0.0310	0.9379	0.4717	0.3451	0.2211	0.2032	0.9400	0.3335	(0.1441,0.0002,0.0000)	29950060
0.4912	0.0531	0.6247	0.8589	0.0886	0.4771	0.1551	0.0907	0.6607	0.9159	(0.1441,-0.0001,0.0000)	32069349
0.8048	0.6173	0.6577	0.2040	0.1965	0.8819	0.2583	0.3428	0.3045	0.6468	(0.1444,-0.0000,0.0000)	33029447
0.9831	0.6182	0.6716	0.2823	0.7938	0.1041	0.8444	0.0033	0.9646	0.2898	(0.1443,0.0000,0.0000)	34152953
0.9769	0.8061	0.6885	0.6307	0.6146	0.0288	0.9751	0.3801	0.8929	0.2650	(0.1441,-0.0001,0.0000)	34622529
0.1681	0.9759	0.6775	0.4843	0.4955	0.2290	0.3911	0.3823	0.9352	0.5722	(0.1441,-0.0001,0.0000)	34756864
0.3063	0.2927	0.1993	0.9319	0.0749	0.0766	0.0990	0.2953	0.4091	0.7058	(0.1442,0.0002,0.0000)	36555306
0.2918	0.3819	0.7417	0.8607	0.3698	0.0340	0.5890	0.0713	0.9567	0.8361	(0.1440,-0.0001,0.0000)	37010968
0.9393	0.9492	0.4603	0.0647	0.2622	0.6585	0.2496	0.7225	0.1423	0.1574	(0.1441,0.0000,0.0000)	37951051
0.7987	0.6513	0.7038	0.6640	0.9687	0.1961	0.3072	0.8727	0.6369	0.8268	(0.1443,-0.0002,0.0000)	38385707
0.5838	0.8829	0.4263	0.7696	0.4589	0.1384	0.1309	0.7711	0.9639	0.0853	(0.1441,0.0002,0.0000)	38979786
0.9589	0.3768	0.7208	0.8120	0.2897	0.5018	0.0727	0.8124	0.6995	0.3711	(0.1444,-0.0001,0.0000)	39521874
0.7909	0.6552	0.8497	0.2028	0.3890	0.1943	0.8120	0.2842	0.5483	0.3328	(0.1443,-0.0000,0.0000)	39803646
0.6345	0.9069	0.6368	0.5843	0.0910	0.8276	0.4774	0.6995	0.0719	0.9929	(0.1444,0.0000,0.0000)	39942479
0.5663	0.9843	0.7713	0.6103	0.4685	0.8650	0.0403	0.9908	0.5731	0.8632	(0.1441,0.0001,0.0000)	40005890
0.8728	0.1389	0.6888	0.2389	0.2400	0.3232	0.1883	0.4522	0.1780	0.4152	(0.1443,-0.0002,0.0000)	42059279
0.6443	0.3559	0.6694	0.2290	0.0085	0.0385	0.8009	0.0353	0.2310	0.3325	(0.1440,0.0001,0.0000)	42425923
0.8126	0.3716	0.7995	0.2568	0.7050	0.0427	0.8813	0.1916	0.2756	0.8944	(0.1440,0.0001,0.0000)	42778753
0.3686	0.9360	0.9034	0.2994	0.2329	0.2854	0.2509	0.9714	0.4710	0.2763	(0.1440,0.0001,0.0000)	43047053
0.6726	0.1739	0.6567	0.5868	0.4450	0.1718	0.4778	0.2616	0.3247	0.9496	(0.1440,-0.0000,0.0000)	44284181
0.4967	0.9421	0.1365	0.0845	0.8819	0.0465	0.3796	0.2688	0.3180	0.5959	(0.1442,-0.0002,0.0000)	45924345
0.9262	0.9746	0.7689	0.6956	0.2547	0.4750	0.6926	0.5954	0.9707	0.1155	(0.1443,-0.0002,0.0000)	46477906
0.7729	0.6378	0.4071	0.5722	0.6549	0.5862	0.2721	0.0655	0.9912	0.6389	(0.1441,-0.0002,0.0000)	48112853
0.9742	0.5811	0.9890	0.1142	0.9221	0.2109	0.8036	0.1524	0.9595	0.4314	(0.1443,0.0001,0.0000)	48859468
0.7333	0.2972	0.7657	0.9021	0.1005	0.5451	0.5451	0.1639	0.7508	0.8150	(0.1442,-0.0001,0.0000)	51245409
0.9510	0.9264	0.1510	0.3334	0.1206	0.9373	0.0889	0.4614	0.2876	0.2751	(0.1444,0.0001,0.0000)	51405190
0.8996	0.0566	0.8450	0.6019	0.6098	0.0170	0.4319	0.2832	0.8729	0.4488	(0.1444,-0.0001,0.0000)	53545576
0.7412	0.6468	0.5186	0.7521	0.5680	0.0024	0.8970	0.3842	0.3932	0.8612	(0.1440,-0.0000,0.0000)	53969979
0.0038	0.8029	0.1487	0.0820	0.5561	0.0820	0.1347	0.1274	0.2084	0.7999	(0.1442,0.0001,0.0000)	54062646
0.9676	0.8643	0.9785	0.6864	0.7605	0.2285	0.8393	0.7588	0.9094	0.5254	(0.1443,0.0001,0.0000)	55384476
0.6878	0.8046	0.3054	0.3159	0.7364	0.4122	0.3731	0.4495	0.1451	0.9572	(0.1443,0.0002,0.0000)	57026435
0.8929	0.4051	0.6063	0.3343	0.0026	0.4203	0.4051	0.4722	0.1116	0.3251	(0.1444,0.0001,0.0000)	57897866
0.5415	0.9449	0.7271	0.3384	0.9531	0.6002	0.2164	0.5565	0.8226	0.9195	(0.1443,0.0001,0.0000)	59705876
0.8537	0.5600	0.2640	0.5553	0.2913	0.4432	0.4826	0.2106	0.2657	0.6681	(0.1442,-0.0001,0.0000)	60595973
0.9041	0.3006	0.3256	0.9792	0.2170	0.1405	0.2559	0.5666	0.5126	0.3717	(0.1444,0.0000,0.0000)	62583556
0.7282	0.7508	0.2890	0.1713	0.4968	0.4555	0.1420	0.0424	0.9153	0.0454	(0.1444,-0.0000,0.0000)	63348944
0.4176	0.9593	0.2999	0.2063	0.3361	0.6356	0.1141	0.1489	0.6895	0.3899	(0.1442,0.0001,0.0000)	63421182
0.4568	0.8509	0.4768	0.6651	0.2522	0.3091	0.0083	0.8983	0.5829	0.2921	(0.1443,0.0002,0.0000)	64759630
0.1762	0.8800	0.8947	0.5307	0.2458	0.0712	0.9825	0.3382	0.4599	0.9036	(0.1440,0.0000,0.0000)	64831114
0.7800	0.6337	0.8042	0.3503	0.5633	0.4435	0.0468	0.8833	0.5951	0.3693	(0.1443,-0.0002,0.0000)	69530589
0.6821	0.3076	0.4948	0.6534	0.1724	0.3670	0.1158	0.1804	0.8772	0.2574	(0.1442,0.0002,0.0000)	71182849
0.7991	0.7354	0.7361	0.3958	0.4309	0.3647	0.3459	0.5409	0.8658	0.1211	(0.1440,0.0000,0.0000)	71549124
0.6943	0.1135	0.6344	0.4778	0.1273	0.0381	0.5206	0.0825	0.4561	0.3935	(0.1444,0.0001,0.0000)	72604884
0.4993	0.3496	0.9867	0.8256	0.1451	0.0486	0.4831	0.8130	0.4293	0.6788	(0.1443,0.0000,0.0000)	73797229
0.7246	0.9738	0.3933	0.7195	0.7795	0.5397	0.5202	0.3364	0.8197	0.9092	(0.1442,0.0001,0.0000)	74651478
0.4713	0.9350	0.9866	0.2926	0.7760	0.7480	0.0418	0.8331	0.7970	0.8458	(0.1443,-0.0001,0.0000)	75735655
0.8395	0.4873	0.0435	0.5250	0.4151	0.0722	0.1818	0.2874	0.4807	0.1419	(0.1440,-0.0001,0.0000)	76622889
0.8513	0.7738	0.5701	0.1187	0.5064	0.6591	0.1888	0.3954	0.6303	0.2923	(0.1440,0.0001,0.0000)	77309138

0.3547	0.8712	0.7219	0.5903	0.2872	0.5590	0.2915	0.6775	0.4844	0.8555	(0.1442,0.0001,0.0000)	78735287
0.9785	0.6815	0.5069	0.5012	0.3318	0.5277	0.4265	0.6615	0.2029	0.5644	(0.1442,0.0000,0.0000)	79476777
0.1532	0.9853	0.3882	0.1546	0.3441	0.5863	0.3681	0.0964	0.2802	0.9748	(0.1443,-0.0000,0.0000)	82111410
0.1611	0.9112	0.8203	0.8056	0.0733	0.0674	0.8990	0.5109	0.5093	0.8118	(0.1441,0.0001,0.0000)	82770079
0.4911	0.6019	0.7641	0.2828	0.8746	0.0486	0.1127	0.6474	0.7941	0.4704	(0.1442,0.0000,0.0000)	83802401
0.8108	0.8484	0.9455	0.7751	0.8075	0.3360	0.7023	0.7105	0.9563	0.7860	(0.1440,0.0000,0.0000)	85162304
0.3744	0.8399	0.7840	0.6751	0.1226	0.1150	0.7893	0.4749	0.6063	0.5314	(0.1442,0.0002,0.0000)	85547334
0.7383	0.4810	0.5314	0.0679	0.5574	0.2335	0.2321	0.1847	0.6260	0.2102	(0.1441,0.0001,0.0000)	86056958
0.8755	0.1924	0.9246	0.4051	0.1645	0.4234	0.2901	0.1828	0.9887	0.0693	(0.1442,0.0002,0.0000)	86856272
0.2138	0.2422	0.9214	0.7163	0.1732	0.0987	0.4660	0.0875	0.9072	0.7346	(0.1441,-0.0000,0.0000)	87013313
0.6923	0.4388	0.6192	0.4897	0.1303	0.2069	0.0339	0.7570	0.4955	0.0470	(0.1441,-0.0001,0.0000)	93176799
0.3255	0.7923	0.9016	0.3426	0.0629	0.6408	0.4917	0.3695	0.4644	0.7729	(0.1440,0.0000,0.0000)	96519374
0.6259	0.6279	0.2222	0.8639	0.4050	0.4456	0.3713	0.1633	0.6423	0.8598	(0.1443,-0.0001,0.0000)	97336759