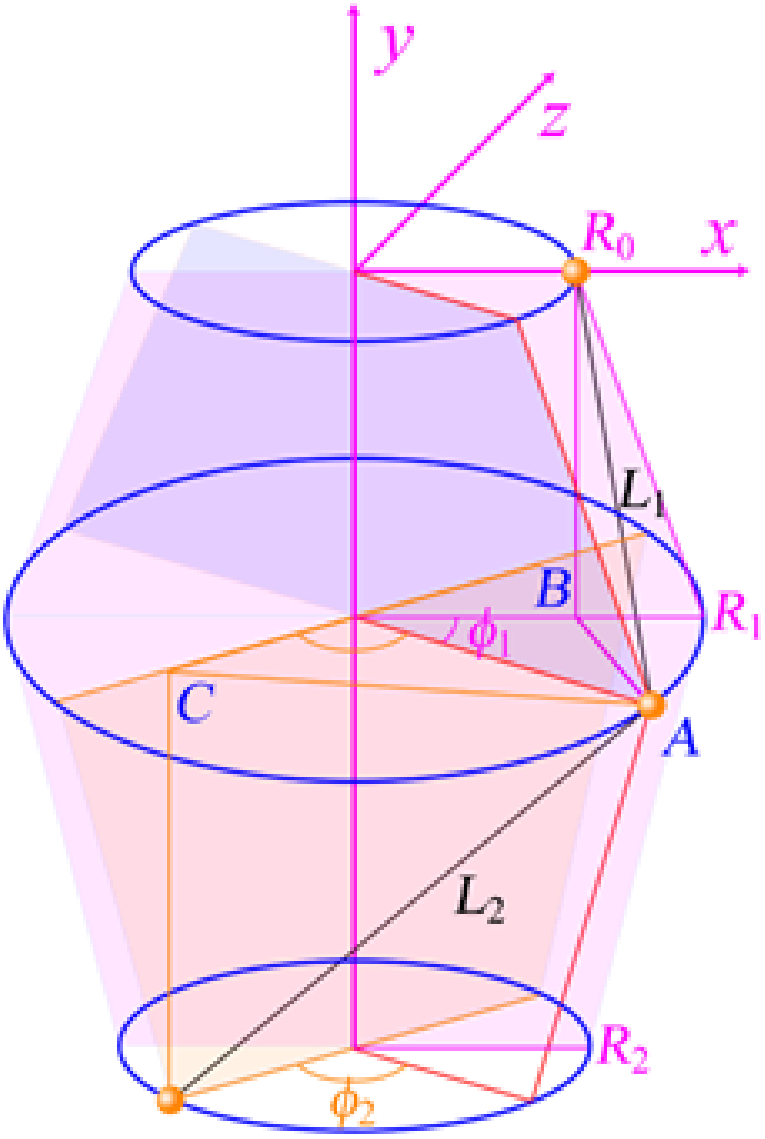
从一种叫 Astrojax 的玩具讲起



摘要

Astrojax 又称” 太空悠悠球”，是一款风靡全美国的玩具，其结构类似于双摆，但又有明显不同——中间球可以沿着绳子滑动。目前国际上对于双摆的研究较为成熟，仿真多样，但是对于 Astrojax 这款新兴玩具的仿真和稳定状态的研究还几乎是空白。本文重点研究了 Astrojax 的水平轨道稳定态和仿真算法。Astrojax 的稳定状态类似于弹簧振子的稳定受迫运动状态，即系统能量的输入和输出相等，系统达到稳定状态，因此我们从弹簧振子讲起，再从单摆过渡到 astrojax。我们的仿真模型是基于 Unity3D 开发的，背后的物理模型我们使用 C# 进行构建，我们利用保辛算法对运动进行演化，得到了与实际相符的仿真结果, 我们同时针对 windows 平台和 Android 平台发布了我们的仿真程序，这样大家就可以在自己的手机或者电脑上操纵 Astrojax。

关键词： Astrojax; 弹簧振子; Unity3D; 稳定受迫振动

目录

1 Astrojax 介绍

Astrojax 是一款风靡全美国的玩具：将绳子穿过一个带有洞的球，这样球就可以沿着绳子自由移动。把另一个球系在绳子的一端。当你周期性地移动绳子的自由端时，你可以观察到两个球的复杂运动。此系统不同于双摆—中间的球可以沿着绳子自由滑动。因其运动类似于行星，故命名为 Astrojax.

Astrojax 富含教育意义，美国太空总署 NASA，将 astrojax 作为其【太空玩具】教育计划的一部分，并在 2003 年由宇航员带入太空进行了演示。

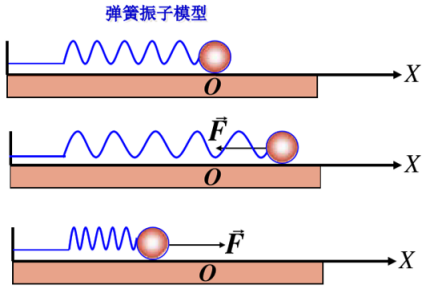
Astrojax 作为一个非线性系统，有无穷多种稳定模式, 可以理解为：一个耗散系统中的受迫运动，对应模型是受迫谐振子。摩檫力是一种耗散，当输入功率等于输出功率时，以某种稳定的受迫振动频率运动。每一种玩法都对应了一种稳态，我们所寻找的稳态，其实就是一种稳定的受迫振动状态。

加阻尼加驱动可能有稳定解，这种稳定解是什么呢？如何找到这种稳定解，利用同样的思路找到更多稳定解，从而为游戏带来更过花样！所以我们从弹簧振子讲起，再从单摆过渡到 astrojax, 最后分析分析一下我们的仿真模型原理.

2 弹簧振子

2.1 简谐振动

弹簧振子是研究简谐振动的一种理想模型，将光滑水平面物体（可视为质点）与一端固定的轻弹簧相连，就构成了一个弹簧振子。将物体在平衡位置不远处释放，物体将在回复力作用下返回平衡位置；由于惯性，物体将越过平衡位置，此后在回复力作用下减速，到达最大位移后返回，如此往复，形成在平衡位置的振动。



设物体质量为 m，以 O 为坐标原点沿水平方向建立 Ox 坐标轴，当 m 在平衡位置附近发生不大的位移 x 时，作用在 m 上的弹性力可以表示为

$$F = -kx$$

式中 k 是弹簧的劲度系数，负号反映力的方向始终与位移方向相反，即指向平衡位置。这种与 x 成正比的回复力称为线性回复力。根据牛顿第二定律，可列出动力学方程为：

$$m\frac{d^2x}{dt^2} = -kx$$

令 $\omega^2 = \frac{k}{m}$，则上面的方程可以改写为

$$\frac{d^2x}{dt^2} + \omega^2 x = 0$$

这是一个二阶线性齐次微分方程，其解可以表示为

$$x = Acos(\omega t + \phi) \qquad (1)$$

式中 A、 为两个积分常量， 则由系统自身特性决定。可见弹簧振子做简谐运动. 对公式（1）两边对时间 t 求一阶和二阶导数，分别得到速度 v 和加速度 a，即

$$v = \frac{dx}{dt} = -A\omega sin(\omega t + \phi)$$

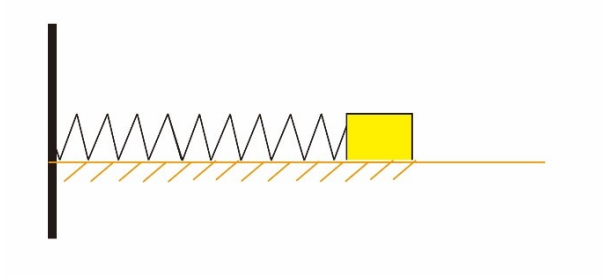
$$a = \frac{d^2x}{dt^2} = -A\omega cos(\omega t + \phi)$$

可见他们也做简谐振动可见他们也做简谐振动。

2.2 阻尼振动

前面讨论的弹簧振子简谐振动，是既无外界输入能量，也无阻力存在从而没有能量耗散的理想振动，所以振动系统的能量守恒，振幅不会随时间发生变化，这样的振动称为无阻尼自由振动，然而任何实际振动过程都不可能没有能量损耗，例如弹簧振子的空气阻力和摩擦力、振荡电路的电阻等，它们都会使振动能量转化为热能: 此外，振动还会以波的形式向四

周辐射能量. 所以，如无其他能量补充，系统的振幅和能量都会逐渐衰减，直至最后停止. 这种系统在某种阻碍作用下，造成能量损失而使振幅随时间衰减的振动称为阻尼振动。



机械振动中阻尼以黏性阻力为主，而电磁振荡中辐射引起的阻尼常常与黏性阻力相似. 因此，下面仅考虑黏性阻力，在运动速率不大时，物体所受粘性阻力 F 可以认为与速率成正比，其方向与速度方向相反，即

$$F_f = -\gamma \frac{dx}{dt}$$

式中　称为阻力系数，考虑黏性阻力后，水平弹簧振子的运动方程变为

$$m\frac{d^2x}{dt^2} = -kx - \gamma \frac{dx}{dt}$$

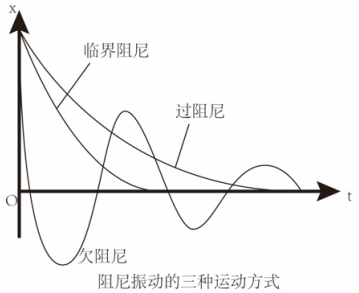
或

$$\frac{d^2x}{dt^2} + 2\beta\frac{dx}{dt} + \omega^2 x = 0 \qquad (2)$$

式中 $\omega = \sqrt{\frac{k}{m}}$ 是振动系统无阻尼时的固有角频率，$\beta = \frac{\gamma}{2m}$ 是表征系统阻尼大小的常量，称为阻尼系数。该微分方程特征值有两个，分别为

$$\lambda = -\beta \pm \sqrt{\beta^2 + \omega_0^2}$$

按　大小不同，方程（1）的三种不同形式的解对应阻尼振动的三种可能的运动方式（如下图）



阻尼振动的三种运动方式

以下对三种情况分别讨论

2.2.1 欠阻尼运动

当阻尼较小，即当 $\beta^2 < \omega^2$ 时，方程（2）的解为

$$x = A_0 e^{-\beta t} cos(\omega t + \phi)$$

可见系统的振幅不断衰减，所以不是简谐报动，但仍然在平衡位置附近作往复运动，可以看作是一种准周期运动。常称为欠阻尼振动. 式中

$$\omega = \sqrt{\omega_0^2 - \beta^2} < \omega_0$$

可见，与无阻尼自由振动相比，由于阻尼存在，其“频率”变小，而“周期”则变长，为

$$T = \frac{2\pi}{\omega} = \frac{2\pi}{\sqrt{\omega_0^2 - \beta^2}} > \frac{2\pi}{\omega}$$

振幅衰减的快慢程度，可用相邻两次振动的振幅之比的对数来反映，称为对数减缩率，用 Λ 表示，即

$$T = \frac{2\pi}{\omega} = \frac{2\pi}{\sqrt{\omega_0^2 - \beta^2}} > \frac{2\pi}{\omega}$$

可见 Λ 与　成正比。

2.2.2 过阻尼运动

当阻尼较大，即当 $\beta^2 > \omega^2$ 时，方程 (2) 的解为

$$x = c_1 e^{-(\beta - \sqrt{\beta^2 - \omega_0^2})t} + c_2 e^{-(\beta + \sqrt{\beta^2 - \omega_0^2})t}$$

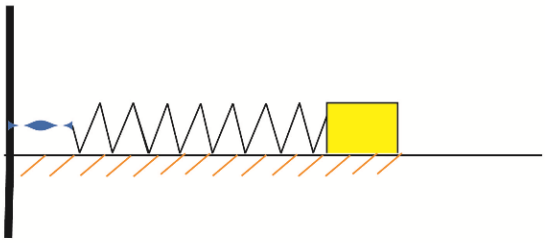
这不是一种周期运动，物体将缓慢的逼近平衡位置，但不会越过平衡位置，即称为过阻尼振动。

2.2.3 临界阻尼振动

当 $\beta^2 = \omega^2$ 时, 方程 (2) 的解为

$$x = (c_1 + c_2 t) e^{-\beta t}$$

这是一种刚好介于欠阻尼振动和过阻尼之间的临界情况，称为临界阻尼。和前两种情况相比，这种非周期性运动能够在最短时间回到平衡位置并停止下来，因此要让物体在不发生振动的情况下最快回到平衡位置，常用施加临界阻尼的方法。例如指针式仪表中常使仪表工作在临界阻尼状态，这样指针可以快速地停在读数值处.



2.3 受迫振动

在有阻尼的情况下，由于能量损耗，系统最终会停止在平衡位置，要使系统的振动状态持久而不衰减，可以对系统施加一个周期性的外界驱动力，通过做功不断向系统提供能量来实现，这种在周期性外界驱动力作用下的振动，称为受迫振动

设系统受到的简谐驱动力为 F=cos ft 的作用，F 是力幅，ω_f 是驱动力变化的角频率，则受迫振动的微分方程

$$m\frac{d^2x}{dt^2}+\gamma\frac{dx}{dt}+kt=Fcos(\omega_f t)$$

或者

$$\frac{d^2x}{dt^2}+2\beta\frac{dx}{dt}+\omega_0^2 t=f_0cos(\omega_f t)$$

该微分方程的解为：

$$x=A_0e^{-\beta t}cos(\omega t+\phi)+Acos(\omega_f t+\phi) \qquad (3)$$

其中第二项为一种稳定的等幅振动，它表示阻尼振动衰减后系统在简谐外力作用下受迫振动的稳定状态，成为稳态响应

$$x=Acos(\omega t+\phi) \qquad (4)$$

将该解带入（3），容易得到

$$A=\frac{f_0}{\sqrt{(\omega_0^2-\omega_f^2)^2+4\beta^2\omega_f^2}}$$

$$tan(\phi)=-\frac{2\beta\omega_f}{\omega_0^2-\omega_f^2}$$

4）式表示的稳定受迫振动状态，形式上虽与简谐振动类似，但它不是简谐振动，因为它的频率为外力频率 f. 与固有频率 0 无关. 振幅 A 和 也并非取决于初始条件，而是与系统、阻尼和外界驱动力的特征有关。可见，系统在无阻尼也没有外界作用，即在线性回复力作用下的无阻尼自由振动才是简谐振动，简谐振动的频率仅由系统性质决定。

2.4 共振

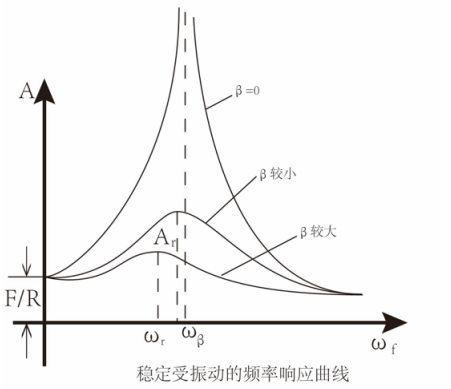
稳定受迫振动的频率虽与外力相同，但相位却不一样，由于阻尼的存在，使得振动对外里的响应滞后，相位上落后了 ϕ，根据受迫振动的方程解可得知，当外界驱动力的频率为某一特定值时，受迫振动中位移振幅将达到极大值，这一现象称为位移共振，称为共振。共振时对应的 ω_f 值为 ω_t, 称为共振角频率。由极值条件 $\frac{dA}{d\omega_f}=0$, 可以求得 ω_f 和共振时的振幅 At 分别为

$$\omega_f=\sqrt{\omega^2-2\beta^2}$$

,

$$A=\frac{f_0}{2\beta\sqrt{\omega_0^2-2\beta^2}}$$

下图所示是不同阻尼情况下 A 随外界驱动力的角频率变化的曲线 f。由此看出一般共振频率不等于系统的固有频率，阻尼 越小，则共振频率越接近于固有角频率 ω_0，Ar 也越大，当 $\beta << \omega_r$ 时，共振频率 $\omega_0 \approx \omega_0$，共振振幅趋于无穷大.



稳定受振动的频率响应曲线

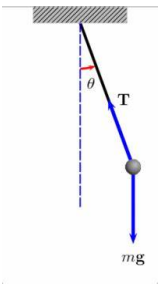
对受迫振动解对时间求导数可得速度振幅为

$$v_m=\omega_f A+\frac{\omega_f f_0}{\sqrt{\omega_0^2-\omega_f^2+4\beta^2\omega_f^2}}$$

当 $\omega_f=\omega_0$ 时速度振幅最大，为 $\frac{f}{2\beta}$，这称为速度共振。速度共振时振动位移的相位比驱动力落后 $\frac{\pi}{2}$，即振动速度与驱动力同相位，因此驱动力总是做正功，从而外界把能量转化为系统能量，因此速度共振也称为能量共振。一般情况下，由于位移共振与速度共振频率不同，位移共振时驱动力并非总是做正功。但在阻尼 β 趋近于零的情况下，当 $\omega_r=\omega_0$ 时位移共振和速度共振同时发生，使位移振幅和速度振幅急剧增大，系统则发生强烈共振。

3 单摆

单摆是能够产生往复摆动的一种装置，将无重细杆或不可伸长的细柔绳一端悬于重力场内一定点，另一端固结一个重小球，就构成单摆。若小球只限于铅直平面内摆动，则为平面单摆，若小球摆动不限于铅直平面，则为球面单摆。



3.1 球面单摆的圆周运动 (无阻尼)

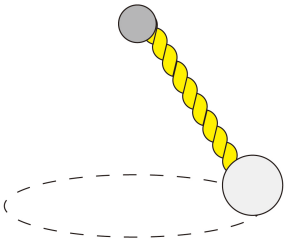
球的运动轨迹为圆周，圆周半径为 $Lsin(\theta)$ ，可列出如下方程：

$$mgtan(\theta) = m\omega^2 Lsin(\theta)$$

当极角给定时，我们可以求得其稳定运动时的角速度和半径：

$$R = Lsin(\theta)$$

$$\omega = \sqrt{\frac{1}{Lcos(\theta)}}$$

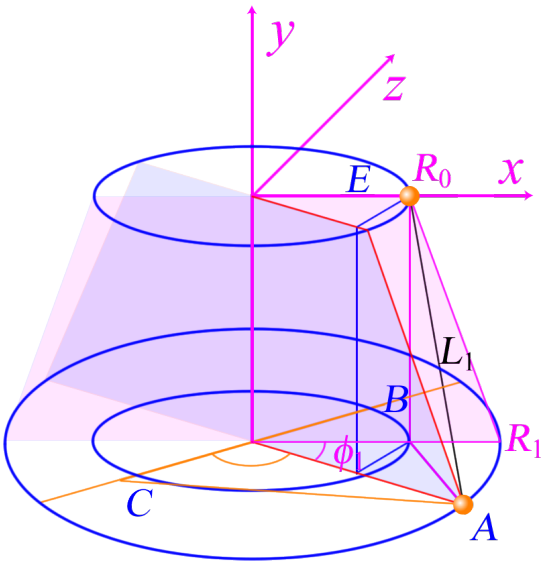


3.2 球面单摆的圆周运动有阻尼

当考虑阻尼时，假设阻力的大小与小球的速度大小成正比，方向与小球的运动速度方向相反，从而得出一下方程：

$$\frac{dv}{dt} = -\eta v$$

$$m\frac{v^2}{Lsin(\theta)} = mgtan(\theta)$$



3.3 球面单摆的圆周运动 (受迫运动)

对于有阻尼球面摆而言，R_1, ϕ_1, h_1 是已知的，

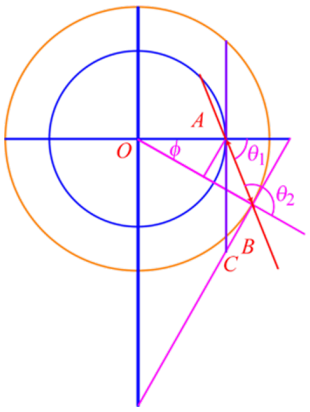
$$T_{01||} = \eta\omega R_1$$

$$t_{01\perp} = m_1\omega^2 R_1$$

$$T_{01y} = m_1 g$$

绳长约束

$$L_1^2 = R_0^2 + R_1^2 - 2R_0R_1cos(\phi) + H_1^2$$



受力方向约束

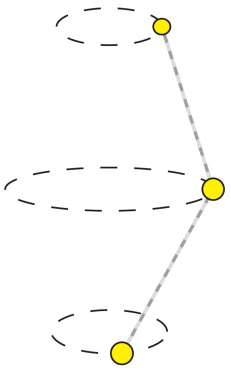
$$\frac{T_{01y}}{\sqrt{T_{01||}^2 + t_{01\perp}^2}} = \frac{H_1}{\sqrt{L_1^2 = R_0^2 + R_1^2 - 2R_0R_1cos(\phi)}}$$

受力方向约束

$$\frac{T_{01||}}{t_{01\perp}} = tan(\pi - \alpha_1) = \frac{sin(\phi)}{\frac{R_1}{R_0} - cos(\phi)}$$

4 Astrojax

4.1 1.Astrojax——复杂的双摆球运动



还记得曾经风靡一时的悠悠球吗，它已经逐渐被人们所遗忘，不过目前却出现了一个类似的替代品，悠悠球是一根绳，一个球的花式运动，从 1A 到 5A 玩法多种多样。最近国际上新出现的类似的产品由三个球一根绳组成，一个球在最上端，有人来控制，一个球在最下端，另一个在中间。它就是 astrojax，如图所示。一个新兴的悠悠球，它也有许多种玩法，比如 horizontal orbits, vertical orbits, butterfly orbits, or wild and crazy patterns。这些不过是开始而已。

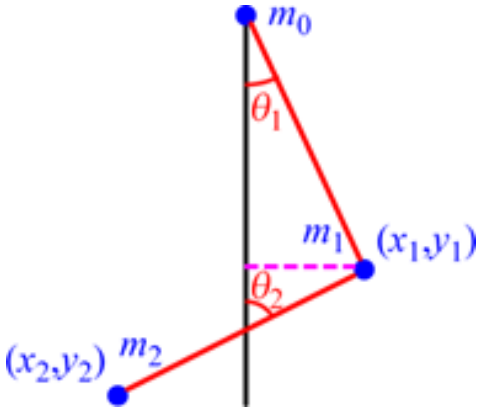
这项运动，已经有许多人在参与了，玩法更是多种多样，它的兴起得益于 NASA 宇航员在太空表演的一段视频，于是越来越多的人发现了它的存在。Astrojax 是一款风靡全美国的玩具：将绳子穿过一个带有洞的球，这样球就可以沿着绳子自由移动。把另一个球系在绳子的一端。当你周期性地移动绳子的自由端时，你可以观察到两个球的复杂运动。此系统不同于

双摆—中间的球可以沿着绳子自由滑动。因其运动类似于行星，故命名为 Astrojax。更是有许多能人义士开发了多种多样的应用领域。最简单的像是，舞台表演这一类的玩法，进一步还有医学领域的物理疗法帮助病人们做康复训练，更深层次的，就是专业球手的手眼协调高难度花式动作。除此之外，这样有趣的玩具怎么可能不引起科学家们的重视呢，各个物理方面的科学家们对这一现象展开了多种多样的研究，能量，摆动，周期等一系列话题，今天，我们就稳定状态也展开一次对玩具的研究，实现在电脑上玩新玩具。

4.2 水平轨道无阻尼运动

4.2.1 模型建立

首先，让我们来看一下它在水平轨道的无阻尼运动，这是 astrojax 的最简单的玩法之一，也是最简单的稳定状态，仔细看一下这一运动状态。最上端有一个质量为 m0 的小球，中间有一个 m1 的不固定小球在轨道中间可以自由旋转，最下端是一个质量为 m2 的小球，跟随上面的小球一起做圆周运动。那么这样问题就很清楚了，就是如右图所示的简单模型。



4.2.2 方程求解

可以发现这里一共有六个未知量 T 球的圆周运动周期, ω 圆周运动的角频率, θ_1 上方两个球连线与竖直方向的夹角, θ_2 下方两个小球与竖直方向的夹角, L1 第一段绳子的长度, L2 后半段绳子的长度，同时还有五个方程，

$$T\cos\theta_1 = m_1 g + T\cos\theta_2$$

$$T\sin\theta_1 + T\sin\theta_2 = m_1\omega^2 l_1\sin\theta_1$$

$$T\sin\theta_2 = m_2\omega^2(l_2\sin\theta_2 - l_1\sin\theta_1)$$

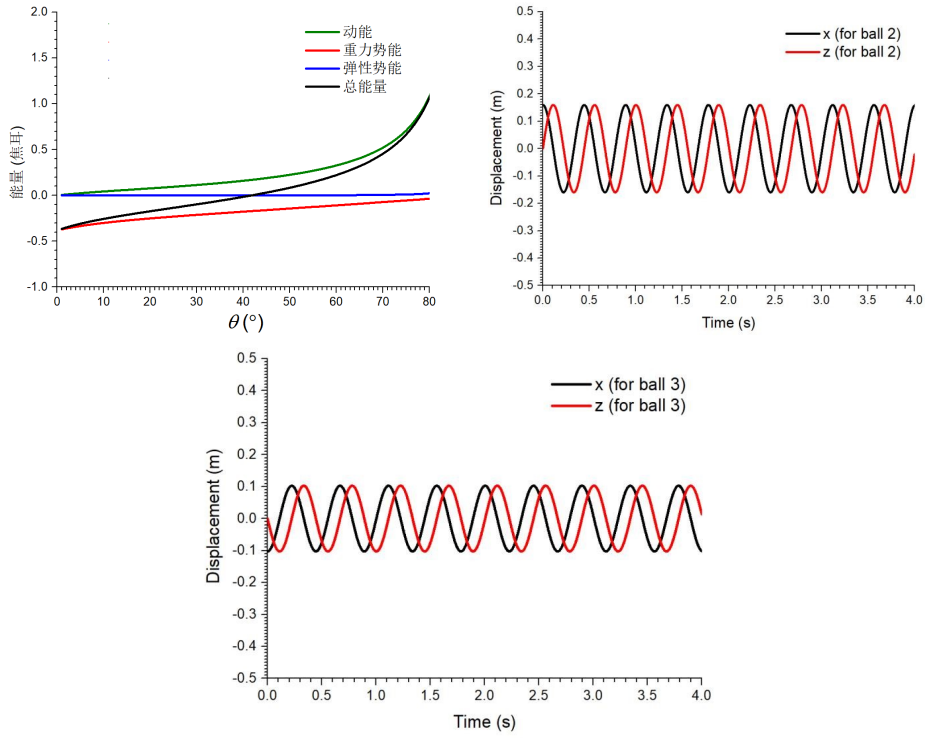
$$T\cos\theta_2 = m_2 g$$

$$l_1 + l_2 = l_0$$

这样的话，只要我们给定了其中任何一个未知量，那么剩下的未知量就都是可解的了，因此，这样的运动状态就是可以完全确定下来的了。那么，这就是我们通过计算机模拟仿真实现的第一个图景。

4.2.3 问题拓展

同时就这个问题我们还做了一些与实际相关的问题的探讨，我们通过上面这些方程得出的数值解，输入到计算机中进行数据的解析，得到了 astrojax 在无阻尼驱动状态下，达到稳定状态后能量随着极角的变化，以及中间球和底端球的坐标随时间的变化。

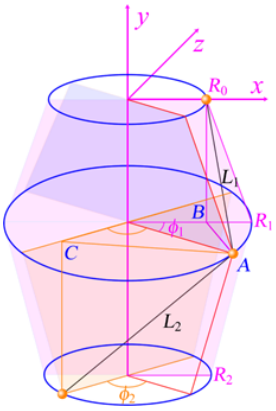


4.3 水平轨道受迫运动——基础招式

4.3.1 模型建立

既然我们已经分析清楚了最基础的运动状态，那么就可以来看看实际的应用场景，astrojax 既然是作为一项玩具运动项目在美国兴起的，那么它的最基础的招式就当然应该作为我们运动分析必不可少的一环，也就是，接下来我们应该看一看 astrojax 在水平运动的时候的具体运动过程的探讨.

我们可以将问题简化一下，将三个小球看作三个质点，将中间的小球记作 A, 将上面的小球记作 B, 最下面的小球记作 C，并且假设他们在空间范围内做逆时针转动。既然我们已经将这样的实际模型放在了空间范围内进行研究，就应该应用一下最普遍的办法，空间直角坐标系，因为最后是要应用于 unity3D 进行虚拟仿真的模型建立，那么就可以直接应用 unity3D 的内置坐标系来建立对应的方程组。如右图所示。



4.3.2 方程求解

选取顶端的小球做圆周运动的圆心作为空间直角坐标系的原点，水平向右为 x 轴，水平向后的 z 轴，以及竖直向上的 y 轴，建立左手坐标系，那么，xyz 三个方向的单位向量就是

$\vec{x} = (1, 0, 0)'$

$\vec{y} = (0, 1, 0)'$

$\vec{z} = (0, 0, 1)'$

可以假设顶端小球半径为 R0，中间小球半径为 R1，底端小球半径为 R2，顶端小球与中间小球的绳子长度为 L1，中间小球与底端小球的绳子长度为 L2，第一个小球所在位置投影到第二个小球圆周运动的平面，那么两条线的夹角记作 Φ1，这就是这两个小球做圆周运动时对应的相位差，同理，第二个小球投影到底端小球，这两个小球的相位差记作 Φ2，于是得到 B、A、C 三个小球的对应坐标位置。

$B : ((R_0 \cos(\omega t)), 0, R_0 \sin(\omega t))$

$A : (R_1 \cos(\omega t - \phi_1), -h_1, R_0 \sin(\omega t - \phi_1))$

$C : (R_2 \cos(\omega t - \phi_1 - \phi_2), -h_1 - h_2, R_0 \sin(\omega t - \phi_1 - \phi_2))$

由此就可以得到 AB 和 CA 向量的坐标表示，因此得到相应的单位向量的坐标表示。

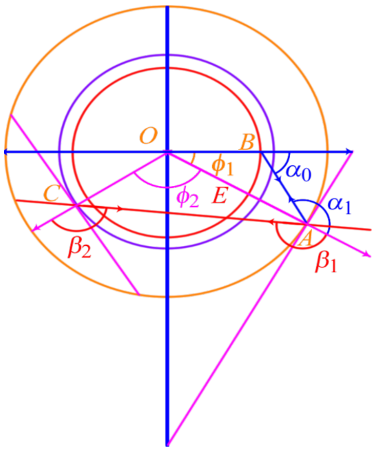
$\vec{AB} = (R_0 \cos(\omega t) - R_1 \cos(\omega t - \phi_1), h_1, R_0 \sin(\omega t) - R_0 \sin(\omega t - \phi_1))$

$\vec{CA} = (R_1 \cos(\omega t - \phi_1) - R_2 \cos(\omega t - \phi_1 \phi_2), h_2, R_0 \sin(\omega t - \phi_1) - R_0 \sin(\omega t - \phi_1 - \phi_2))$

$\vec{e_1} = \frac{\vec{AB}}{|\vec{AB}|}$

$\vec{e_2} = \frac{\vec{CA}}{|\vec{CA}|}$

从上往下俯视观察，得到对应的平面投影，



这样的话问题就简化为在平面运动的圆周运动，这样的运动就不得不提到圆周运动最普遍的问题处理方法，将 A 和 C 分解为对应的切向加速度以及对应的轴向加速度，并且可以由此得到对应方向的单位向量。

$\vec{A_\perp} = (-R_1 \cos(\omega t - \phi_1), 0, -R_1 \sin(\omega t - \phi_1))$

$\vec{C_\perp} = (-R_2 \cos(\omega t - \phi_1 - \phi_2), 0, -R_2 \sin(\omega t - \phi_1 - \phi_2))$

$e_{\vec{A\perp}} = \frac{\vec{A_\perp}}{|\vec{A_\perp}|}$

$e_{\vec{C\perp}} = \frac{\vec{C_\perp}}{|\vec{C_\perp}|}$

$\vec{A_{||}} = (-R_1\omega \sin(\omega t - \phi_1), 0, R_1\omega \cos(\omega t - \phi_1))$

$\vec{C_{||}} = (-R_2\omega \sin(\omega t - \phi_1 - \phi_2), 0, -R_2\omega \cos(\omega t - \phi_1 - \phi_2))$

$e_{\vec{A||}} = \frac{\vec{A_{||}}}{|\vec{A_{||}}|}$

$e_{\vec{C||}} = \frac{\vec{C_{||}}}{|\vec{C_{||}}|}$

当我们建立了这些基础向量，对于这个实际难题就可以直接简化为最简单的图景。因为对于 astrojax 来说中间小球是要自由滑动的，因此可以忽略掉它与绳子之间的摩擦阻力，那么，对于中间小球就可以简化为动滑轮，这样的话，绳子两端就可以近似处理为两端相等的绳子拉力，于是，将中间与底端小球看作一个整体，将上半段绳子投影到竖直方向，这个分力就应

该与中间和底端的小球重力平衡，同样的，这段绳子投影到水平面内，这个分力就可以作为后两个小球做圆周运动的轴向力，驱动中间和底端小球做圆周运动。同理，对于后半段绳子做同样的投影处理，就可以都得到对于底端小球的受力分析。因此得到了这样的六个方程，

$$T\vec{e_1}\cdot\vec{y}=(m_1+m_2)g$$
$$T\vec{e_2}\cdot\vec{y}=m_2g$$
$$T\vec{e_1}\cdot\vec{e_{A\perp}}-T\vec{e_2}\cdot\vec{e_{A\perp}}=\omega^2m_1R_1$$
$$T\vec{e_1}\cdot\vec{e_{A||}}-T\vec{e_2}\cdot\vec{e_{A||}}=\eta\omega R_1$$
$$T\vec{e_2}\cdot\vec{e_{C||}}=\eta\omega R_2$$
$$T\vec{e_2}\cdot\vec{e_{C\perp}}=\omega^2m_2R_2$$

这样的话，对于之前的六个未知量就都可解了。既然我们已经得到的这样的六个未知量，于是我们就可以通过如下公式

Out[48]=
$$\frac{h1\,T}{\sqrt{\text{Abs}[h1]^2+\text{Abs}\left[\sqrt{1-a^2}\,R1\right]^2+\text{Abs}[R0-a\,R1]^2}}$$

Out[49]=
$$\left|\frac{h2\,T}{\sqrt{\text{Abs}[h2]^2+\text{Abs}[a\,R1-b\,R2]^2+\text{Abs}\left[-\sqrt{1-a^2}\,R1+\sqrt{1-b^2}\,R2\right]^2}}\right|$$

化简 | 分子 | 分母 | 转换成共轭 | 更多

Out[50]=
$$T\left(\frac{(1-a^2)\,R1^2}{\sqrt{\text{Abs}[a\,R1]^2+\text{Abs}\left[\sqrt{1-a^2}\,R1\right]^2}\sqrt{\text{Abs}[h1]^2+\text{Abs}\left[\sqrt{1-a^2}\,R1\right]^2+\text{Abs}[R0-a\,R1]^2}}-\frac{a\,R1\,(R0-a\,R1)}{\sqrt{\text{Abs}[a\,R1]^2+\text{Abs}\left[\sqrt{1-a^2}\,R1\right]^2}\sqrt{\text{Abs}[h1]^2+\text{Abs}\left[\sqrt{1-a^2}\,R1\right]^2+\text{Abs}[R0-a\,R1]^2}}\right)-$$
$$T\left(-\frac{a\,R1\,(a\,R1-b\,R2)}{\sqrt{\text{Abs}[a\,R1]^2+\text{Abs}\left[\sqrt{1-a^2}\,R1\right]^2}\sqrt{\text{Abs}[h2]^2+\text{Abs}[a\,R1-b\,R2]^2+\text{Abs}\left[-\sqrt{1-a^2}\,R1+\sqrt{1-b^2}\,R2\right]^2}}+\frac{\sqrt{1-a^2}\,R1\left(-\sqrt{1-a^2}\,R1+\sqrt{1-b^2}\,R2\right)}{\sqrt{\text{Abs}[a\,R1]^2+\text{Abs}\left[\sqrt{1-a^2}\,R1\right]^2}\sqrt{\text{Abs}[h2]^2+\text{Abs}[a\,R1-b\,R2]^2+\text{Abs}\left[-\sqrt{1-a^2}\,R1+\sqrt{1-b^2}\,R2\right]^2}}\right)$$

化简 | 转换成共轭 | a 的导数 | a 的积分 | 更多.....

Out[51]=
$$T\left(\frac{a\sqrt{1-a^2}\,R1^2\,w}{\sqrt{\text{Abs}[h1]^2+\text{Abs}\left[\sqrt{1-a^2}\,R1\right]^2+\text{Abs}[R0-a\,R1]^2}\sqrt{\text{Abs}[a\,R1\,w]^2+\text{Abs}\left[\sqrt{1-a^2}\,R1\,w\right]^2}}+\frac{\sqrt{1-a^2}\,R1\,(R0-a\,R1)\,w}{\sqrt{\text{Abs}[h1]^2+\text{Abs}\left[\sqrt{1-a^2}\,R1\right]^2+\text{Abs}[R0-a\,R1]^2}\sqrt{\text{Abs}[a\,R1\,w]^2+\text{Abs}\left[\sqrt{1-a^2}\,R1\,w\right]^2}}\right)-$$
$$T\left(\frac{\sqrt{1-a^2}\,R1\,(a\,R1-b\,R2)\,w}{\sqrt{\text{Abs}[h2]^2+\text{Abs}[a\,R1-b\,R2]^2+\text{Abs}\left[-\sqrt{1-a^2}\,R1+\sqrt{1-b^2}\,R2\right]^2}\sqrt{\text{Abs}[a\,R1\,w]^2+\text{Abs}\left[\sqrt{1-a^2}\,R1\,w\right]^2}}+\frac{a\,R1\left(-\sqrt{1-a^2}\,R1+\sqrt{1-b^2}\,R2\right)w}{\sqrt{\text{Abs}[h2]^2+\text{Abs}[a\,R1-b\,R2]^2+\text{Abs}\left[-\sqrt{1-a^2}\,R1+\sqrt{1-b^2}\,R2\right]^2}\sqrt{\text{Abs}[a\,R1\,w]^2+\text{Abs}\left[\sqrt{1-a^2}\,R1\,w\right]^2}}\right)$$

Out[52]=
$$T\left(-\frac{\sqrt{1-b^2}\,R2\,(a\,R1-b\,R2)\,w}{\sqrt{\text{Abs}[h2]^2+\text{Abs}[a\,R1-b\,R2]^2+\text{Abs}\left[-\sqrt{1-a^2}\,R1+\sqrt{1-b^2}\,R2\right]^2}\sqrt{\text{Abs}[b\,R2\,w]^2+\text{Abs}\left[\sqrt{1-b^2}\,R2\,w\right]^2}}-\frac{b\,R2\left(-\sqrt{1-a^2}\,R1+\sqrt{1-b^2}\,R2\right)w}{\sqrt{\text{Abs}[h2]^2+\text{Abs}[a\,R1-b\,R2]^2+\text{Abs}\left[-\sqrt{1-a^2}\,R1+\sqrt{1-b^2}\,R2\right]^2}\sqrt{\text{Abs}[b\,R2\,w]^2+\text{Abs}\left[\sqrt{1-b^2}\,R2\,w\right]^2}}\right)$$

Out[53]=
$$T\left(-\frac{b\,R2\,(a\,R1-b\,R2)}{\sqrt{\text{Abs}[b\,R2]^2+\text{Abs}\left[\sqrt{1-b^2}\,R2\right]^2}\sqrt{\text{Abs}[h2]^2+\text{Abs}[a\,R1-b\,R2]^2+\text{Abs}\left[-\sqrt{1-a^2}\,R1+\sqrt{1-b^2}\,R2\right]^2}}+\frac{\sqrt{1-b^2}\,R2\left(-\sqrt{1-a^2}\,R1+\sqrt{1-b^2}\,R2\right)}{\sqrt{\text{Abs}[b\,R2]^2+\text{Abs}\left[\sqrt{1-b^2}\,R2\right]^2}\sqrt{\text{Abs}[h2]^2+\text{Abs}[a\,R1-b\,R2]^2+\text{Abs}\left[-\sqrt{1-a^2}\,R1+\sqrt{1-b^2}\,R2\right]^2}}\right)$$

计算得到相应的数值解，将这些得到的数值解带入到我们建立的虚拟仿真实验中就可以实现真正的在电脑上玩玩具。

5　仿真算法分析

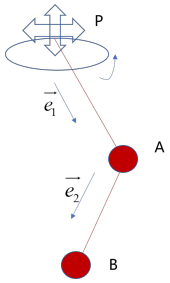
5.1　保辛算法

因为哈密顿系统的的正则方程在辛变换下形式不变，辛算法是基于哈密顿力学的基本原理而提出的保哈密顿系统的差分法，它使离散化后的差分方程保持原有的系统的辛结构，因此辛算法具有长时间的稳定性。

一般的算法，误差（步长以及计算机字长造成）会随步数指数增加。减小步长会造成迭代次数的增加。如果保证在积分每一步都作辛变换，则可以显著控制误差的积累。

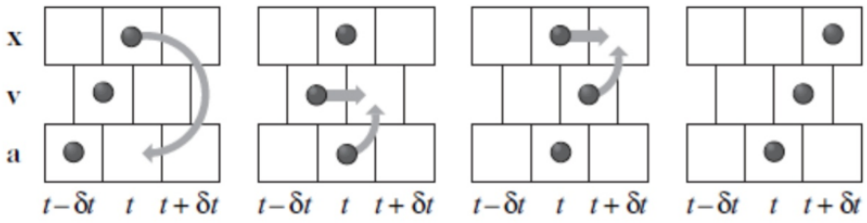
5.2　蛙跳法具体应用

根据每一时刻的位置矢量，动量来计算该时刻的受力，求得其加速度，速度，利用“蛙跳法“解微分方程求其位置变化；



因为中间球是穿在绳子上的，因此可以认为绳子上的拉力大小是一样的；此系统为哈密顿系统，我们采用保辛算法（Symplectic）迭代求解。

5.2.1 蛙跳法原理介绍



蛙跳法原理图示

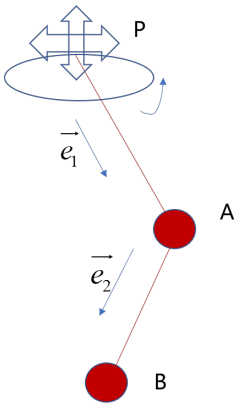
积分的起点是 t 时刻的坐标 x(t) 和 $t-\frac{1}{2}\delta t$ 时刻的速度 $v(t-\frac{1}{2}\delta t)$

第一跳；青蛙（积分计算者的目光焦点）首先站在 x(t) 的位置上，根据坐标计算出粒子受力，自然推出 t 时刻的加速度 a(t)，完成第一跳。

第二跳：根据 a(t) 算出 $t+\frac{1}{2}\delta t$ 时刻的速度。$v(t+\frac{1}{2}\delta t)=v(t-\frac{1}{2}\delta t)+a(t)\cdot\delta t$ 第二跳完成。

第三跳：根据 $t+\frac{1}{2}\delta t$ 时刻的速度计算 t 时刻的位置 $x(t+\delta t)=x(t)+v(t+\frac{1}{2}\delta t)\cdot\delta t$ 第三跳完成。而 t 时刻的速度则用，$t-\frac{1}{2}\delta t$ 和 $t+\frac{1}{2}\delta t$ 时刻的速度的平均即可。

5.2.2 具体分析



注意到，两球碰撞的情形；

我们采取数值模拟中常用的策略，考虑球具较大的弹性系数；两球碰撞过程中具有微小的形变，借此计算其受力、加速度、位移变化。

运动过程中，绳子会有一定程度上的伸长，根据伸长量和劲度系数计算两球受力；当绳长松弛时，绳子拉力为零；

5.3 建立方程

n: 代表第 n 个时间点；P：顶端球的位置；A: 中间球位置；B: 末端球位置；

$$\overrightarrow{P_nA_n} = \overrightarrow{OA_n} - \overrightarrow{OP_n}$$
$$\overrightarrow{A_nB_n} = \overrightarrow{OB_n} - \overrightarrow{OA_n}$$
$$\overrightarrow{e_1} = \frac{\overrightarrow{P_nA_n}}{|\overrightarrow{P_nA_n}|}$$
$$\overrightarrow{e_2} = \frac{\overrightarrow{A_nB_n}}{|\overrightarrow{A_nB_n}|}$$
$$\Delta L_n = |\overrightarrow{P_nA_n}| + |\overrightarrow{A_nB_n}| - L_0$$
$$f_0 = k \cdot \Delta L_n$$
$$\overrightarrow{f_{A_n}} = -f_0\overrightarrow{e_1} + f_0\overrightarrow{e_2} - \overrightarrow{G_{A_{n-1}}}$$
$$\overrightarrow{f_{B_n}} = -f_0\overrightarrow{e_2} - \overrightarrow{G_{B_{n-1}}}$$

k 劲度系数可由实验测得，绳子弹力大小

采用“蛙跳法”迭代更新各个球的位置，速度，加速度。（注意到在此体系下）

$$\overrightarrow{a_{A_n}} = \frac{\overrightarrow{f_{A_n}}}{m_{A_n}}$$

$$\overrightarrow{a_{B_n}} = \frac{\overrightarrow{f_{B_n}}}{m_{B_n}}$$

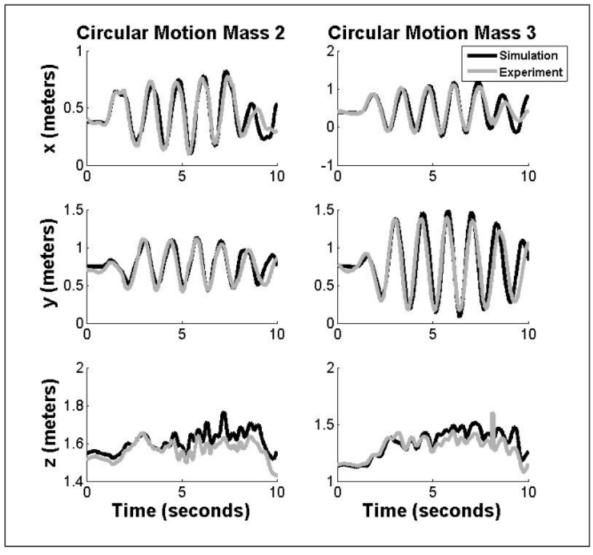
$$\overrightarrow{v_{A_{n+\frac{1}{2}}}} = \overrightarrow{v_{A_{n-\frac{1}{2}}}} + T \cdot \overrightarrow{a_{A_{n-1}}}$$

$$\overrightarrow{v_{B_{n+\frac{1}{2}}}} = \overrightarrow{v_{B_{n-\frac{1}{2}}}} + T \cdot \overrightarrow{a_{B_{n-1}}}$$

$$\overrightarrow{x_{A_{n+1}}} = \overrightarrow{x_{A_{n-1}}} + T \cdot \overrightarrow{v_{A_{n+\frac{1}{2}}}}$$

$$\overrightarrow{x_{B_{n+1}}} = \overrightarrow{x_{B_{n-1}}} + T \cdot \overrightarrow{v_{B_{n+\frac{1}{2}}}}$$

5.4 仿真误差分析



如图所示为中间球和底部球的位置坐标随时间变化的仿真与实验对比图，我们先使用两台高速摄像机拍摄从一个特定位置无初速度释放后两球的运动轨迹，使用专业图像分析软件 Tracker 追踪并量化其位置坐标随时间变化，将小球的质量、绳子长度，劲度系数、空气阻尼系数和初始位置的坐标输入到仿真程序中，对比其坐标随时间的变化。通过上图我们发现，仿真与实验数据符合较好，在较长的演化时间内 $MSE \leqslant 0.0001$

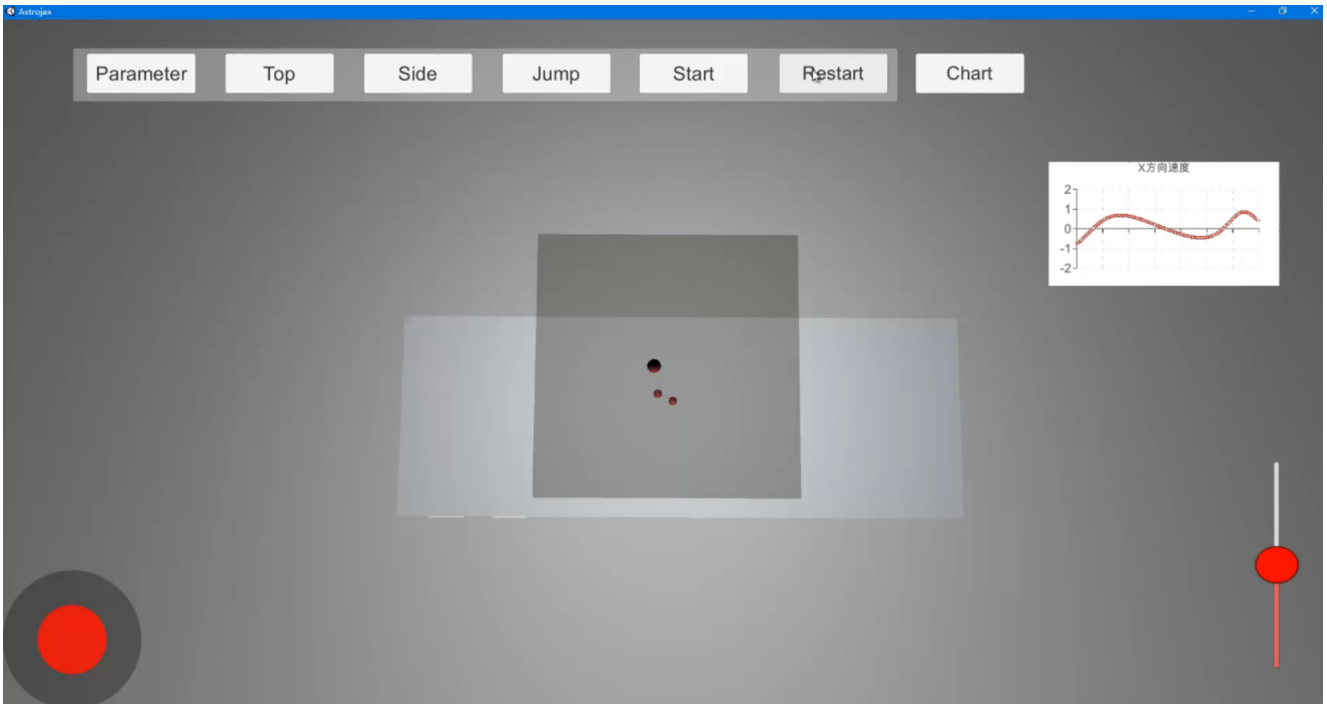
5.5 虚拟仿真优点

该仿真实验装置简便，可用手机端 APP 进行操作，在这个人手一机的时代，使用这样的手机文件可以让更多的人接触到这个在美国风靡的新型玩具。同时在正确的物理背景支撑下，所用的保辛算法又保证了实验数据测量的准确性、可靠性、对真实情况的高度拟合性，可以保证这个 APP 在运营期间容错率大大提高，保证用户的稳定体验，让用户在千百次的实验中得到稳定的数据保证。基于这些优势，预期可以成为教学成本低、操作难度小、实验结果直观可靠的教学工具，完全可以在教学期间用这样的工具

作为实际玩具的替代品，不需要老师再去带一些额外的实验教学器材来辅助教学，可以仅用一台电脑来达到更高的教学效果，具有很大的推广价值。同时又因为其较高的性价比，应当可以为使用者带来可观的经济效益。而与此同时，我们认为作品已经的成熟，但从设计思路、使用的算法等方面还有一定改进的空间，我们在今后仍会不断完善这个作品。

6 附录

6.1 windows 平台操作



我们将仿真得出的运动轨迹与实验进行对比，发现在较长时间内，我们的仿真模型与实际符合较好。

借助仿真程序，可以把很多实验搬到大家的手机上。虚拟仿真可以验证、展现出我们的理论分析。

同学可以在虚拟仿真中可以自己调节参数，探究出更多的稳定态。

如图为我们的仿真程序，左下角的红色按钮类似于我们玩手机游戏时的操纵杆，可以控制最上端球的水平移动，而右下角的红色按钮，可以控制最上端球的竖直运动；我们可以选择在不同的视角下观察系统的运动状态；我们也可以导入数据，自己设定数据条件，即，自定义玩法；右上角的曲线表示末端球在 x 方向上的速度随着时间的变化；。

6.2 C # 代码（部分）

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
using System.IO;
using System.Text;
//using System.Diagnostics;

public class BallController : MonoBehaviour
{
// 图表组件
public SpeedChart chart;

private int nball;
private int ndyn;
private float dt;
private float k;
private float damping;
private float ks;
private float m;
private float g;
private float L0;
private float L1;
private float L2;
private float dL;
private float t;
private float driver_f;
private float driver_r;
```

```
30 private float driver_a;
31 private float driver_p;
32 private float L;
33 private float Fsr;
34 private float radius;
35 private float cradius;
36 private float diameter;
37 private float cdiameter;
38 private float amp;
39 private float force_radius;
40 private Vector3[] X;
41 private Vector3[] V;
42 private Vector3[] F;
43 private Vector3 Fs2;
44 private Vector3 Fs1;
45 private Vector3 Ft2;
46 private Vector3 Ft1;
47 private Vector3[,] F_pair;
48 private Vector3[] a;
49 private Vector3 e0;
50 private Vector3 e1;
51 private Vector3 e;
52 private Vector3 dX;
53 private Vector3 origin;
54 private Vector3 CX0;
55 private Vector3 CX1;
56 private float[,] Fr;
57 private float[,] D;
58 private List<GameObject> balls;
59 private List<MoveController> ballscripts;
60 private List<GameObject> strings;
61 private List<MoveString> stringscripts;
62 public GameObject Ballpref;
63 public GameObject Stringpref;
64
65 // Start is called before the first frame update
```

```
66 void Start()
67 {
68 Initiate_params();
69
70 for (int i = 0; i < nball; i++)
71
72 MoveController ballscript = newBall.GetComponent<MoveController>();
73 balls.Add(newBall.gameObject);
74 ballscripts.Add(ballscript);
75 }
76
77 for (int i = 0; i < 2; i++)
78 {
79 GameObject newString = Instantiate(Stringpref);
80 newString.transform.localScale = new Vector3(10.0f, 10.0f, 10.0f);
81 MoveString stringscript = newString.GetComponent<MoveString>();
82 strings.Add(newString.gameObject);
83 stringscripts.Add(stringscript);
84 }
85
86 Initiate_xv();
87
88 UpdatePos();
89 }
90
91 public void Run()
92 {
93 Get_xv();
94 Distance();
95 //L0 = L;
96 Get_bond();
97
98
99 InvokeRepeating("UpdatePos", 0f, 0.01f);
100 }
101
```

```
102 void UpdatePos()
103 {
104
105 Get_xv();
106 Update_xv(nball, X, V, dt, ndyn, 1);
107 //Debug.Log("L0 = " + L0);
108 //Debug.Log("L  = " + L );
109 Set_xv();
110 Update_position();
111
112 // 更新图表
113 chart.AddData("",V[2].x);
114
115 if (Input.GetKey(KeyCode.Escape))
116 {
117 Screen.fullScreen = false;  //退出全屏
118 }
119 //按A全屏
120 if (Input.GetKey(KeyCode.A))
121 {
122 Screen.SetResolution(1920, 1080, true);
123 Screen.fullScreen = true;  //设置成全屏
124 }
125
126 }
127 {
128 origin = new Vector3(0.0f, 0.0f, 0.0f);
129 if (key == "dp")
130 {
131 damping = float.Parse(value);
132 }
133 else if (key == "l0")
134 {
135 L0 = float.Parse(value);
136 }
137 else if (key == "g")
```

```
138 {
139 g = float.Parse(value);
140 }
141 else if (key == "X0x")
142 {
143 ballscripts[0].x.x = float.Parse(value) + origin.x;
144 }
145 else if (key == "X0y")
146 {
147 ballscripts[0].x.y = float.Parse(value) + origin.y;
148 }
149 else if (key == "X0z")
150 {
151 ballscripts[0].x.z = float.Parse(value) + origin.z;
152 }
153 else if (key == "X1x")
154 {
155 ballscripts[1].x.x = float.Parse(value) + origin.x;
156 }
157 else if (key == "X1y")
158 {
159 ballscripts[1].x.y = float.Parse(value) + origin.y;
160 }
161 else if (key == "X1z")
162 {
163 ballscripts[1].x.z = float.Parse(value) + origin.z;
164 }
165 else if (key == "X2x")
166 {
167 ballscripts[2].x.x = float.Parse(value) + origin.x;
168 }
169 else if (key == "X2y")
170 {
171 ballscripts[2].x.y = float.Parse(value) + origin.y;
172 }
173 else if (key == "X2z")
```

```
174 {
175 ballscripts[2].x.z = float.Parse(value) + origin.z;
176 }
177 else if (key == "V0x")
178 {
179 ballscripts[0].v.x = float.Parse(value);
180 }
181 else if (key == "V0y")
182 {
183 ballscripts[0].v.y = float.Parse(value);
184 }
185 else if (key == "V0z")
186 {
187 ballscripts[0].v.z = float.Parse(value);
188 }
189 else if (key == "V1x")
190 {
191 ballscripts[1].v.x = float.Parse(value);
192 }
193 else if (key == "V1y")
194 {
195 ballscripts[1].v.y = float.Parse(value);
196 }
197 else if (key == "V1z")
198 {
199 ballscripts[1].v.z = float.Parse(value);
200 }
201 else if (key == "V2x")
202 {
203 ballscripts[2].v.x = float.Parse(value);
204 }
205 else if (key == "V2y")
206 {
207 ballscripts[2].v.y = float.Parse(value);
208 }
209 else if (key == "V2z")
```

```
210 {
211 ballscripts[2].v.z = float.Parse(value);
212 }
213 Get_xv();
214
215 {
216 ballscripts[0].x.y = value * 0.1f;
217 }
218 public void ChangeXZ(float xratio, float yratio)
219 {
220 float x0 = -0.2f;
221 float z0 = -0.2f;
222 float wid = 0.4f;
223 float hei = 0.4f;
224 ballscripts[0].x.x = x0 + wid * xratio;
225 ballscripts[0].x.z = z0 + hei * yratio;
226
227 }
228
229
230 public void Jump(Vector3 pos)
231 {
232 //X[0].y += 5;
233 ballscripts[0].x.y = pos.y * 0.1f;
234
235 }
236
237 //-----------------------------------------------------------------
238 void Initiate_params()
239 {
240 //-----------------------------------------------------------------
241 // Initiation
242 //
243 nball = 3;
244 ndyn = 20;
245 dt = 0.0001f;
```

```
246 radius = 0.01f;
247 cradius = radius / 5.0f;
248 diameter = radius * 2.0f;
249 cdiameter = cradius * 2.0f;
250 damping = 0.005f;
251 //driver_f = 0.1f;
252 driver_f = 1.0f;
253 //driver_r = 2.0f;
254 driver_p = 0.0f;
255 //amp = 0.1f;
256 t = 0.0f;
257 //damping = 0.000f;
258 //driver_f = 0.0f;
259 //driver_r = 0.0f;
260 k = 28000.0f;
261 ks = 28000.0f;
262 m = 31.88f;
263 m = m / 1000.0f;
264 g = 9.8f;
265 // g = g * 0.01f;
266 force_radius = 2.0f * radius;
267 L0 = 0.61f;
268 //
269 balls = new List<GameObject>();
270 ballscripts = new List<MoveController>();
271 strings = new List<GameObject>();
272 stringscripts = new List<MoveString>();
273
274 e = new Vector3();
275 origin = new Vector3();
276 dX = new Vector3();
277 e0 = new Vector3();
278 e1 = new Vector3();
279 CX0 = new Vector3();
280 CX1 = new Vector3();
281 Fs2 = new Vector3();
```

```
282 Fs1 = new Vector3();
283 Ft2 = new Vector3();
284 Ft1 = new Vector3();
285 X = new Vector3[nball];
286 V = new Vector3[nball];
287 a = new Vector3[nball];
288 F = new Vector3[nball];
289 F_pair = new Vector3[nball, nball];
290 D = new float[nball, nball];
291 Fr = new float[nball, nball];
292 }
293
294 }
295 void Get_xv()
296 {
297 for (int i = 0; i < nball; i++)
298 {
299 X[i] = ballscripts[i].x;
300 V[i] = ballscripts[i].v;
301 }
302 }
303 void Set_xv()
304 {
305 for (int i = 0; i < nball; i++)
306 {
307 ballscripts[i].x = X[i];
308 ballscripts[i].v = V[i];
309 }
310 }
311 void Update_position()
312 {
313 for (int i = 0; i < nball; i++)
314 {
315 balls[i].transform.position = ballscripts[i].x;
316 }
317 }
```

```
318 void Distance()
319 {
320 for (int i = 0; i < nball; i++)
321 {
322 for (int j = i+1; j < nball; j++)
323 {
324 if (i == j)
325 {
326 D[i, j] = 0.0f;
327 }
328 else
329 {
330 D[i, j] = (X[i] - X[j]).magnitude;
331 D[j, i] = D[i, j];
332 }
333 }
334 }
335 L1 = D[0, 1];
336 L2 = D[1, 2];
337 L = L1 + L2;
338 }
339
340 {
341 for (int i = 0; i < nball - 1; i++)
342 {
343 for (int j = i + 1; j < nball; j++)
344 {
345 if (D[i, j] < force_radius)
346 {
347 float dr = force_radius - D[i, j];
348 Fr[i, j] = k * dr;
349 }
350 else
351 {
352 Fr[i, j] = 0.0f;
353 }
```

```
354 e = (X[i] - X[j]) / D[i, j];
355 // The force act on i
356 F_pair[i, j] = e * Fr[i, j];
357 // The force act on j
358 F_pair[j, i] = -F_pair[i, j];
359 }
360 }
361
362 for (int i = 0; i < N; i++)
363 {
364 V[i].x = V[i].x + a[i].x * dt / 2.0f;
365 V[i].y = V[i].y + a[i].y * dt / 2.0f;
366 V[i].z = V[i].z + a[i].z * dt / 2.0f;
367 }
368 //------------------------------------------------------------------
369 for (int i = 0; i < N; i++)
370 {
371 t = t + dt;
372 if (i == -1)
373 {
374 if (driver_f > 0.0f) {
375 //driver_a = driver_f * dt;
376 driver_a = driver_f * t;
377 dX = new Vector3(0.0f, 0.0f, 0.0f);
378 //------------------------------------------------------------------
379
380 // move along y axis
381 //
382 // dX.y = amp * (float)Math.Cos(driver_a + driver_p);
383 dX.y = amp * (float)Math.Cos(driver_a);
384 UnityEngine.Debug.Log(driver_f);
385 UnityEngine.Debug.Log(dt);
386 //------------------------------------------------------------------
387 // X[i] = origin + dX;
388 }
389 }
```

```
390 else
391 {
392 X[i].x = X[i].x + V[i].x * dt;
393 X[i].y = X[i].y + V[i].y * dt;
394 X[i].z = X[i].z + V[i].z * dt;
395 }
396 }
397 //------------------------------------------------------------------------
398 Cal_force();
399 //------------------------------------------------------------------------
400 for (int i = 0; i < N; i++)
401 {
402 V[i].x = V[i].x + a[i].x * dt / 2.0f;
403 V[i].y = V[i].y + a[i].y * dt / 2.0f;
404 V[i].z = V[i].z + a[i].z * dt / 2.0f;
405 //------------------------------------------------------------------------
406 }
407 Get_bond();
408 }
409 }
410
411
412 // Update is called once per frame
413 void Update()
414 {
415
416
417 }
418
419
420 }
```

6.3 Python 代码 (部分)

```
1 import numpy as np
2 import matplotlib as mpl
```

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

#定义基本物理量，M表示质量，T为时间间隔;
#x为位置，v为速度，g为重力系数，k为绳子张力系数
Ma = 0.1
Mb = 0.1
g = 9.8
T = 0.0001
k = 1000
x0 = np.asarray([0.0 , 0.0, 0.0])
xa = np.asarray([0.5 , 0, 0])
xb = np.asarray([1.0 , 0, 0])
va = np.asarray([0.0 , 0.0, 0.0])
vb = np.asarray([0.0 , 0.0, 0.0])
l0a = (np.sum((xa - x0) ** 2)) ** 0.5
lab = (np.sum((xa - xb) ** 2)) ** 0.5
l0 = l0a + lab
print(x0,xa,xb,l0a,lab)
def fun(x0,xa,xb,va,vb):
#确定受力大小
disab = (np.sum((xa - xb) ** 2)) ** 0.5
dis0a = (np.sum((xa - x0) ** 2)) ** 0.5
dis0 = disab + dis0aC
# if disab - lab > 0:
#     fab = k * (disab - lab) #
# else :
#     fab = 0

# if dis0a - l0a > 0:
#     f0a = k * (dis0a - l0a)
# else:
#     f0a = 0

if dis0 - l0 > 0:
f0 = k * (dis0 - l0)
```

```
39  else:
40  f0 = 0
41  #确定受力方向并归一化
42  vec0a = xa - x0
43  vecba = xa - xb
44  vec0a = vec0a / (np.sum(vec0a ** 2)) ** 0.5
45  vecba = vecba / (np.sum(vecba ** 2)) ** 0.5
46  fa = -vec0a * f0 - vecba * f0 +  np.asarray([0, 0, -Ma * g])
47  fb = vecba * f0 + np.asarray([0, 0, -Mb * g])
48  #确定加速度
49  a_a = fa / Ma
50  a_b = fb / Mb
51  #执行移动模块并
52  va = va + a_a * T
53  vb = vb + a_b * T
54  xa = xa + va * T
55  xb = xb + vb * T
56
57  return xa,xb,va,vb
58
59  num = 0
60  x1 = []
61  y1 = []
62  z1 = []
63  x2 = []
64  y2 = []
65  z2 = []
66  N = 100
67
68  Tx0 = []
69  theta = 0
70  for i in range(N):
71  x = 0.0*np.asarray([np.sin(theta),np.cos(theta),0])
72  Tx0.append(x)
73  theta = theta + 0.01
74  while num < N:
```

```
xa,xb,va,vb = fun(Tx0[num],xa,xb,va,vb)
x1.append(xa[0])
y1.append(xa[1])
z1.append(xa[2])
x2.append(xb[0])
y2.append(xb[1])
z2.append(xb[2])
num = num + 1
vec0a = xa - x0
vecba = xa - xb
t = (np.sum(vec0a ** 2)) ** 0.5 + (np.sum(vecba ** 2)) ** 0.5
print(t)
# mpl.rcParams['legend.fontsize'] = 10
```