狭义相对论相关问题的可视化

魏泽林　张明轩　唐承哲
山东大学物理学院

2021 年 12 月 27 日

摘要

本文从狭义相对论的基本假设出发，推导出洛伦兹变换，阐明狭义相对论的基本时空观，并利用上述结论求解相对论完全非弹性碰撞和静止粒子的衰变，最后运用 Python、VPython 等软件进行了可视化处理，使物理图像更加直观。

关键词：　狭义相对论，可视化，Python，VPython

Abstract

From the basic postulations of the special relativity, this article gives the Lorentz Transformation, explaining the basic principles of time and space.The following part uses the conclusions and equations to solve the complete inelastic collision and rest particle decay.Finally, we use Python and VPython to make the physical problems visualised.

Keywords:　special relativistic,visualisation,Python,VPython

目录

1 基本原理

1905 年，爱因斯坦在著名的《论运动物体的电动力学》一文中首次概括出以下两条原理：

(1) 狭义相对性原理：任何真实的物理规律在所有惯性系中应形式不变。

(2) 光速不变原理：任意一个惯性系中的观测者所测得的真空中光速恒为 c。

我们用 $P(t,x,y,z)$ 表示一个事件，对于任意两个事件 $P_0(t_0,x_0,y_0,z_0)$ 和 $P(t,x,y,z)$ 时间间隔和空间间隔分别为

$$\Delta t = t - t_0$$
$$\Delta l = \left[(x-x_0)^2+(y-y_0)^2+(z-z_0)^2\right]^{\frac{1}{2}} \tag{1}$$

我们构造时空间隔 s，如果取事件 $P_0=(0,0,0,0)$, 则令 $s^2=c^2t^2-(x^2+y^2+z^2)$。这里定义的 s 称作时空间隔。经过证明，对相互作匀速运动的两个观察者 S 和 S', 测量两事件的时间间隔和空间间隔一般不相同，而对时空间隔却相同，即 $s^2=s'^2$。

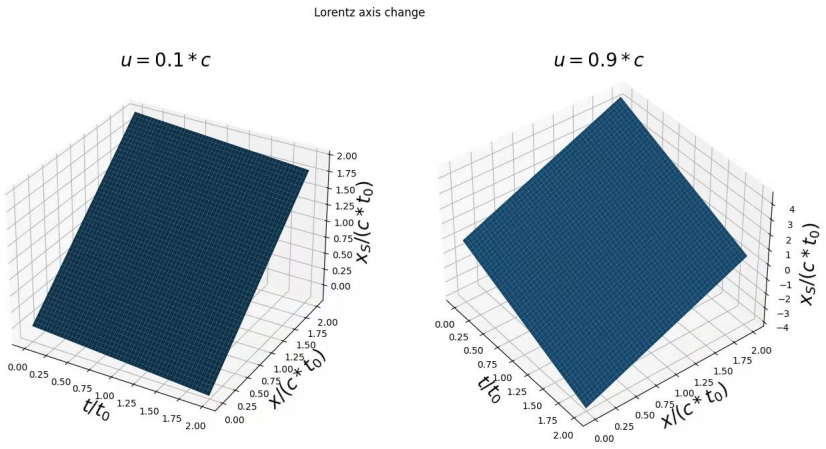

图 1: 此为特殊情况下的洛伦兹坐标变换平面

对于任意两个相邻事件,如果它们的时空间隔相差无穷小,即 $P_0(t_0, x_0, y_0, z_0)$ 和 $P(t_0 + dt, x_0 + dx, y_0 + dy, z_0 + dz)$，一般地

$$ds^2 = c^2dt^2 - (dx^2 + dy^2 + dz^2) = inv. \tag{2}$$

以上即是爱因斯坦两个基本原理的数学表述。

2 洛伦兹变换

2.1 洛伦兹坐标变换

设 S' 系相对于 S 系的运动速度为 u。

首先研究 y 和 z 坐标的变换。为此考虑 S 系的 y 平面，即 $y = a$。由空间的均匀性，知道 S 系的平面在 S' 系中也是平面，即 $y' = a'$。在一般情况下，平面的两个坐标之间的关系应该是 $a' = K(u)a$，根据狭义相对性原理，交换 S 和 S' 的记号，还应该有 $a = K(u)a'$，故 $K^2(u) = 1$。又因为 y 和 y' 的方向相同，故 a 和 a' 的符号相同。最后得 y 坐标的关系为 $y' = y$，同理求得 z 坐标的变换关系 $z' = z$。接下来根据间隔不变性求 x 和 t 的变换关系。根据以上两式时空间隔变为

$$c^2t'^2 - x'^2 = c^2t^2 - x^2 \tag{3}$$

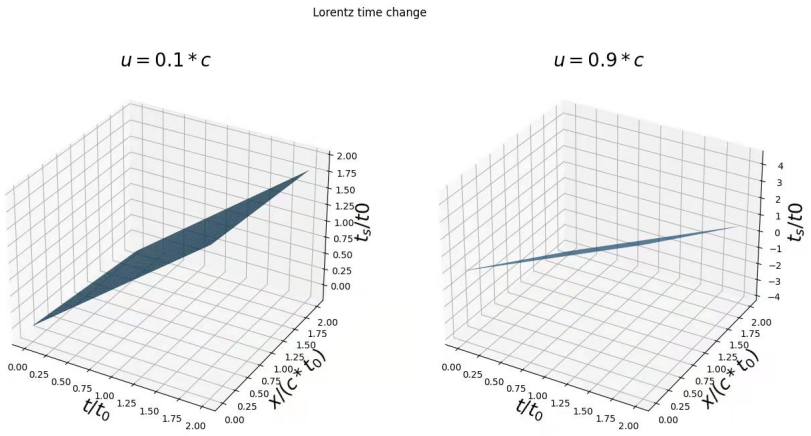

图 2: 此为特殊情况下的洛伦兹时间变换平面

因为线性变换保持二次齐式不变，上式说明 x' 和 t' 只能是 x，t 的线性函数，假设其形式为

$$t' = a_{00}t + a_{01}x \tag{4}$$

$$x' = a_{10}t + a_{11}x \tag{5}$$

其中的变换系数均为常数。将它们代入 (3)，得

$$x^2 - c^2t^2 = (c^2a_{01}^2)x^2 + (a_{10}^2 - c^2a_{00}^2)t^2 + (a_{10}a_{11} - c^2a_{01}a_{00})xt \tag{6}$$

由于 x，t 的任意性，比较两边系数得到

$$c^2a_{01}^2 + a_{11}^2 = 1 \tag{7}$$

$$a_{10}a_{11} - c^2a_{01}a_{00} = 0 \tag{8}$$

$$a_{10}^2 - c^2a_{00}^2 = -c^2 \tag{9}$$

另外，在 S 系中测得 S' 系的原点 $x' = 0$ 以速度 u 运动，此原点在 S 系中的坐标为 $x = vt$，代入 (5) 得

$$a_{10} = -ua_{11} \tag{10}$$

将 (10) 代入 (8) 和 (9) 消去 a_{10}，利用符号 $\beta = \frac{v}{c}$，可得

$$\beta a_{11}^2 - c a_{01} a_{00} = 0 \tag{11}$$

$$\beta^2 a_{11}^2 - a_{00}^2 = -1 \tag{12}$$

由 (7)，(11) 和 (12) 可得

$$a_{00} = \pm\sqrt{\frac{1}{1-\beta^2}} \equiv \pm\gamma, a_{11} = \pm\gamma \tag{13}$$

如果要求变换不改变时间的方向，则 $a_{00} > 0$。如果还要求变换不改变 x 轴的正方向，则 $a_{11} > 0$。由此可解得另外两个变换系数

$$a_{10} = -\gamma u, \quad a_{01} = -\frac{\gamma u}{c^2} \tag{14}$$

综合上面的结果，将系数写成矩阵形式

$$A(u) = (a_{\mu\nu}) = \begin{bmatrix} \gamma & -\frac{\gamma u}{c^2} & 0 & 0 \\ -\gamma u & \gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

式中的下标 ν，$\mu = 0\ 1\ 2\ 3$ 是矩阵的行和列的标记，对应于坐标 $t\ x\ y\ z$，并采用了惯用符号

$$\gamma = \frac{1}{\sqrt{1-\beta^2}}, \beta = \frac{u}{c} \tag{16}$$

于是有

$$\begin{aligned} t' &= \gamma(t - \frac{u}{c^2}x) \\ x' &= \gamma(x - ut) \\ y' &= y \\ z' &= z \end{aligned} \tag{17}$$

这就是洛伦兹变换。

已知事件在 S 系的时空坐标，通过 (15) 式就可求得事件在 S' 系的时空坐标。反过来，已知事件在 S' 系的时空坐标，想要求其在 S 系的坐标只需求 (17) 式的逆变换即可。

洛伦兹变换的重要意义是否定了绝对时间和绝对空间的观念，时间和空间不再是彼此独立无关的，它们通过 (17) 式密切联系着，并且还和惯性系的具体特性有关。

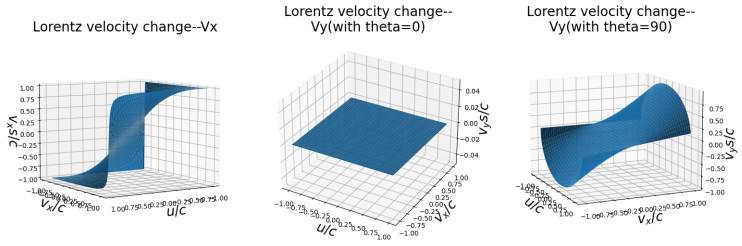

图 3: 此为速度分量变换曲面

2.2 洛伦兹速度变换

设运动的质点相对于 S 系和 S' 系的速度分别是 (v_x, v_y, v_z) 和 (v'_x, v'_y, v'_z)，将 (17) 式后三式求得的微分 dx', dy', dz' 除以第一式的微分 dt'，稍加整理后即得

$$\begin{aligned} v'_x &= \frac{v_x - u}{1 - \frac{v_x u}{c^2}} \\ v'_y &= \frac{v_y}{\gamma(1 - \frac{v_x u}{c^2})} \\ v'_z &= \frac{v_z}{\gamma(1 - \frac{v_x u}{c^2})} \end{aligned} \tag{18}$$

上式即为洛伦兹速度变换。

下面给出洛伦兹速度变换在极坐标下的表达式。设质点的速度在 S 系和 S' 系中的速度分别为 (v, θ) 和 (v', θ')，其中 u u' 是速度大小，θ θ' 是速度与 $x(x')$ 轴的夹角，即有

$$\begin{aligned} v &= \sqrt{v_x^2 + v_y^2 + v_z^2}, \quad cos\theta = \frac{v_x}{v} \\ v' &= \sqrt{v_x'^2 + v_y'^2 + v_z'^2}, \quad cos\theta' = \frac{v'_x}{v'} \end{aligned} \tag{19}$$

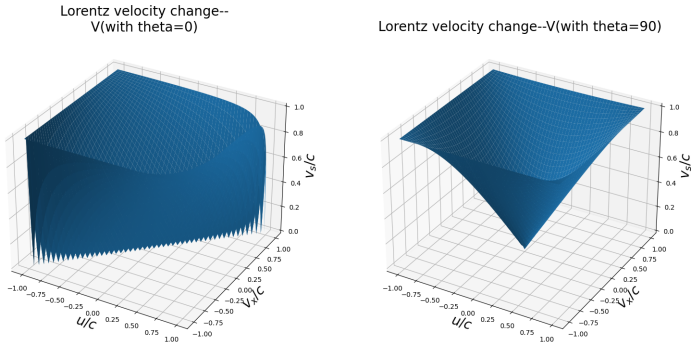

图 4: 此为速度大小变换曲面

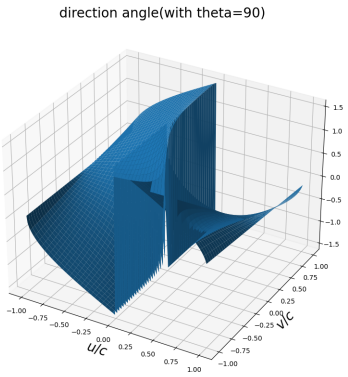

图 5: 此图为极坐标下 $\theta = 90°$ 方向角变换曲面，因为 $\theta = 0$ 时结论是平凡的，故不再讨论

由洛伦兹速度变换及其逆变换可以得到洛伦兹速度变换的极坐标表示为

$$v' = c(1 - \frac{(1-\frac{v^2}{c^2})(1-\frac{u^2}{c^2})}{(1-\frac{uvcos\theta}{c^2})^2})^{\frac{1}{2}}$$
$$cot\theta' = \gamma(cot\theta - \frac{u}{v}csc\theta) \tag{20}$$

[讨论](1) 当 $\theta = 0$ 时，S 系中的速度只有 x 分量，v_y 和 v'_y 恒为零，v'_y 是一个水平面。

(2) 当 $\theta = 0$ 时，变换后的速度 v' 如同一个“方边扁嘴漏斗”，当 $v = u$ 时，可得 $v' = 0$，这就是“漏斗”的“底线”。不论 $v = c$，还是 $u = c$，都可得 $v' = c$，这就是“漏斗”的“方边”。

(3) 当 $\theta = 0$ 时，当 $v > u$ 时，$\theta' = 0$，v' 与 x 轴正向相同；当 $v < u$ 时，$\theta' = \pm\pi$，v' 与 x 轴正向相反。

(4) 当 $\theta = \frac{\pi}{2}$ 时，S 系中的速度只有 y 分量，v_x 为零。在 u 一定的情况下，v'_y 随 v_y 直线增加，在 v_y 一定的情况下，v'_y 随 u 按椭圆规律变化。

(5) 当 $\theta = \frac{\pi}{2}$ 时，变换后的速度 v' 如同一个“方边尖嘴漏斗”，当 $v = u = 0$ 时，可得 $v' = 0$，这就是“漏斗”的“底”。只要 $v = c$，或 $u = c$，都 $v' = c$，这就是“漏斗”的“方边”。

(6) 当 $\theta = \frac{\pi}{2}$ 时，如果 $v > 0$，随着 u 从 $-c$ 向 c 变化，速度的方向角 θ' 正方向增加；如果 $v < 0$，表示质点做 $v > 0$，$\theta = \frac{-\pi}{2}$ 的运动，随着 u 从 $-c$ 向 c 变化，速度的方向角 θ' 负方向增加。如果 $v = 0, \theta = \frac{\pi}{2}$ 是没有意义的。

三分量速度并非协变量，为什么我们还要求解它的洛伦兹变换？这是因为在速度的坐标系变换中，我们可以根据角度的变换，推导出相对论几何并非欧几里德几何，而是罗巴切夫斯基几何，具体分析请看下面。

如 (6) 所示，由三个速度 v v' u 构成的三角形 OAB 满足洛伦兹变换式 (20)。因为任意的三角形都是两个直角三角形的组合，我们只需讨论直角三角形的内角和。在 (20) 式中取 $\theta = \frac{\pi}{2}$ 则

$$cot\alpha = -cot\theta' = \frac{u}{v\sqrt{1-\frac{u^2}{c^2}}} \tag{21}$$

由对称性又得

$$cot\beta = \frac{v}{u\sqrt{1-\frac{v^2}{c^2}}} \tag{22}$$

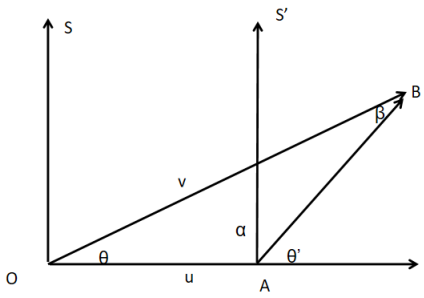

图 6: 参考图

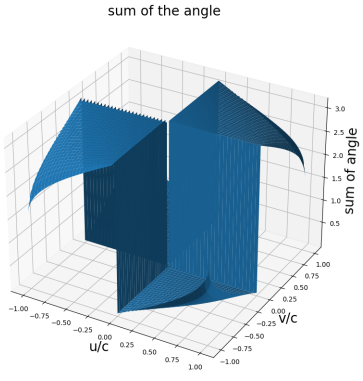

图 7: 此图为两个坐标系下速度变换三角形内角和

```
alpha=np.arctan(1/(ratio1/(ratio2*np.sqrt(1-ratio1**2))))  #变换角alpha
beta=np.arctan(1/(ratio2/(ratio1*np.sqrt(1-ratio2**2))))  #变换角beta
angle=alpha+beta+np.pi/2  #变换三角形内角和
```

图 8: 此为实现内角和的主要代码

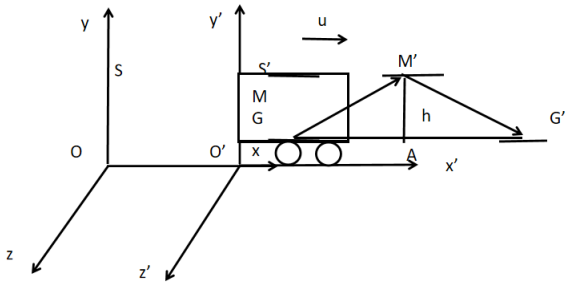

图 9: 参考图

可以证明 $\alpha+\beta+\theta<\pi$，根据几何知识我们可以知道，相对论速度为罗巴切夫斯基几何。

根据微分几何相关知识，我们求得高斯曲率为 $K=-\frac{1}{c^2}$。所以，相对论速度空间是常曲率的双曲空间，曲率半径正好为真空中的光速 c。

总之，相对论速度并不是欧几里德空间的几何量，因此我们不能用欧几里德几何的规律来理解或限制它。存在一个极限值在非欧空间完全是允许的，尽管欧几里德空间不存在这样的几何量。

3 相对性原理

3.1 时间间隔的相对性

如图 9, 一车厢在地面上的铁轨上以速度 $\mathbf{u}$ 作匀速直线运动，车厢高为 h，顶上装有反光镜 M，其正下方装有光信号的发射——接收器 G。在地面上建立坐标系 S,x 方向与车厢运动方向相同，y 方向指向车厢高度的方向，z 方向与两者垂直。在车厢上建立坐标系 S',S' 中的 $x'y'z'$ 轴与 S 中的 xyz 轴平行同向。G 发射光信号，然后接收光信号。在 S' 系中观察：光沿着竖直方向传播，反射后仍沿竖直方向回到 G，因此 G 发射和接收光信号这两件事是在同一地点发生，时间间隔为

$$\Delta t'=\frac{2h}{c} \tag{23}$$

在 S 系中观察：因为 y 和 y' 之间没有相对运动，车厢高不变。由于车厢以速度 $\mathbf{u}$ 沿 x 轴正向运动，在 G 发射和接收光信号的时间间隔里，G 已经运动了一段距离，因此，发射和接收不在同一地点发生，光的传播路径是 $G \to M' \to G'$。设经过的时间间隔为 Δt，由于 u 钢塑不变，所以

$$GM' = M'G' = c\Delta t/2 \tag{24}$$

又由于

$$GA = AG' = u\Delta t/2 \tag{25}$$

根据勾股定理可得

$$(c\Delta t/2)^2 = (u\Delta t/2)^2 + h^2 \tag{26}$$

解得

$$\Delta t = \frac{2h}{\sqrt{c^2 - u^2}} \tag{27}$$

利用 (27) 式可得

$$\Delta t = \frac{\Delta t'}{\sqrt{1 - u^2/c^2}} \tag{28}$$

其中，$\Delta t'$ 是在 S' 系中观察到的两件事情的时间间隔，由于两件事是在同一地点发生的，这样的时间间隔称为本征时或固有时；Δt 是 S 系中观察到的两件事情的时间间隔，当两件事情发生在运动系 S' 中的同一地点是 i、，在 S 系中就不在同一地点，Δt 是运动时。由此可见：在所有的时间间隔中，本征时最短，其他参考系中测得的同样两件事情的时间间隔都比本征时长。

当然，我们利用洛伦兹变换可以更加方便的导出时间的延缓效应，这个例子能够更加直观地向我们解释为何会有时间延缓。

时间膨胀是一种相对效应，在 S' 中的观察者看到 S 系中的也变慢了。这是因为在 S' 系看，S 系以速度 $-\mathbf{u}$ 向相反的方向运动。B 钟测量 A 钟的运动时间 $\Delta t'$, 表示运动时；运动着的 A 钟相对 S 系时静止的，表示本征时 Δt，时间间隔公式就要变成

$$\Delta t' = \frac{\Delta t}{\sqrt{1 - u^2/c^2}} \tag{29}$$

就是说，在 S' 系中观察 S 系的运动时，要用上式，其中 Δt 是固有时，$\Delta t'$ 是运动时。

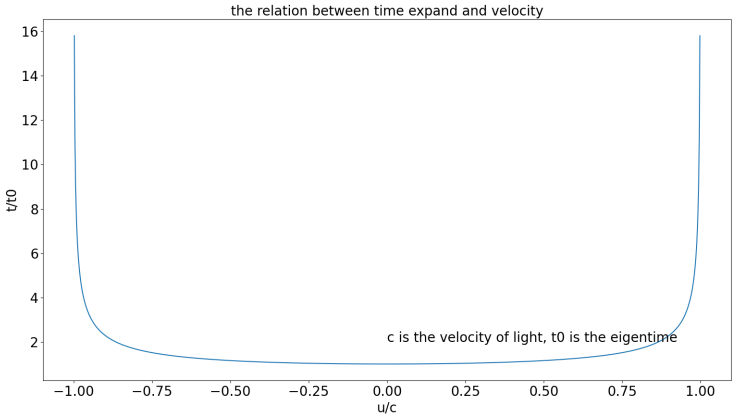

图 10: 时间膨胀曲线

同一个钟既能指示固有时，也能指示运动时，要依对象而定。当一个钟表示本系统中静止物体的同时，就是本征时；表示运动物体的时间时，就是运动时。不论在哪个参考系中观察，运动时 τ 和本征时 τ_0 之间的关系为

$$\tau = \frac{\tau_0}{\sqrt{1 - u^2/c^2}} \tag{30}$$

在 S 系中观察，本征时为 $\tau_0 = \Delta t'$，运动时为 $\tau = \Delta t$；在 S' 系中观察，本征时为 $\tau_0 = \Delta t$ 运动时为 $\tau = \Delta t'$。只有在低速运动的情况下，才有

$$\tau \approx \tau_0 \tag{31}$$

说明两个系统的钟是同步的。因此，日常生活中的钟，不论运动还是静止，都是同步的。

3.2 空间间隔的相对性

如图 11, 在 S 系中，设 x 轴上有一个固定点 P，其坐标为 x_1。在 t 时刻物块的 B 端经过 P 点，经过 Δt 时间之后，即在 $t + \Delta t$，时间之后物块 A 端也经过 P 点，即

$$x_A(t + \Delta t) = x_1 \tag{32}$$

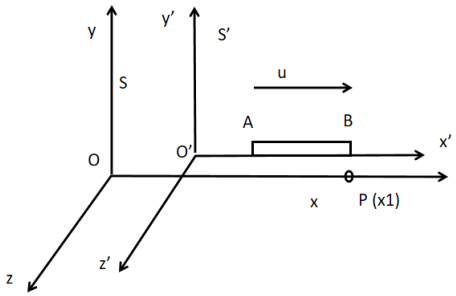

图 11: 参考图

此时 B 端的坐标为

$$\sqrt{x_B}(t+\Delta t) = x_1 + u\Delta t \tag{33}$$

因此，在 S 系中测得 AB 的长度为

$$\Delta l = x_B(t+\Delta t) - x_A(t+\Delta t) = u\Delta t \tag{34}$$

其中，Δt 是物块两端 A 和 B 相继通过 P 点这两件事情之间的时间间隔，由于 P 点在 S 系中是固定的，因此 Δt 是固有时。

在 S' 中观察，物块 AB 静止，而 S 以速度 $-u$ 沿 x' 轴反方向运动，P 点相继通过 AB 两端。设这两事件的时间间隔为 $\Delta t'$, 测得物块的长度为

$$\Delta l' = u\Delta t' \tag{35}$$

因此

$$\frac{\Delta l}{\Delta l'} = \frac{\Delta t}{\Delta t'} \tag{36}$$

P 点是 S 系中的固定点，在 S' 系中，A 和 B 分别通过 P 点这两事件不是在同一地点发生的，因此 $\Delta t'$ 不是固有时而是运动时。

将时间延缓公式运用到 (36) 可得长度公式

$$\Delta l = \Delta l'\sqrt{1-(\frac{u^2}{c^2})} \tag{37}$$

$\Delta l'$ 是观察者在 S' 系中测得的 AB 之长，由于 AB 相对观察者静止，因此是本征长度或固有长度；Δl 是在 S 系中所测的 AB 运动时的长度，称为运

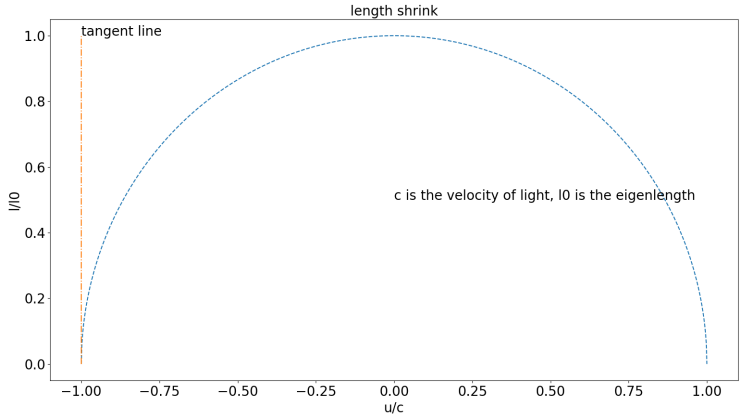

图 12: 长度收缩曲线

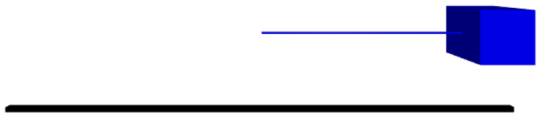

图 13: 图中为低速速度下小车的长度

动长度。从公式可见本征长度最长，其他参考系中测得的运动长度都比本征值短，这种效应称为长度收缩。

我们运用 VPython 进行了动画模拟。

4 光速极值原理

4.1 基本原理

光速不变原理：任意一个惯性系中的观测者所测得的真空中光速恒为 c。

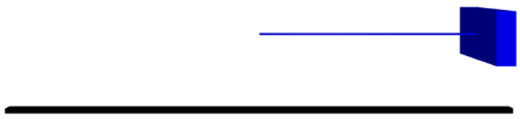

图 14: 图中为高速运动下小车长度，可以看到明显发生收缩

```
#计算长度收缩系数
length_shorten_Factors = (1 / (sqrt(1 - (car.v.x**2 / c**2))),
      1 / (sqrt(1 - (car.v.y**2 / c**2))),
      1 / (sqrt(1 - (car.v.z**2 / c**2))))

#计算小车收缩后尺寸
car.size = vector(car.size.x / length_shorten_Factors[0],
      car.size.y / length_shorten_Factors[1],
      car.size.z / length_shorten_Factors[2])

t=0 # 设定开始时间
dt= 1e-11  # 设定时间步长

while  car.pos.x <=4.5 :  #用物体x方向最大坐标极限设定分析时间

    rate(1000)
    car.pos = car.pos + car.v*dt  #计算小车位置更新
    t = t + dt   #时间迭代
```

图 15: 这是我们实现动画模拟的主要循环

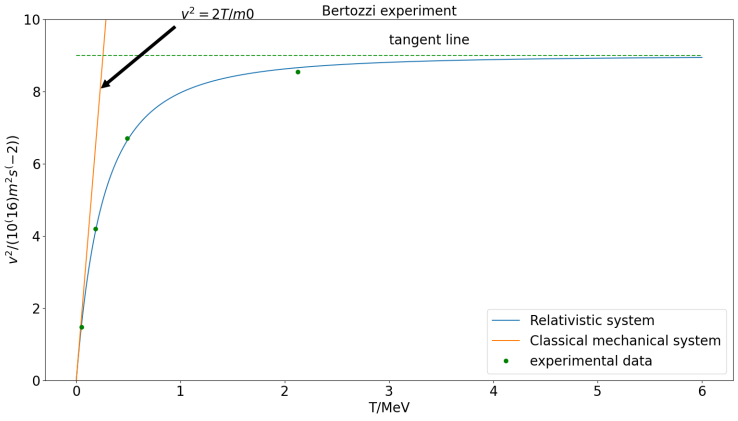

图 16: 图中斜线为经典结果，曲线为相对论条件下的结果，点为实验数据，可见实验数据和相对论符合地很好

4.2 贝托齐实验

1962 年，贝托齐用实验证实了这一结论。如下图所示，电子由静电加速器加速后进入一无电场区域，然后打到铝靶上。电子通过无电场区域的时间可以由示波器测出，因而可算出电子的速率。

电子的动能等于电子电量与加速电压的乘积。根据经典力学，电子的速率的平方用动能表示为

$$v^2 = 2T/m_0 \tag{38}$$

电子的相对论动能为

$$T = mc^2 - m_0c^2 \tag{39}$$

根据质 $-$ 速关系 (47) 式，可得方程

$$T = \frac{m_0c^2}{\sqrt{1 - v^2/c^2}} - m_0c^2 \tag{40}$$

解得

$$v^2 = c^2\left[1 - \frac{(m_0c^2)^2}{(T + m_0c^2)^2}\right] = c^2\frac{(T + 2m_0c^2)T}{(T + m_0c^2)^2} \tag{41}$$

在 $m_0c^2 >> T$ 的情况下，由上式可得非相对论速度公式。

5 其他物理量的关系

5.1 质量与速度

这里我们讨论一个静止质量为 m_0 的粒子，以速率 v 运动时，其运动质量和速度的关系。

首先选定一个静止的 S 系和一个运动的 S' 系，S' 系与粒子具有相同的运动速度 v。设在静止系中我们观察到粒子移动的距离为 dl，记录的运动时间为 dt, 那么有

$$v = \frac{dl}{dt} \tag{42}$$

依照经典的观点，在 S' 系中也我们认为粒子移动的距离也是 dl，但记录的运动时间变为 dt', 这时有

$$u = \frac{dl}{dt'} \tag{43}$$

根据之前的洛伦兹变换我们知道

$$dt' = \sqrt{1 - v^2/c^2}dt \tag{44}$$

据此推知

$$u = \frac{1}{\sqrt{1 - v^2/c^2}}v \tag{45}$$

不难发现 u 是可以达到无穷的，我们给 v 取名为常速度，u 取名为经典速度 (经典理论中我们认为速度是没有上限的)。这时我们可以将动量定义为

$$p = m_0 u = m_0 \frac{1}{\sqrt{1 - v^2/c^2}}v \tag{46}$$

为了产生标准的动量表达式我们需要分配分母，我们知道分母分配给 v 会产生经典速度 u，这是不符合实际的一个速度，所以我们将其分配给质量 m 产生一个 m'，即

$$m' = m_0 \frac{1}{\sqrt{1 - v^2/c^2}} \tag{47}$$

m' 称为相对论质量，这就是该粒子运动质量和速度的关系

5.2 能量与动量

在合外力 F 的作用下，静止质量为 m_0 的物体，速度由零变为 v, 合外力做的功为

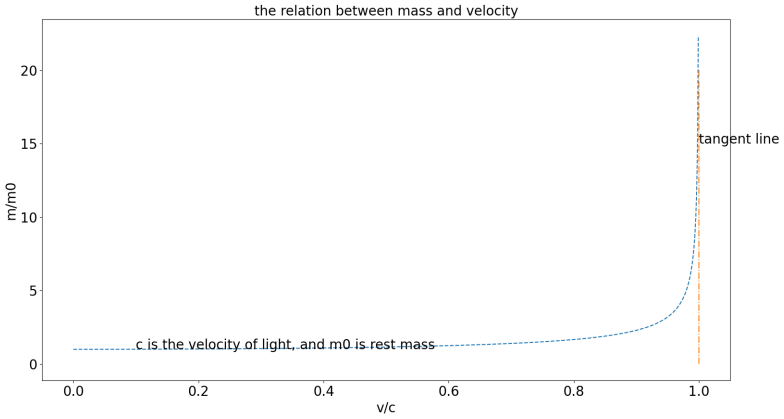

图 17: 相对论条件下速度与质量关系

$$A = \int_0^v F\,dr = \int_0^v \frac{d(mv)}{dt}\,dr = \int_0^v v\,d(mv) = \int_0^v (v^2\,dm + mv\,dv) \tag{48}$$

对质 $-$ 速关系求微分得

$$dm = d\left[\frac{m_0}{(1-v^2/c^2)^{1/2}}\right] = \frac{m_0 v dv}{c^2(1-v^2/c^2)^{3/2}} = \frac{mvdv}{c^2(1-v^2/c^2)} \tag{49}$$

由此得

$$mvdv = (c^2 - v^2)dm \tag{50}$$

带入功的公式得

$$A = \int_{m_0}^{m} [v^2\,dm + (c^2 - v^2)\,dm] = \int_{m_0}^{m} c^2\,dm = mc^2 - m_0c^2 \tag{51}$$

根据质点的动能定理：合外力所做的功等于动能的增量。物体静止时动能为零，故合外力所做的功全部转化为物体的动能。因此在相对论中，速度为 v 的物体的动能为

$$T = mc^2 - m_0c^2 = \frac{m_0c^2}{\sqrt{1-v^2/c^2}} - m_0c^2 \tag{52}$$

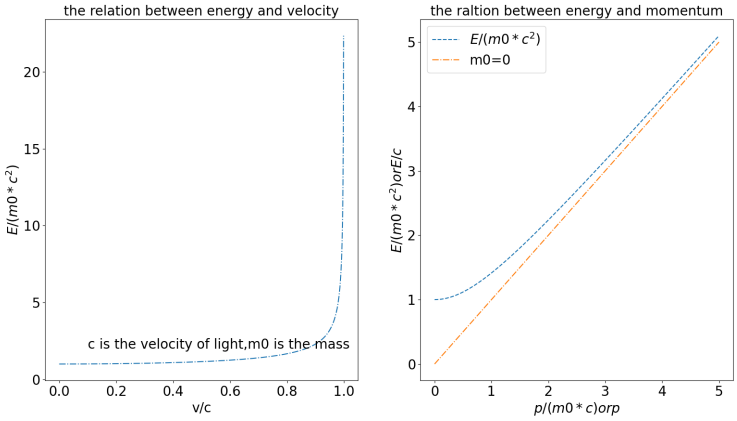

图 18: 左图为能量速度关系，右图为能量动量关系

当 $v << c$ 时，根据公式 $(1+x)^n \approx 1+nx$，运动物体的质量为

$$m = \frac{m_0}{\sqrt{1-v^2/c^2}} = m_0(1-v^2/c^2)^{-1/2} \approx m_0(1+\frac{v^2}{2c^2}) \tag{53}$$

因此

$$T \approx \frac{1}{2}m_0v^2 \tag{54}$$

可见，在低速情况下，相对论力学过渡到经典力学。

在相对论公式 (51) 中，m_0 时物体的静止质量，m 时物体的运动质量。物体的静止能量为

$$E_0 = m_0c^2 \tag{55}$$

物体的总能量为

$$E = mc^2 \tag{56}$$

这就是质 $-$ 能方程，是相对论独有的，没有经典项对应。由此可知：能量守恒必然导致质量守恒，反之，质量守恒也将导致能量守恒。在相对论中，质量守恒和能量守恒是统一的，能量蕴含在质量之中。

利用 (55) 式和 (56) 式以及质 $-$ 速关系可得

$$E^2 - {E_0}^2 = \frac{{m_0}^2c^4}{1-v^2/c^2} - {m_0}^2c^4 = \frac{{m_0}^2v^2c^2}{1-v^2/c^2} = m^2v^2c^2 = p^2c^2 \tag{57}$$

即

$$E^2 = p^2c^2 + {m_0}^2c^4 \tag{58}$$

这就是相对论中的能量 – 动量关系。能量和动量是双曲线的关系。

如果某种粒子静止质量为零，即 $m_0 = 0$，则得

$$E = pc \tag{59}$$

比较 (56) 式，可得

$$p = mc \tag{60}$$

可见：该粒子速度就是光速。因此，光可当做一种静止质量为零的粒子流，对应的粒子称为光子。

6 具体物理问题在相对论下的研究

6.1 相对论完全非弹性碰撞

6.1.1 情况一：一球静止一球运动

[问题简述]：设有静止质量皆为 m_0 的两个粒子 A 和 B，B 静止不动，A 以速度 v 与静止的 B 粒子发生完全非弹性碰撞，碰撞后组成一复合粒子，求解复合粒子的相关物理量。

[求解] 碰撞前运动粒子的质量为

$$m = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{61}$$

设复合粒子的速度为 V，静止质量为 M_0, 其运动质量为

$$M = \frac{M_0}{\sqrt{1 - \frac{V^2}{c^2}}} \tag{62}$$

由于碰撞过程中动量守恒，$mv = MV$, 可得方程

$$\frac{m_0 v}{\sqrt{1 - \frac{v^2}{c^2}}} = \frac{M_0 V}{\sqrt{1 - \frac{V^2}{c^2}}} \tag{63}$$

由能量守恒 $m_0c^2 + mc^2 = Mc^2$, 可得质量守恒方程

$$m_0 \frac{\sqrt{1 - \frac{v^2}{c^2}} + 1}{\sqrt{1 - \frac{v^2}{c^2}}} = \frac{M_0}{\sqrt{1 - \frac{V^2}{c^2}}} \tag{64}$$

(63) 除以 (64), 可以得到复合粒子的速度

$$V = \frac{v}{1+\sqrt{1-\frac{v^2}{c^2}}} \tag{65}$$

可见复合粒子的速度随入射粒子速度的增加而增加，但小于入射粒子的速度。

设 $\beta = \frac{v}{c}$, 由 (64) 式可得复合粒子的静止质量为

$$\begin{aligned} M_0 &= m_0 \frac{1+\sqrt{1-\beta^2}}{\sqrt{1-\beta^2}} \sqrt{1-\frac{V^2}{c^2}} \\ &= m_0 \frac{1+\sqrt{1-\beta^2}}{\sqrt{1-\beta^2}} \sqrt{1-\frac{\beta^2}{(1+\sqrt{1-\beta^2})}} \\ &= m_0 \sqrt{\frac{(1+\sqrt{1-\beta^2})^2-\beta^2}{1-\beta^2}} \\ &= m_0 \sqrt{\frac{2(1-\beta^2+\sqrt{1-\beta^2})}{1-\beta^2}} \end{aligned} \tag{66}$$

即

$$M_0 = m_0 \sqrt{2(1+\frac{1}{\sqrt{1-\frac{v^2}{c^2}}})} \tag{67}$$

当 $v \ll c$ 时，由 (65) 和 (67) 得到 $V = \frac{v}{2}$，$M_0 = 2m_0$, 这是经典力学的碰撞结果。

碰撞前后的静止质量差是复合粒子静止质量的增量

$$\begin{aligned} \Delta M_0 &= M_0 - 2m_0 \\ &= m_0 \left[\sqrt{2(1+\frac{1}{\sqrt{1-\frac{v^2}{c^2}}})} - 2 \right] \end{aligned} \tag{68}$$

显然，碰撞后复合粒子的静止质量大于两个粒子的静止质量之和。

粒子碰撞前的动能为

$$\begin{aligned} T_m &= mc^2 - m_0c^2 \\ &= m_0c^2(\frac{1}{\sqrt{1-\frac{v^2}{c^2}}} - 1) \end{aligned} \tag{69}$$

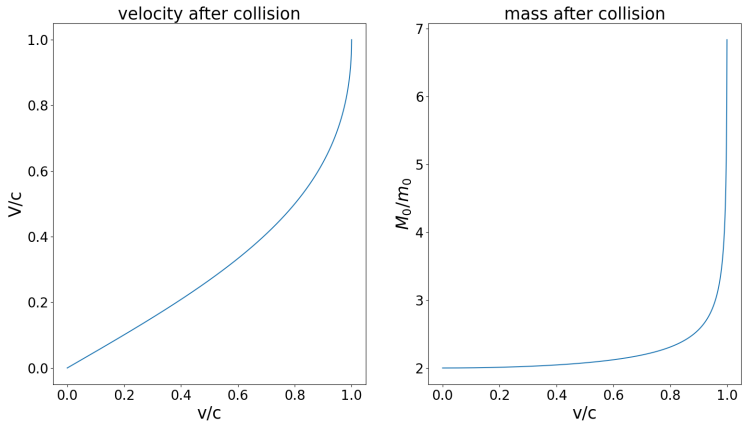

图 19: 相对论下碰撞后复合粒子速度与静止质量随初速度变化曲线

碰撞后的动能为

$$\begin{aligned} T_M &= Mc^2 - M_0c^2 \\ &= M_0c^2(\frac{1}{\sqrt{1-\frac{V^2}{c^2}}} - 1) \end{aligned} \tag{70}$$

利用 (63) 和 (65) 可得

$$T_M = m_0c^2(\frac{1}{\sqrt{1-\frac{v^2}{c^2}}} + 1) - M_0c^2 \tag{71}$$

粒子损失的动能为 $\Delta T = \Delta M_0c^2$, 可见粒子损失的动能转化为复合粒子的静止能量。这是符合能量守恒定律的。

6.1.2 情况二：两球均有初速度

[问题简述]：设两个静止质量皆为 m_0 的粒子 A 和 B 以速度 v_1 和 v_2 在一条直线上运动，它们发生完全非弹性碰撞后结果分析。

[求解] 如果 $v_1 > v_2 > 0$, 表示 B 在前、A 在后，A 与 B 碰撞；如果 $0 < v_1 < v_2$, 表示 A 在前、B 在后，B 与 A 发生碰撞；如果 $v_1 = v_2$，表示 B 与 A 不碰撞，这时，$M = 2m_0$，因而可以当做无碰撞粘合。当 $v_1 < 0$ 或 $v_2 < 0$ 时，表示 A 或 B 反向运动。当 v_1 和 v_2 符号相反时，表示对碰。

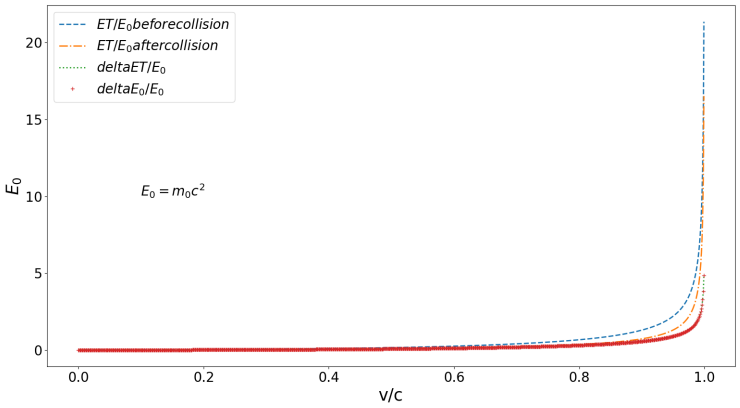

图 20: 相对论下碰撞中碰撞前动能，碰撞后动能，损失的动能，增加的静止质量

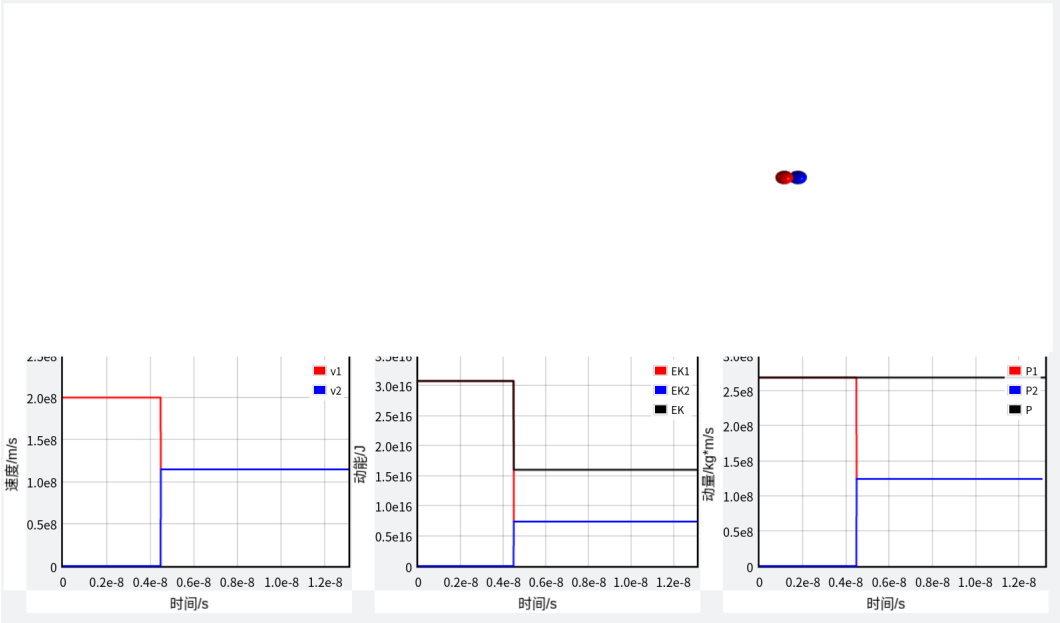

图 21: 我们运用 VPython 对情况一过程进行了模拟，程序设置 $m_1 = m_2 = 1$，初速度 $v_1 = 10^8, v_2 = 0$

```
while ball1.pos.x>=-2 and ball2.pos.x<=2:


    rate(1000)

    ball1.pos.x=ball1.pos.x+v1*dt
    ball2.pos.x=ball2.pos.x+v2*dt

    #计算碰撞后的速度

    if (ball2.pos.x-ball1.pos.x)<=0.1 and v2==0:

        w=np.sqrt(1-(v1/c)**2)
        v1x=v1/(1+w)
        v1=v1x
        v2=v1x

    EK1=(m1/np.sqrt(1-(v1/c)**2)-m1)*c**2
    EK2=(m2/np.sqrt(1-(v2/c)**2)-m2)*c**2
    EK=M*(1/np.sqrt(1-(v1xd/c)**2)-1)*c**2

    P1=m1*v1/np.sqrt(1-(v1/c)**2)
    P2=m2*v2/np.sqrt(1-(v2/c)**2)
    P=M*v1xd/np.sqrt(1-(v1xd/c)**2)


    v1curve.plot(t,v1)
    v2curve.plot(t,v2)

    #动能绘制
    EK1curve.plot(t,EK1)
    EK2curve.plot(t,EK2)
    if (v1==v2):

        EKcurve.plot(t,EK)
    else:
        EKcurve.plot(t,EK1+EK2)

    #动量绘制
    P1curve.plot(t,P1)
    P2curve.plot(t,P2)
    Pcurve.plot(t,P)


    t=t+dt
```

图 22: 这是我们实现图 21的主要循环

设 $\beta_1 = \frac{v_1}{c}$ $\beta_2 = \frac{v_2}{c}$ $\beta = \frac{V}{c}$，根据碰撞过程中动量守恒可得方程

$$\frac{m_0 + v_1}{\sqrt{1-\beta_1^2}} + \frac{m_0 v_2}{\sqrt{1-\beta_2^2}} = \frac{M_0 V}{\sqrt{1-\beta^2}} \tag{72}$$

由能量守恒可得方程

$$\frac{m_0}{\sqrt{1-\beta_1^2}} + \frac{m_0}{\sqrt{1-\beta_2^2}} = \frac{M_0}{\sqrt{1-\beta^2}} \tag{73}$$

将 (72) 除以 (73) 可得碰撞后的速度

$$V = c\frac{\frac{\beta_1}{\sqrt{1-\beta_1^2}} + \frac{\beta_2}{\sqrt{1-\beta_2^2}}}{\frac{1}{\sqrt{1-\beta_1^2}} + \frac{1}{\sqrt{1-\beta_2^2}}} \tag{74}$$

将上式代入 (73) 中可得

$$M_0 = m_0\sqrt{1-\beta^2}\left(\frac{1}{\sqrt{1-\beta_1^2}} + \frac{1}{\sqrt{1-\beta_2^2}}\right) \tag{75}$$

由 (74) 可得

$$\sqrt{1-\beta^2} = \frac{\sqrt{\left(\frac{1}{\sqrt{1-\beta_1^2}} + \frac{1}{\sqrt{1-\beta_2^2}}\right)^2 - \left(\frac{\beta_1}{\sqrt{1-\beta_1^2}} + \frac{\beta_2}{\sqrt{1-\beta_2^2}}\right)^2}}{\frac{1}{\sqrt{1-\beta_1^2}} + \frac{1}{\sqrt{1-\beta_2^2}}} \tag{76}$$

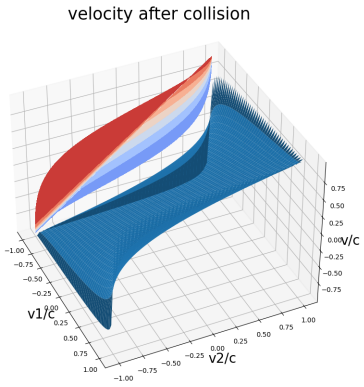

图 23: 情况二相对论下碰撞后复合粒子速度曲面，投影可以更加直观看出碰后速度与两球初速度的关系

所以碰撞后的静止质量为

$$M_0 = m_0\sqrt{(\sqrt{\frac{1+\beta_1}{1-\beta_1}}+\sqrt{\frac{1+\beta_2}{1-\beta_2}})+(\sqrt{\frac{1-\beta_1}{1+\beta_1}}+\sqrt{\frac{1-\beta_2}{1+\beta_2}})} \tag{77}$$

碰撞后的速度和静止质量取决于两粒子碰撞前的速度。当 $v_2 = 0$ 时，表示 B 粒子碰撞前是静止的，A 粒子与 B 粒子碰撞后的速度和静止质量就是 (65) 式和 (67) 式。当 $v_2 = -v_1$ 时，两个粒子对碰，碰撞后的速度为 0，静止质量为

$$M_0 = \frac{2m_0}{\sqrt{1-\beta_1^2}} \tag{78}$$

可见两个粒子对碰，碰撞后对碰粒子的静止质量要大于两个粒子的静止质量之和。

6.2 静止粒子的衰变

[问题简述] 静止质量为 m_0 的粒子衰变成为静止质量分别为 m_{10} 和 m_{20} 的两个粒子 A 和 B, 求衰变后粒子的速度、动量和能量。

[求解] 设衰变后 A 粒子和 B 粒子的能量分别为 E_1 和 E_2，动量分别为 p_1 和 p_2，根据能量守恒可得方程

$$m_0c^2 = E_1 + E_2 \tag{79}$$

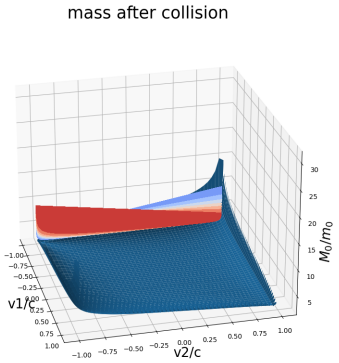

图 24: 情况二相对论下碰撞后复合粒子静止质量曲线, 投影可以更加直观看出静止质量与两球初速度关系

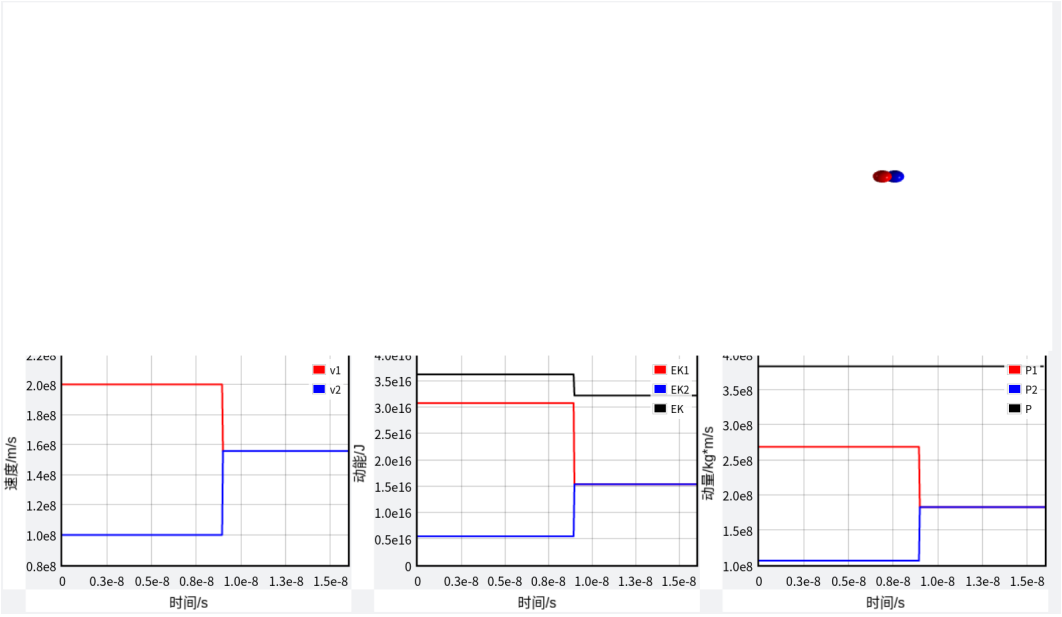

图 25: 我们运用 VPython 对情况二过程进行了模拟，程序设置 $m_1 = m_2 = 1$，初速度 $v_1 = 2 \times 10^8, v_2 = 10^8$

```
while ball1.pos.x>=-2 and ball2.pos.x<=3:

    rate(1000)

    ball1.pos.x=ball1.pos.x+v1*dt
    ball2.pos.x=ball2.pos.x+v2*dt

    #计算碰撞后的速度、动能、动量

    if (ball2.pos.x-ball1.pos.x)<=0.1 and v2==1e8:

        ratio1=1/np.sqrt(1-(v1/c)**2)
        ratio2=1/np.sqrt(1-(v2/c)**2)
        v1x=c*((v1/c)*ratio1+(v2/c)*ratio2)/(ratio1+ratio2)
        v1=v1x
        v2=v1x

    #动能
    EK1=(m1/np.sqrt(1-(v1/c)**2)-m1)*c**2
    EK2=(m2/np.sqrt(1-(v2/c)**2)-m2)*c**2
    EK_tot=EK1+EK2
    EK=EK_tot

    #动量
    P1=np.sqrt((m1**2/(1-(v1/c)**2)-m1**2)*c**2)
    P2=np.sqrt((m2**2/(1-(v2/c)**2)-m2**2)*c**2)
    P=np.sqrt((M**2/(1-(v1xd/c)**2)-M**2)*c**2)

    if (ball2.pos.x-ball1.pos.x)<=0.1:

        EK=(M/np.sqrt(1-(v1x/c)**2)-M)*c**2


    #速度绘制
    v1curve.plot(t,v1)
    v2curve.plot(t,v2)

    #动能绘制
    EK1curve.plot(t,EK1)
    EK2curve.plot(t,EK2)
    EKcurve.plot(t,EK)

    #动量绘制
    P1curve.plot(t,P1)
    P2curve.plot(t,P2)
    Pcurve.plot(t,P)

    t=t+dt
```

图 26: 这是我们实现图 25的主要循环

根据动量守恒可得方程

$$p_1 + p_2 = 0 \tag{80}$$

根据能量和动量的关系可得方程

$$E_1^2 = (p_1 c)^2 + (m_{10}c^2)^2, \quad E_2^2 = (p_2 c)^2 + (m_{20}c^2)^2 \tag{81}$$

两式相减并利用 (80) 可得

$$E_1^2 - E_1^2 = (m_{10}^2 - m_{20}^2)c^4 \tag{82}$$

利用 (79) 式可得

$$m_0(E_1 - E_2) = (m_{10}^2 - m_{20}^2) \tag{83}$$

联立 (79) 式可得 A 粒子的能量

$$E_1 = \frac{(m_0^2 + m_{10}^2 - m_{20}^2)c^2}{2m_0} \tag{84}$$

由于

$$E_1 = \frac{m_{10}c^2}{\sqrt{1 - v_1/c^2}} \tag{85}$$

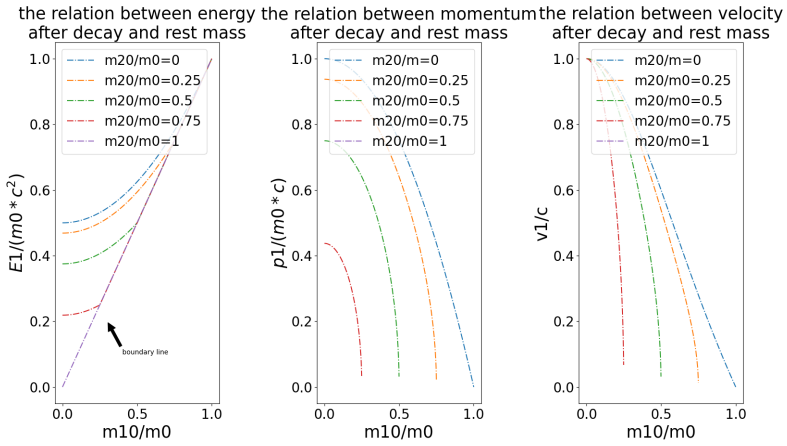

图 27: 静止粒子衰变后 A 粒子能量，A 粒子动量，A 粒子速度与静止质量的关系

利用 (84) 式解的 A 粒子的速度为

$$\begin{aligned} v_1 &= c\sqrt{1-(\frac{m_{10}c^2}{E_1})^2} \\ &= c\frac{\sqrt{[(m_0+m_{10})^2-m_{20}^2][(m_0-m_{10})^2-m_{20}^2]}}{m_0^2+m_{10}^2-m_{20}^2} \end{aligned} \tag{86}$$

由此可知

$$m_0 \geq m_{10}+m_{20} \tag{87}$$

即衰变后的两个粒子的静止质量之和不得大于衰变前粒子的静止质量。

A 粒子的动量为

$$p_1 = m_1 v_1 = \frac{E_1 v_1}{c^2} \tag{88}$$

利用 (84) 式和 (86) 式可得

$$p_1 = c\frac{\sqrt{[(m_0+m_{10})^2-m_{20}^2][(m_0-m_{10}^2-m_{20}^2)]}}{2m_0} \tag{89}$$

用同样的方法可求 B 粒子的 E_2，v_2 和 p_2。由于物理量是对称的，只要将相应公式的下标 1 和下标 2 对管就可以得到 B 粒子的能量、速度和动量。

```
#定义衰变能量函数
def F(ratiom10,ratiom20):
    cond=[False if (i>1-ratiom20) else True for i in ratiom10]  #分段条件
    y=cond*(1+ratiom10**2-ratiom20**2)/2+ratiom10*(ratiom10>1-ratiom20)
    return y
```

图 28: 这是我们定义的衰变函数

[讨论](1) 如果 $m_{10}=0$，表示衰变的 A 粒子是光子，则其能量、速度和动量分别为

$$E_1=\frac{(m_0^2-m_{20}^2)c^2}{2m_0},\quad v_1=c,\quad p_1=c\frac{m_0^2-m_{10}^2}{2m_0} \tag{90}$$

可见 A 粒子的速度就是光速，其动量和能量满足关系式 $E_1=p_1c$

(2) 如果 $m_{20}=0$，表示 B 粒子是光子，则 A 粒子的能量、速度和动量分别为

$$E_1=\frac{(m_0^2+m_{10}^2)c^2}{2m_0},\quad v_1=c\frac{m_0^2-m_{10}^2}{m_0^2+m_{10}^2},\quad p_1=c\frac{m_0^2-m_{10}^2}{2m_0} \tag{91}$$

当 $m_10\to m_0$ 时，则 $E_1\to m_0c^2, E_2\to 0, v_1\to 0, p_1\to 0$，表示静止粒子没有衰变的情况，也没有光子发出。

(3) 如果 $m_{20}=m_0-m_{10}$，则 A 粒子的能量、速度、和动量分别为

$$E_1=m_{10}c^2,\quad v_1=p_1=0 \tag{92}$$

同理可得

$$E_2=m_{20}c^2,\quad v_2=p_2=0 \tag{93}$$

在 $m_{20}=m_0-m_{10}$ 条件下，一个粒子可分成两部分，但不会发生衰变。在 m_{20} 一定的情况下，E_1 表示最大能量。

(4) 如果 $m_{10}=m_{20}$，表示 A 和 B 两粒子静止质量相等，则它们的能量、速度和动量分别为

$$E_1=E_2=\frac{1}{2}m_0c^2,\quad v_1=v_2=c\frac{\sqrt{m_0^2-4m_{10}^2}}{m_0},\quad p_1=p_2=\frac{1}{2}c\sqrt{m_0^2-4m_{10}^2} \tag{94}$$

两粒子的能量各为衰变前静止粒子能量的一半。如果 $m_{10}=m_{20}=0$，表示 A 和 B 两粒子都是光子，速度和动量分别为

$$v_1=v_2=c,\quad p_1=p_2=\frac{1}{2}m_0c \tag{95}$$

如果 $m_{10}=m_{20}=\frac{m_0}{2}$，则两粒子的速度和动量都为零。说明一个静止粒子分成静止质量各一半的粒子时，不会发生衰变。

7 附录

7.1 参考文献

[1]David J.Griffith.Introduction To Electrodynamics[M].Cambridge.Cambridge University Press,2017:532-537.
[2]Robert Johansson.Numerical Python:Scientific Computing and Data Science Applications with Numpy,SciPy,and matplotlib[M]. 北京. 清华大学出版社,2020.
[3] 郭硕鸿. 电动力学 [M]. 北京. 高等教育出版社,2008:223-230.
[4] 刘辽　费保骏　张允中. 狭义相对论 [M]. 北京. 科学出版社,2008:16-37.
[5] 周群益.MATLAB 可视化大学物理学 [M]. 北京. 清华大学出版社,2011.
[6] 张继春.Python 编程与 3D 物理学仿真 [M]. 北京. 电子工业出版社,2021.

7.2 源码

```
#
relativisitic_time_expand.py
import numpy as np
import matplotlib.pyplot as plt

ratio=np.linspace(-1,1,1000)

tt=1/np.sqrt(1-ratio**2)

plt.figure()

ax=plt.subplot(111)
ax.plot(ratio,tt)

ax.set_xlabel('u/c',fontsize=20)
ax.set_ylabel('t/t0',fontsize=20)
ax.set_title('the relation between time expand and velocity',fontsize=20)

ax.tick_params(labelsize=20)

ax.text(0,2,'c is the velocity of light, t0 is the eigentime',fontsize=20
                                        )


plt.show()
```

```
#
relativisitic_mass.py
#相对论速度质量关系

import numpy as np
import matplotlib.pyplot as plt

ratio=np.linspace(0,1,1000)  #比值为物体速度v/c

mm=1/np.sqrt(1-ratio**2)  #mm为物体质量与静止质量比值

plt.figure()

ax=plt.subplot(111)

ax.plot(ratio,mm,'--')  #绘制速度质量关系曲线

x0=[1 for i in np.arange(0,20,0.001)]  #绘制切线
ax.plot(x0,np.arange(0,20,0.001),'-.')

ax.set_xlabel('v/c',fontsize=20)
ax.set_ylabel('m/m0',fontsize=20)
ax.set_title('the relation between mass and velocity',fontsize=20)

ax.tick_params(labelsize=20)

ax.text(0.1,1,'c is the velocity of light, and m0 is rest mass',fontsize
                                        =20)
ax.text(1,15,'tangent line',fontsize=20)

plt.show()
```

```
#
relativisitic_energy_momentum.py
#相对论动量 速度 能量关系

import numpy as np
import matplotlib.pyplot as plt

ratio1=np.linspace(0,1,1000)  #ratio1代表v/c

E_r=1/np.sqrt(1-ratio1**2)  #能量与速度的关系

plt.figure()
```

```
ax=plt.subplot(121)

ax.plot(ratio1,E_r,'-.')  #绘制能量速度关系

ax.set_xlabel('v/c',fontsize=20)
ax.set_ylabel('$E/(m0*c^2)$',fontsize=20)
ax.set_title('the relation between energy and velocity',fontsize=20)
ax.text(0.1,2,'c is the velocity of light,m0 is the mass',fontsize=20)

ax.tick_params(labelsize=20)

ratio2=np.linspace(0,5,1000)  #ratio2代表 p/(m0*c)

E_E=np.sqrt(1+ratio2**2)  #能量与动量的关系

axp=plt.subplot(122)

axp.plot(ratio2,E_E,'--',label='$E/(m0*c^2)$')  #绘制能量与动量关系
axp.plot(ratio2,ratio2,'-.',label='m0=0')

axp.set_xlabel('$p/(m0*c) or p$',fontsize=20)
axp.set_ylabel('$E/(m0*c^2) or E/c$',fontsize=20)
axp.set_title('the raltion between energy and momentum',fontsize=20)

axp.tick_params(labelsize=20)

axp.legend(loc='upper left',frameon=True,fontsize=20)

plt.show()
```

```
#
Bertozzi_v_limit.py
from matplotlib import pyplot as plt
from numpy import linspace
from matplotlib import font_manager as fm
from matplotlib import figure

m0=9.11e-31 #电子的质量
e=1.6e-19 #元电荷的电荷量
c=3e8 #光速
e0=m0*c**2
e0=e0/e/1e6 #单位换算
T=linspace(0,6,400)

u=9*(1-e0**2/(T+e0)**2) #u是相对论体系下电子速度的平方
u0=2*T/e0*9 #经典力学体系下电子速度的平方
```

```
figure,ax=plt.subplots()
plt.ylim((0,10))

plt.annotate('$v^2=2T/m0$',xy=(0.2,8),xytext=(1.,10),arrowprops=dict(
                                        facecolor='black',shrink=0.05),
                                        fontsize=20)  #指示箭头

ax.plot(T,u,label="Relativistic system")

ax.plot(T,u0,label="Classical mechanical system")

y0=[9 for i in T]  #渐近线
ax.plot(T,y0,'--')

ax.plot(2.123,8.55,'go',label="experimental data") #实验数据
ax.plot(0.487,6.71,'go')
ax.plot(0.049,1.48,'go')
ax.plot(0.185,4.20,'go')

ax.set_xlabel('T/MeV',fontsize=20)
ax.set_ylabel('$v^2/(10^(16)m^2s^(-2))$',fontsize=20)
ax.set_title('Bertozzi experiment',fontsize=20)

ax.text(3,9.3,'tangent line',fontsize=20)

ax.tick_params(labelsize=20)

ax.legend(loc='lower right',frameon=True,fontsize=20)

plt.show()
```

```
#
particle_decay.py
#粒子衰变问题的相对论研究

import numpy as np
import matplotlib.pyplot as plt
from pylab import *

ratiom10=np.linspace(0,1,1000)

#定义衰变能量函数

def F(ratiom10,ratiom20):
cond=[False if (i>1-ratiom20) else True for i in ratiom10]  #分段条件
```

```
y=cond*(1+ratiom10**2-ratiom20**2)/2+ratiom10*(ratiom10>1-ratiom20)
return y

plt.figure()
ax1=plt.subplot(131)  #静止粒子衰变后的能量与静止质量的关系

ax1.plot(ratiom10,F(ratiom10,0),'-.',label='m20/m0=0')
ax1.plot(ratiom10,F(ratiom10,0.25),'-.',label='m20/m0=0.25')
ax1.plot(ratiom10,F(ratiom10,0.5),'-.',label='m20/m0=0.5')
ax1.plot(ratiom10,F(ratiom10,0.75),'-.',label='m20/m0=0.75')
ax1.plot(ratiom10,F(ratiom10,1),'-.',label='m20/m0=1')

ax1.legend(loc='upper left',frameon=True,fontsize=20)

ax1.annotate('boundary line',xy=(0.3,0.2),xytext=(0.4,0.1),arrowprops=
                                    dict(facecolor='black',shrink=0.05
                                    ))  #指示箭头的添加

ax1.set_xlabel('m10/m0',fontsize=25)
ax1.set_ylabel('$E1/(m0*c^2)$',fontsize=25)
ax1.set_title('the relation between energy\nafter decay and rest mass',
                                    fontsize=25)
ax1.tick_params(labelsize=20)

#定义衰变后的动量函数

def P(ratiom10,ratiom20):
p=np.sqrt(((1+ratiom10)**2-ratiom20**2)*((1-ratiom10)**2-ratiom20**2))
return p

ax2=plt.subplot(132)  #静止粒子衰变后的动量与静止质量的关系

ax2.plot(ratiom10,P(ratiom10,0),'-.',label='m20/m=0')
ax2.plot(ratiom10,P(ratiom10,0.25),'-.',label='m20/m0=0.25')
ax2.plot(ratiom10,P(ratiom10,0.5),'-.',label='m20/m0=0.5')
ax2.plot(ratiom10,P(ratiom10,0.75),'-.',label='m20/m0=0.75')
ax2.plot(ratiom10,P(ratiom10,1),'-.',label='m20/m0=1')

ax2.legend(loc='upper right',frameon=True,fontsize=20)

ax2.set_xlabel('m10/m0',fontsize=25)
ax2.set_ylabel('$p1/(m0*c)$',fontsize=25)
ax2.set_title('the relation between momentum\nafter decay and rest mass'
                                    ,fontsize=25)
ax2.tick_params(labelsize=20)
```

```
#定义衰变后的动量函数

def V(ratiom10,ratiom20):
up=sqrt(((1+ratiom10)**2-ratiom20**2)*((1-ratiom10)**2-ratiom20**2))
down=1+ratiom10**2-ratiom20**2
v=up/down
return v

ax3=plt.subplot(133)  #静止粒子衰变后的速度与静止质量的关系

ax3.plot(ratiom10,V(ratiom10,0),'-.',label='m20/m=0')
ax3.plot(ratiom10,V(ratiom10,0.25),'-.',label='m20/m0=0.25')
ax3.plot(ratiom10,V(ratiom10,0.5),'-.',label='m20/m0=0.5')
ax3.plot(ratiom10,V(ratiom10,0.75),'-.',label='m20/m0=0.75')
ax3.plot(ratiom10,V(ratiom10,1),'-.',label='m20/m0=1')

ax3.legend(loc='upper right',frameon=True,fontsize=20)

ax3.set_xlabel('m10/m0',fontsize=25)
ax3.set_ylabel('v1/c',fontsize=25)
ax3.set_title('the relation between velocity\nafter decay and rest mass'
                                        ,fontsize=25)
ax3.tick_params(labelsize=20)

plt.show()
```

```
#
Lorentz_axis_time_change.py
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d as axes3d
import matplotlib.animation as animation

#设置参数

c=3*1e8
t0=20.
u1=0.1*c
u2=0.9*c
x=np.arange(0,2*c*t0,c*t0*0.005)
x_01=x/(c*t0)
t=np.arange(0,2*t0,0.1)
t_01=t/t0
x_0,t_0=np.meshgrid(x_01,t_01)

fig1=plt.figure()  #洛伦兹空间变换平面
```

```
def rotate(angle):
ax1.view_init(azim=angle)

plt.suptitle('Lorentz axis change')

ax1=plt.subplot(121,projection='3d')
ax1.plot_surface(t_0,x_0,(x_0-t_0*u1/c)/np.sqrt(1-(u1/c)**2))  #坐标系以
                                        0.1倍光速运动时的情况

ax1.set_xlabel('$t/t_0$')
ax1.set_ylabel('$x/(c*t_0)$')
ax1.set_zlabel('$x_S/(c*t_0)$')
ax1.set(title='$u=0.1*c$')

rotate_animation=animation.FuncAnimation(fig=fig1,func=rotate,frames=np.
                                        arange(0,3600+2,8),interval=50,
                                        repeat=True,blit=False)


def rotate2(angle):
ax2.view_init(azim=angle)

ax2=plt.subplot(122,projection='3d')
ax2.plot_surface(t_0,x_0,(x_0-t_0*u2/c)/np.sqrt(1-(u2/c)**2))  #坐标系以
                                        0.9倍光速运动时的的情况

ax2.set_xlabel('$t/t_0$')
ax2.set_ylabel('$x/(c*t_0)$')
ax2.set_zlabel('$x_S/(c*t_0)$')
ax2.set(title='$u=0.9*c$')

rotate_animation2=animation.FuncAnimation(fig=fig1,func=rotate2,frames=
                                        np.arange(0,3600+2,8),interval=50,
                                        repeat=True,blit=False)  #添加动画


fig2=plt.figure()  #洛伦兹时间变换平面

plt.suptitle('Lorentz time change')

ax3=plt.subplot(121,projection='3d')
ax3.plot_surface(t_0,x_0,(t_0-x_0*u1/c)/np.sqrt(1-(u1/c)**2))  #坐标系以
                                        0.1倍光速运动时的情况

ax3.set_xlabel('$t/t_0$')
```

```
ax3.set_ylabel('$x/(c*t_0)$')
ax3.set_zlabel('$t_s/t0$')
ax3.set(title='$u=0.1*c$')

def rotate3(angle):
ax3.view_init(azim=angle)

rotate_animation3=animation.FuncAnimation(fig=fig2,func=rotate3,frames=
                                          np.arange(0,3600+2,8),interval=50,
                                          repeat=True,blit=False)  #添加动画


ax4=plt.subplot(122,projection='3d')
ax4.plot_surface(t_0,x_0,(t_0-x_0*u2/c)/np.sqrt(1-(u2/c)**2))  #坐标系以
                                          0.9倍光速运动时的的情况

ax4.set_xlabel('$t/t_0$')
ax4.set_ylabel('$x/(c*t_0)$')
ax4.set_zlabel('$t_s/t0$')
ax4.set(title='$u=0.9*c$')

def rotate4(angle):
ax4.view_init(azim=angle)

rotate_animation4=animation.FuncAnimation(fig=fig2,func=rotate4,frames=
                                          np.arange(0,3600+2,8),interval=50,
                                          repeat=True,blit=False)  #添加动画


plt.show()
```

```
#
Lorentz_velocity_change.py
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d as axes3d
import matplotlib.animation as animation
from matplotlib import cm
from pylab import *

#设置参数

c=3*1e8
u_0=np.arange(-1*c,1*c,c*0.01)
u=u_0/c
v_0=np.arange(-1*c,1*c,c*0.01)
```

```
v=v_0/c
U,V=np.meshgrid(u,v)

fig1=plt.figure()

ax1=plt.subplot(131,projection='3d')  #绘制X分量速度的变换曲面
ax1.plot_surface(U,V,(V-U)/(1-V*U))

ax1.set_xlabel('$u/c$')
ax1.set_ylabel('$v_x/c$')
ax1.set_zlabel('$v_xs/c$')
ax1.set(title='Lorentz velocity change--Vx')

def rotate1(angle):
ax1.view_init(azim=angle)

rot_animation1=animation.FuncAnimation(fig=fig1,func=rotate1,frames=np.
                                       arange(0,3600+2,8),interval=50,
                                       repeat=True,blit=False)  #添加动画

y_theta_0=0*np.sqrt(1-U**2)/(1-V*U)  #Y分量

ax2=plt.subplot(132,projection='3d')  #取特殊情况Vz=0,theta=0时Y分量的速
                                      度变换曲面
ax2.plot_surface(U,V,y_theta_0)

ax2.set_xlabel('$u/c$')
ax2.set_ylabel('$v_x/c$')
ax2.set_zlabel('$v_ys/c$')
ax2.set(title='Lorentz velocity change--Vy(with theta=0)')

def rotate2(angle):
ax2.view_init(azim=angle)

rot_animation2=animation.FuncAnimation(fig=fig1,func=rotate2,frames=np.
                                       arange(0,3600+2,8),interval=50,
                                       repeat=True,blit=False)  #添加动画


y_theta_90=V*np.sqrt(1-U**2)/(1-U/c)  #Y分量

ax3=plt.subplot(133,projection='3d')  #取特殊情况Vz=0,theta=90时Y分量的
                                      速度变换曲面
ax3.plot_surface(U,V,y_theta_90)

ax3.set_xlabel('$u/c$')
```

```
ax3.set_ylabel('$v_x/c$')
ax3.set_zlabel('$v_ys/c$')
ax3.set(title='Lorentz velocity change--Vy(with theta=90)')

def rotate3(angle):
ax3.view_init(azim=angle)

rot_animation3=animation.FuncAnimation(fig=fig1,func=rotate3,frames=np.
                                       arange(0,3600+2,8),interval=50,
                                       repeat=True,blit=False)  #添加动画


fig2=plt.figure()

VV1=np.sqrt(1-(1-V*V)*(1-U*U)/(1-V*U)**2)  #速度大小

ax4=plt.subplot(121,projection='3d')  #取特殊情况z=0,theta=0时速度变换曲
                                      面
ax4.plot_surface(U,V,VV1)

ax4.set_xlabel('$u/c$')
ax4.set_ylabel('$v_x/c$')
ax4.set_zlabel('$v_s/c$')
ax4.set(title='Lorentz velocity change--V(with theta=0)')

def rotate4(angle):
ax4.view_init(angle)

rot_animation4=animation.FuncAnimation(fig=fig2,func=rotate4,frames=np.
                                       arange(0,3600+2,8),interval=50,
                                       repeat=True,blit=False)  #添加动画

VV2=np.sqrt(1-(1-V*V)*(1-U*U)/1) #速度大小

ax5=plt.subplot(122,projection='3d')  #取特殊情况Vz=0,theta=90时速度变换
                                      曲面
ax5.plot_surface(U,V,VV2)

ax5.set_xlabel('$u/c$')
ax5.set_ylabel('$v_x/c$')
ax5.set_zlabel('$v_s/c$')
ax5.set(title='Lorentz velocity change--V(with theta=90)')

def rotate5(angle):
ax5.view_init(angle)
```

```
rot_animation5=animation.FuncAnimation(fig=fig2,func=rotate5,frames=np.
                                       arange(0,3600+2,8),interval=50,
                                       repeat=True,blit=False)  #添加动画


thetas=np.arctan((V*np.sqrt(1-U**2))/-U)  #方向角的大小，选取特殊情况
                                       theta=90,Vz=0

fig3=plt.figure()  #方向角的变化曲面绘制

ax6=plt.subplot(111,projection='3d')

ax6.plot_surface(U,V,thetas)

ax6.set_xlabel('$u/c$')
ax6.set_ylabel('$v/c$')
ax6.set(title='direction angle(with theta=90)')

def rotate6(angle):
ax6.view_init(angle)

rot_animation6=animation.FuncAnimation(fig=fig3,func=rotate6,frames=np.
                                       arange(0,3600+2,8),interval=50,
                                       repeat=True,blit=False)  #添加动画

plt.show()
```

```
#
sum_angle.py
#绘制不同坐标系间速度变换三角形的内角和

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
from matplotlib import cm
import matplotlib.animation as animation

ratio10=np.linspace(-1,1,1000)  #比值为坐标系约化速度u/c

ratio20=np.linspace(-1,1,1000)  #比值为一个坐标系中的物体约化速度v/c

ratio1,ratio2=np.meshgrid(ratio10,ratio20)

alpha=np.arctan(1/(ratio1/(ratio2*np.sqrt(1-ratio1**2))))  #变换角alpha

beta=np.arctan(1/(ratio2/(ratio1*np.sqrt(1-ratio2**2))))  #变换角beta
```

```
angle=alpha+beta+np.pi/2  #变换三角形内角和

fig=plt.figure()

ax=plt.subplot(111,projection='3d')

ax.plot_surface(ratio1,ratio2,angle)  #绘制内角和曲面

ax.set_xlabel('u/c',fontsize=20)
ax.set_ylabel('v/c',fontsize=20)
ax.set_zlabel('sum of angle',fontsize=20)

ax.set_title('sum of the angle',fontsize=20)

def rotate(angle):
ax.view_init(azim=angle)

rot_animation=animation.FuncAnimation(fig=fig,func=rotate,frames=np.
                                      arange(0,3600+2,8),interval=50,
                                      repeat=True,blit=False)  #绘制动画

plt.show()
```

```
#
relativisitic_length_shrink.py
import numpy as np
import matplotlib.pyplot as plt

ratio=np.linspace(-1,1,1000)  #比例代表坐标系速度u/c

ll=np.sqrt(1-ratio**2)  #ll为测量长度与本征长度的比值

plt.figure()

ax=plt.subplot(111)  #绘制长度收缩曲线
ax.plot(ratio,ll,'--')
x0=[-1 for i in np.arange(0,1,0.001)]
ax.plot(x0,np.arange(0,1,0.001),'-.')


ax.set_xlabel('u/c',fontsize=20)
ax.set_ylabel('l/l0',fontsize=20)
ax.set_title('length shrink',fontsize=20)

ax.tick_params(labelsize=20)
```

```
ax.text(0,0.5,'c is the velocity of light, l0 is the eigenlength',
                                        fontsize=20)
ax.text(-1,1,'tangent line',fontsize=20)

plt.show()
```

```
#
length_shrink_low.py

from vpython import *

scene.background=color.white  #设置画布背景为白色
scene.center=vector(0,1,0)  #设置画布中心

base = box(pos =vector(0,-1.5,0),size=vector(10,0.1,0.1),color = color.
                                        black) # 创建底座
car = box(size=vector(1,1,1), color = color.blue,make_trail=True) # 创建
                                        小车模型
scene.autoscale=0

c = 3e8
car.pos = vector(0,0,0)
car.v = vector(2e7, 0, 0)

#计算长度收缩系数
length_shorten_Factors = (1 / (sqrt(1 - (car.v.x**2 / c**2))),
1 / (sqrt(1 - (car.v.y**2 / c**2))),
1 / (sqrt(1 - (car.v.z**2 / c**2))))

#计算小车收缩后尺寸
car.size = vector(car.size.x / length_shorten_Factors[0],
car.size.y / length_shorten_Factors[1],
car.size.z / length_shorten_Factors[2])

t=0 # 设定开始时间
dt= 1e-11  # 设定时间步长

while  car.pos.x <=4.5 :  #用物体x方向最大坐标极限设定分析时间

rate(1000)
car.pos = car.pos + car.v*dt  #计算小车位置更新
t = t + dt   #时间迭代

print(t)
print(car.size)
```

```
#
length_shrink_fast.py
from vpython import *

scene.background=color.white  #设置画布背景为白色
scene.center=vector(0,1,0)  #设置画布中心

base = box(pos =vector(0,-1.5,0),size=vector(10,0.1,0.1),color = color.
                                        black) # 创建底座
car = box(size=vector(1,1,1), color = color.blue,make_trail=True) # 创建
                                        小车模型
scene.autoscale=0

c = 3e8
car.pos = vector(0,0,0)
car.v = vector(2.8e8, 0, 0)

#计算长度收缩系数
length_shorten_Factors = (1 / (sqrt(1 - (car.v.x**2 / c**2))),
1 / (sqrt(1 - (car.v.y**2 / c**2))),
1 / (sqrt(1 - (car.v.z**2 / c**2))))

#计算小车收缩后尺寸
car.size = vector(car.size.x / length_shorten_Factors[0],
car.size.y / length_shorten_Factors[1],
car.size.z / length_shorten_Factors[2])

t=0 # 设定开始时间
dt= 1e-11  # 设定时间步长

while  car.pos.x <=4.5 :  #用物体x方向最大坐标极限设定分析时间

rate(1000)
car.pos = car.pos + car.v*dt  #计算小车位置更新
t = t + dt   #时间迭代

print(t)
print(car.size)
```

```
#
length_no_shrink.py
from vpython import *

scene.background=color.white  #设置画布背景为白色
scene.center=vector(0,1,0)  #设置画布中心
```

```
base = box(pos =vector(0,-1.5,0),size=vector(10,0.1,0.1),color = color.
                                        black) # 创建底座
car = box(size=vector(1,1,1), color = color.blue,make_trail=True) # 创建
                                        小车模型
scene.autoscale=0

c = 3e8
car.pos = vector(0,0,0)
car.v = vector(100, 0, 0)

#计算长度收缩系数
length_shorten_Factors = (1 / (sqrt(1 - (car.v.x**2 / c**2))),
1 / (sqrt(1 - (car.v.y**2 / c**2))),
1 / (sqrt(1 - (car.v.z**2 / c**2))))

#计算小车收缩后尺寸
car.size = vector(car.size.x / length_shorten_Factors[0],
car.size.y / length_shorten_Factors[1],
car.size.z / length_shorten_Factors[2])

t=0 # 设定开始时间
dt= 1e-4  # 设定时间步长

while  car.pos.x <=4.5 :  #用物体x方向最大坐标极限设定分析时间

rate(1000)
car.pos = car.pos + car.v*dt  #计算小车位置更新
t = t + dt   #时间迭代

print(t)
print(car.size)
```

```
#
collision_line1.py
#两个质量相同的小球，一个小球静止，一个小球以v碰向另一个小球发生完全非弹
                                        性碰撞

import matplotlib.pyplot as plt
import numpy as np

ratio=np.linspace(0,1,1000)  #比值为v/c

Vc=ratio/(1+np.sqrt(1-ratio**2)) #碰撞后的速度

#碰撞后速度的绘制
```

```
plt.figure()

ax1=plt.subplot(121)

ax1.plot(ratio,Vc)
ax1.set_ylabel('V/c',fontsize=25)
ax1.set_xlabel('v/c',fontsize=25)
ax1.set_title('velocity after collision',fontsize=25)
ax1.tick_params(labelsize=20)

Mm=np.sqrt(2*(1+1/np.sqrt(1-ratio**2)))  #碰撞后的静止质量

ax2=plt.subplot(122)

ax2.plot(ratio,Mm)
ax2.set_ylabel('$M_0/m_0$',fontsize=25)
ax2.set_xlabel('v/c',fontsize=25)
ax2.set_title('mass after collision',fontsize=25)
ax2.tick_params(labelsize=20)

T_before=1/np.sqrt(1-ratio**2)-1  #碰撞前动能
T_after=1/np.sqrt(1-ratio**2)+1-Mm  #碰撞后动能

deltaT=Mm-2  #损失的动能

deltaE=Mm-2 #增加的静止能量

#动能相关图像的描绘

plt.figure()
ax=plt.subplot(111)

ax.plot(ratio,T_before,'--',lw=2,label='$ET/E_0 before collision$')
ax.plot(ratio,T_after,'-.',lw=2,label='$ET/E_0 after collision$')
ax.plot(ratio,deltaT,':',lw=2,label='$delta ET/E_0$')
ax.plot(ratio,deltaE,'+',lw=1,label='$delta E_0/E_0$')

ax.set_xlabel('v/c',fontsize=25)
ax.set_ylabel('$E_0$',fontsize=25)

ax.legend(loc='upper left',frameon=True,fontsize=20)

ax.text(0.1,10.0,'$E_0=m_0c^2$',fontsize=20)

ax.tick_params(labelsize=20)
plt.show()
```

```
#
collision_line2.py
#两个质量相同的小球分别以v1,v2速度运动然后发生相对论完全非弹性碰撞

import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import axes3d
from matplotlib import cm
import matplotlib.animation as animation
from matplotlib.widgets import Slider

beta11=np.linspace(-1,1,1000)  #设置v1/c的范围
beta21=np.linspace(-1,1,1000)  #设置v2/c的范围

beta1,beta2=np.meshgrid(beta11,beta21)

Vc=(beta1/np.sqrt(1-beta1**2)+beta2/np.sqrt(1-beta2**2))/(1/np.sqrt(1-
                                    beta1**2)+1/np.sqrt(1-beta2**2))
                                    #推导出的末态速度

fig=plt.figure()
ax1=plt.subplot(111,projection='3d')
ax1.plot_surface(beta1,beta2,Vc)  #绘制三维速度平面

ax1.set_xlabel('v1/c',fontsize=20)
ax1.set_ylabel('v2/c',fontsize=20)
ax1.set_zlabel('v/c',fontsize=20)
ax1.set_title('velocity after collision',fontsize=25)

def rotate1(angle):
ax1.view_init(angle)

rot_animation1=animation.FuncAnimation(fig=fig,func=rotate1,frames=np.
                                    arange(0,3600+2,8),interval=50,
                                    repeat=True,blit=False)  #添加动画

ax1.contourf(beta1,beta2,Vc,zdir='x',offset=-1,cmap=cm.coolwarm)  #作出
                                    投影，更加直观看出速度变化

Mm=np.sqrt((np.sqrt((1+beta1)/(1-beta1))+np.sqrt((1+beta2)/(1-beta2)))*(
                                    np.sqrt((1-beta1)/(1+beta1))+np.
                                    sqrt((1-beta2)/(1+beta2))))  #推导
                                    出的末态静止质量

fig2=plt.figure()
ax2=plt.subplot(111,projection='3d')
```

```
ax2.plot_surface(beta1,beta2,Mm)  #绘制三维静止质量平面

ax2.set_xlabel('v1/c',fontsize=20)
ax2.set_ylabel('v2/c',fontsize=20)
ax2.set_zlabel('$M_0/m_0$',fontsize=20)
ax2.set_title('mass after collision',fontsize=25)


ax2.contourf(beta1,beta2,Mm,zdir='x',offset=-1,cmap=cm.coolwarm)  #作出
                                        投影，更加直观看出质量变化

def rotate2(angle):
ax2.view_init(angle)

rot_animation2=animation.FuncAnimation(fig=fig2,func=rotate2,frames=np.
                                        arange(0,3600+2,8),interval=50,
                                        repeat=True,blit=False)  #添加动画

#绘制Slider图像

#碰撞后速度函数的定义
def spv(betas):
ratios=np.linspace(-1,1,1000)
Vvs=(ratios/np.sqrt(1-ratios**2)+betas/np.sqrt(1-betas**2))/(1/np.sqrt(1
                                        -ratios**2)+1/np.sqrt(1-betas**2))
return ratios,Vvs

figs=plt.figure()

axs=plt.subplot(111)

ratios,Vs=spv(0)

l,=plt.plot(ratios,Vs)

axcolor='lightgoldenrodyellow'

oms=plt.axes([0.25,0.15,0.65,0.03],facecolor=axcolor)

soms=Slider(oms,r'$v_2/c$',-1,1,valinit=0)

def update(val):
s=soms.val
ratios,Vs=spv(s)
l.set_ydata(Vs)
l.set_xdata(ratios)
```

```
figs.canvas.draw_idle()

soms.on_changed(update)  #Slider绘制

axs.set_xlabel('$v_1/c$')
axs.set_ylabel('$v/c$')

#碰撞后静止质量函数的定义
def spv2(betas):
ratio2s=np.linspace(-1,1,1000)
Mms=np.sqrt((np.sqrt((1+ratio2s)/(1-ratio2s))+np.sqrt((1+betas)/(1-betas
                                        )))*(np.sqrt((1-ratio2s)/(1+
                                        ratio2s))+np.sqrt((1-betas)/(1+
                                        betas))))
return ratio2s,Mms

fig2s=plt.figure()

ax2s=plt.subplot(111)

ratio2s,Ms=spv2(0)

l2,=plt.plot(ratio2s,Ms)

om2s=plt.axes([0.25,0.15,0.65,0.03],facecolor=axcolor)

som2s=Slider(om2s,r'$v_2/c$',-1,1,valinit=0)


def update2(val):
s2=som2s.val
ratio2s,Ms=spv2(s2)
l2.set_ydata(Ms)
l2.set_xdata(ratio2s)
fig2s.canvas.draw_idle()

som2s.on_changed(update2)  #Slider绘制

ax2s.set_ylabel('$M_0/m_0$')
ax2s.set_xlabel('$v_1/c$')

plt.show()
```

```
#
nonrelativisitic_collision1.py
from vpython import *
```

```
s1=canvas(width=1200,height=400,background=color.white,center=vector(1,0
                                    ,0) ,align="left")  #定义画布
s1.camera.pos=vector(0,1,1) #设置摄像机位置
s1.camera.axis=vector(0,-1,-1)  #设置摄像机方位

ball1=sphere(pos=vector(0,0,0),radius=0.05,color=color.red)  #定义小球
                                    ball1
ball2=sphere(pos=vector(1,0,0),radius=0.05,color=color.blue)  #定义小球
                                    ball2

#定义曲线显示窗口
g1=graph(width=400,height=300,xtitle="时间/s",ytitle="速度/m/s" ,align="
                                    left"  )
v1curve=gcurve(color=color.red,graph=g1,label="v1 ")  #定义ball1速度曲线
v2curve=gcurve(color=color.blue,graph=g1,label="v2 ")  #定义ball2速度曲
                                    线

#定义动能显示窗口
g2=graph(width=400,height=300,xtitle="时间/s",ytitle="动能/J"  ,align="
                                    left" )
EK1curve=gcurve(color=color.red,graph=g2,label="EK1 ")  #定义ball1动能曲
                                    线
EK2curve=gcurve(color=color.blue,graph=g2,label="EK2 ")  #定义ball2动能
                                    曲线
EKcurve=gcurve(color=color.black,graph=g2,label="EK ")  #定义总动能曲线

#定义动量曲线显示窗口
g3=graph(width=400,height=300,xtitle="时间/s",ytitle="动量/kg*m/s"  ,
                                    align="left" )
P1curve=gcurve(color=color.red,graph=g3,label="P1 ")  #定义ball1动量曲线
P2curve=gcurve(color=color.blue,graph=g3,label="P2 ")  #定义ball2动量曲
                                    线
Pcurve=gcurve(color=color.black,graph=g3,label="P ")  #定义总动量曲线

#参数设置

m1=1
m2=1

v1=1
v2=0

t=0
dt=0.001
```

```
while ball1.pos.x>=-2 and ball2.pos.x<=2:

rate(1000)

ball1.pos.x=ball1.pos.x+v1*dt
ball2.pos.x=ball2.pos.x+v2*dt
#计算碰撞后的速度

if (ball2.pos.x-ball1.pos.x)<=0.1:
v1x=(m1*v1+m2*v2)/(m1+m2)
v1=v1x
v2=v1x

#动能计算
EK1=0.5*m1*v1**2
EK2=0.5*m2*v2**2
EK=EK1+EK2

#动量计算
P1=m1*v1
P2=m2*v2
P=P1+P2

#速度绘制
v1curve.plot(t,v1)
v2curve.plot(t,v2)

#动能绘制
EK1curve.plot(t,EK1)
EK2curve.plot(t,EK2)
EKcurve.plot(t,EK)

#动量绘制
P1curve.plot(t,P1)
P2curve.plot(t,P2)
Pcurve.plot(t,P)

t=t+dt  #迭代时间


print("v1=%.2f"%v1,"v2=%.2f"%v2)
```

```
#
nonrelativistic_collision2.py
#非相对论条件两球的完全非弹性碰撞，两球均有初速度
```

```
from vpython import *

s1=canvas(width=1200,height=400,background=color.white,center=vector(1,0
                                    ,0) ,align="left")  #定义画布
s1.camera.pos=vector(0,1,1) #设置摄像机位置
s1.camera.axis=vector(0,-1,-1)  #设置摄像机方位

ball1=sphere(pos=vector(0,0,0),radius=0.05,color=color.red)  #定义小球
                                    ball1
ball2=sphere(pos=vector(1,0,0),radius=0.05,color=color.blue)  #定义小球
                                    ball2

#定义曲线显示窗口
g1=graph(width=400,height=300,xtitle="时间/s",ytitle="速度/m/s" ,align="
                                    left"  )
v1curve=gcurve(color=color.red,graph=g1,label="v1 ")  #定义ball1速度曲线
v2curve=gcurve(color=color.blue,graph=g1,label="v2 ")  #定义ball2速度曲
                                    线

#定义动能显示窗口
g2=graph(width=400,height=300,xtitle="时间/s",ytitle="动能/J"  ,align="
                                    left" )
EK1curve=gcurve(color=color.red,graph=g2,label="EK1 ")  #定义ball1动能曲
                                    线
EK2curve=gcurve(color=color.blue,graph=g2,label="EK2 ")  #定义ball2动能
                                    曲线
EKcurve=gcurve(color=color.black,graph=g2,label="EK ")  #定义总动能曲线

#定义动量曲线显示窗口
g3=graph(width=400,height=300,xtitle="时间/s",ytitle="动量/kg*m/s"  ,
                                    align="left" )
P1curve=gcurve(color=color.red,graph=g3,label="P1 ")  #定义ball1动量曲线
P2curve=gcurve(color=color.blue,graph=g3,label="P2 ")  #定义ball2动量曲
                                    线
Pcurve=gcurve(color=color.black,graph=g3,label="P ")  #定义总动量曲线

#参数设置

m1=1
m2=1

v1=2
v2=1

t=0
```

```
dt=0.001

while ball1.pos.x>=-2 and ball2.pos.x<=3:

rate(1000)

ball1.pos.x=ball1.pos.x+v1*dt
ball2.pos.x=ball2.pos.x+v2*dt
#计算碰撞后的速度

if (ball2.pos.x-ball1.pos.x)<=0.1:
v1x=(m1*v1+m2*v2)/(m1+m2)
v1=v1x
v2=v1x

#动能计算
EK1=0.5*m1*v1**2
EK2=0.5*m2*v2**2
EK=EK1+EK2

#动量计算
P1=m1*v1
P2=m2*v2
P=P1+P2

#速度绘制
v1curve.plot(t,v1)
v2curve.plot(t,v2)

#动能绘制
EK1curve.plot(t,EK1)
EK2curve.plot(t,EK2)
EKcurve.plot(t,EK)

#动量绘制
P1curve.plot(t,P1)
P2curve.plot(t,P2)
Pcurve.plot(t,P)

t=t+dt


print("v1=%.2f"%v1,"v2=%.2f"%v2)
```

```
#
relativistic_collision1.py
```

```
#相对论下完全非弹性碰撞，一球无初速度
from vpython import *
import numpy as np

s1=canvas(width=1200,height=400,background=color.white,center=vector(1,0,0
                                    ) ,align="left")
s1.camera.pos=vector(0,1,1) #设置摄像机位置
s1.camera.axis=vector(0,-1,-1)  #设置摄像机方位

ball1=sphere(pos=vector(0,0,0),radius=0.05,color=color.red)
ball2=sphere(pos=vector(1,0,0),radius=0.05,color=color.blue)

g1=graph(width=400,height=300,xtitle="时间/s",ytitle="速度/m/s" ,align="
                                    left"  )  #定义曲线显示窗口
v1curve=gcurve(color=color.red,graph=g1,label="v1 ")  #定义ball1速度曲线
v2curve=gcurve(color=color.blue,graph=g1,label="v2 ")  #定义ball2速度曲线

#定义动能显示窗口
g2=graph(width=400,height=300,xtitle="时间/s",ytitle="动能/J"  ,align="
                                    left" )
EK1curve=gcurve(color=color.red,graph=g2,label="EK1 ")  #定义ball1动能曲线
EK2curve=gcurve(color=color.blue,graph=g2,label="EK2 ")  #定义ball2动能曲
                                    线
EKcurve=gcurve(color=color.black,graph=g2,label="EK ")  #定义总动能曲线

#定义动量曲线显示窗口
g3=graph(width=400,height=300,xtitle="时间/s",ytitle="动量/kg*m/s"  ,align
                                    ="left" )
P1curve=gcurve(color=color.red,graph=g3,label="P1 ")  #定义ball1动量曲线
P2curve=gcurve(color=color.blue,graph=g3,label="P2 ")  #定义ball2动量曲线
Pcurve=gcurve(color=color.black,graph=g3,label="P ")  #定义总动量曲线

#参数设置
m1=1
m2=1

v1=2e8
v2=0
c=3e8

t=0
dt=1e-11

Beta=np.sqrt(1-(v1/c)**2)

M=m1*np.sqrt(2*(1+1/Beta))
```

```
wd=np.sqrt(1-(v1/c)**2)
v1xd=v1/(1+wd)

while ball1.pos.x>=-2 and ball2.pos.x<=2:



rate(1000)

ball1.pos.x=ball1.pos.x+v1*dt
ball2.pos.x=ball2.pos.x+v2*dt

#计算碰撞后的速度

if (ball2.pos.x-ball1.pos.x)<=0.1 and v2==0:

w=np.sqrt(1-(v1/c)**2)
v1x=v1/(1+w)
v1=v1x
v2=v1x
#计算能量、动量

EK1=(m1/np.sqrt(1-(v1/c)**2)-m1)*c**2
EK2=(m2/np.sqrt(1-(v2/c)**2)-m2)*c**2
EK=M*(1/np.sqrt(1-(v1xd/c)**2)-1)*c**2

P1=m1*v1/np.sqrt(1-(v1/c)**2)
P2=m2*v2/np.sqrt(1-(v2/c)**2)
P=M*v1xd/np.sqrt(1-(v1xd/c)**2)



v1curve.plot(t,v1)
v2curve.plot(t,v2)

#动能绘制
EK1curve.plot(t,EK1)
EK2curve.plot(t,EK2)
if (v1==v2):

EKcurve.plot(t,EK)
else:
EKcurve.plot(t,EK1+EK2)

#动量绘制
P1curve.plot(t,P1)
```

```
P2curve.plot(t,P2)
Pcurve.plot(t,P)


t=t+dt


print("v1=%.2f"%v1,"v2=%.2f"%v2)
```

```
#
relativistic_collision2.py
#相对论条件下两球的完全非弹性碰撞，两球均有初速度

from vpython import *
import numpy as np

s1=canvas(width=1200,height=400,background=color.white,center=vector(1,0
                                        ,0) ,align="left")
s1.camera.pos=vector(0,1,1) #设置摄像机位置
s1.camera.axis=vector(0,-1,-1)  #设置摄像机方位

ball1=sphere(pos=vector(0,0,0),radius=0.05,color=color.red)
ball2=sphere(pos=vector(1,0,0),radius=0.05,color=color.blue)

#定义曲线显示窗口
g1=graph(width=400,height=300,xtitle="时间/s",ytitle="速度/m/s" ,align="
                                        left"  )
v1curve=gcurve(color=color.red,graph=g1,label="v1 ")  #定义ball1速度曲线
v2curve=gcurve(color=color.blue,graph=g1,label="v2 ")  #定义ball2速度曲
                                        线

#定义动能显示窗口
g2=graph(width=400,height=300,xtitle="时间/s",ytitle="动能/J"  ,align="
                                        left" )
EK1curve=gcurve(color=color.red,graph=g2,label="EK1 ")  #定义ball1动能曲
                                        线
EK2curve=gcurve(color=color.blue,graph=g2,label="EK2 ")  #定义ball2动能
                                        曲线
EKcurve=gcurve(color=color.black,graph=g2,label="EK ")  #定义总动能曲线

#定义动量曲线显示窗口
g3=graph(width=400,height=300,xtitle="时间/s",ytitle="动量/kg*m/s"  ,
                                        align="left" )
P1curve=gcurve(color=color.red,graph=g3,label="P1 ")  #定义ball1动量曲线
```

```
P2curve=gcurve(color=color.blue,graph=g3,label="P2 ")  #定义ball2动量曲
                                                       线
Pcurve=gcurve(color=color.black,graph=g3,label="P ")  #定义总动量曲线

#参数设置
m1=1
m2=1

v1=2e8
v2=1e8
c=3e8

t=0
dt=1e-11

beta1=v1/c
beta2=v2/c

Beta1=np.sqrt((1+beta1)/(1-beta1))
Beta2=np.sqrt((1+beta2)/(1-beta2))
Beta3=np.sqrt((1-beta1)/(1+beta2))
Beta4=np.sqrt((1-beta2)/(1+beta2))
M=m1*np.sqrt((Beta1+Beta2)*(Beta3+Beta4))

ratio1d=1/np.sqrt(1-(v1/c)**2)
ratio2d=1/np.sqrt(1-(v2/c)**2)

v1xd=c*((v1/c)*ratio1d+(v2/c)*ratio2d)/(ratio1d+ratio2d)

while ball1.pos.x>=-2 and ball2.pos.x<=3:

rate(1000)

ball1.pos.x=ball1.pos.x+v1*dt
ball2.pos.x=ball2.pos.x+v2*dt

#计算碰撞后的速度、动能、动量

if (ball2.pos.x-ball1.pos.x)<=0.1 and v2==1e8:

ratio1=1/np.sqrt(1-(v1/c)**2)
ratio2=1/np.sqrt(1-(v2/c)**2)
v1x=c*((v1/c)*ratio1+(v2/c)*ratio2)/(ratio1+ratio2)
v1=v1x
v2=v1x
```

```
#动能
EK1=(m1/np.sqrt(1-(v1/c)**2)-m1)*c**2
EK2=(m2/np.sqrt(1-(v2/c)**2)-m2)*c**2
EK_tot=EK1+EK2
EK=EK_tot

#动量
P1=np.sqrt((m1**2/(1-(v1/c)**2)-m1**2)*c**2)
P2=np.sqrt((m2**2/(1-(v2/c)**2)-m2**2)*c**2)
P=np.sqrt((M**2/(1-(v1xd/c)**2)-M**2)*c**2)

if (ball2.pos.x-ball1.pos.x)<=0.1:

EK=(M/np.sqrt(1-(v1x/c)**2)-M)*c**2


#速度绘制
v1curve.plot(t,v1)
v2curve.plot(t,v2)

#动能绘制
EK1curve.plot(t,EK1)
EK2curve.plot(t,EK2)
EKcurve.plot(t,EK)

#动量绘制
P1curve.plot(t,P1)
P2curve.plot(t,P2)
Pcurve.plot(t,P)

t=t+dt


print("v1=%.2f"%v1,"v2=%.2f"%v2)
```