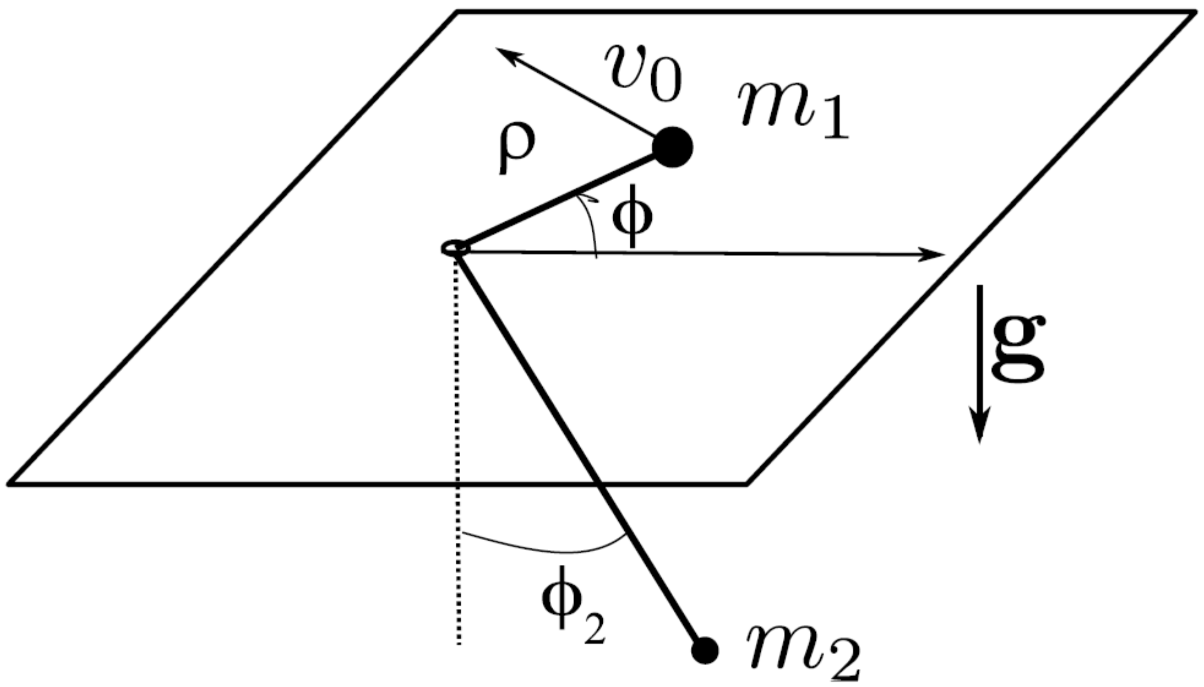
Problema 6(c): Solución y Simulación con VPython

6. Dos partículas de masas m_1 y m_2 están unidas por un hilo de longitud L, como indica la figura. La masa m_1 se mueve en el plano de la mesa y m_2 sólo verticalmente. En $t = 0$, m_1 se encuentra a una distancia $r_0 < L$ del orificio y se le aplica una velocidad v_0 perpendicular al hilo.

   (a) Escriba las ecuaciones de Lagrange y halle sus integrales primeras en términos de las condiciones iniciales.

   (b) Halle la tensión del hilo.

   (c) Repita (a) y (b), pero ahora la masa m_2 puede moverse en las dos direcciones de un plano vertical.



Lagrangiano

Partimos del Lagrangiano

$$\mathcal{L}(q_i, \dot{q}_i) = T - V$$

para este problema se usa las coordenadas ρ y ϕ de la masa m_1 y la coordenada ϕ_2 de la masa m_2 (3 Grados de libertad).

$$\begin{aligned} T &= \frac{m_1}{2}(\dot{\rho}^2 + \rho^2\dot{\phi}^2) + \frac{m_2}{2}(\dot{\rho}^2 + (L-\rho)^2\dot{\phi}_2^2) \\ V &= -m_2 g(L-\rho)\cos(\phi_2) \end{aligned}$$

obteniendo:

$$\mathcal{L} = \frac{m_1}{2}(\dot{\rho}^2 + \rho^2\dot{\phi}^2) + \frac{m_2}{2}(\dot{\rho}^2 + (L-\rho)^2\dot{\phi}_2^2) + m_2 g(L-\rho)\cos(\phi_2)$$

Ecuaciones de Lagrange

Recordemos:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}(q_i, \dot{q}_i, t)}{\partial \dot{q}_i}\right) = \frac{\partial \mathcal{L}(q_i, \dot{q}_i, t)}{\partial q_i}$$

donde q_i, $\dot{q}_i$ las coordenadas y velocidades generalizadas.

$$\mathcal{L} = \frac{m_1}{2}(\dot\rho^2 + \rho^2\dot\phi^2) + \frac{m_2}{2}(\dot\rho^2 + (L-\rho)^2\dot\phi_2^2) + m_2 g(L-\rho)\cos(\phi_2)$$

ρ:

$$\frac{\partial\mathcal{L}}{\partial\dot\rho} = (m_1+m_2)\dot\rho \quad \Rightarrow \quad \frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot\rho}\right) = (m_1+m_2)\ddot\rho, \qquad \frac{\partial\mathcal{L}}{\partial\rho} = m_1\rho\dot\phi^2 - m_2(L-\rho)\dot\phi_2^2 - m_2 g\cos(\phi_2)$$

$$(m_1+m_2)\ddot\rho = m_1\rho\dot\phi^2 - m_2(L-\rho)\dot\phi_2^2 - m_2 g\cos(\phi_2)$$

ϕ:

$$\frac{\partial\mathcal{L}}{\partial\dot\phi} = m_1\rho^2\dot\phi \quad \Rightarrow \quad \frac{d}{dt}(m_1\rho^2\dot\phi) = m_1(\rho^2\ddot\phi + 2\rho\dot\rho\dot\phi) \qquad \frac{\partial\mathcal{L}}{\partial\phi} = 0, (\text{coordenada cíclica}) \quad p_\phi = m_1\rho^2\dot\phi = l_z$$

$$\rho^2\ddot\phi + 2\rho\dot\rho\dot\phi = 0$$

$$\mathcal{L} = \frac{m_1}{2}(\dot\rho^2 + \rho^2\dot\phi^2) + \frac{m_2}{2}(\dot\rho^2 + (L-\rho)^2\dot\phi_2^2) + m_2 g(L-\rho)\cos(\phi_2)$$

ϕ_2:

$$\frac{\partial\mathcal{L}}{\partial\dot\phi_2} = m_2(L-\rho)^2\dot\phi_2 \quad \Rightarrow \quad \frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot\phi_2}\right) = m_2\left((L-\rho)^2\ddot\phi_2 - 2(L-\rho)\dot\phi_2\dot\rho\right)$$

$$\frac{\partial\mathcal{L}}{\partial\phi_2} = -m_2 g\sin(\phi_2)$$

$$(L-\rho)^2\ddot\phi_2 - 2(L-\rho)\dot\phi_2\dot\rho = -g(L-\rho)\sin(\phi_2)$$

Simulación

Pasamos a ecuaciones diferenciales de primer orden en el tiempo:

$$\begin{aligned}
\frac{d\rho}{dt} &= \dot\rho \\
\frac{d\phi}{dt} &= \dot\phi \\
\frac{d\phi_2}{dt} &= \dot\phi_2 \\
\frac{d\dot\rho}{dt} &= \frac{1}{1+m_2/m_1}\left(\rho\dot\phi^2 - \frac{m_2}{m_1}(L-\rho)\dot\phi_2^2 - \frac{m_2}{m_1}g\cos(\phi_2)\right) \\
\frac{d\dot\phi}{dt} &= -\frac{2\dot\phi\dot\rho}{\rho} \\
\frac{d\dot\phi_2}{dt} &= \frac{1}{(L-\rho)^2}\left(-g(L-\rho)\sin(\phi_2) + 2(L-\rho)\dot\phi_2\dot\rho\right)
\end{aligned}$$

¶

Método de Euler (primer orden)

Queremos resolver:

$$\frac{d\vec{y}}{dt} = \vec{g}(\vec{y}, t) \qquad \vec{y}(t_0) = \vec{y}_0$$

dividimos el tiempo en intervalos de longitud Δt y evaluamos $\vec{y}(t_{i+1})$ a primer orden en función de $\vec{y}(t_i)$ en forma iterativa.

$$\vec{y}_{i+1} = \vec{y}_i + \vec{g}(\vec{y}_i, t_i))\Delta t$$

In [3]:
```
from vpython import *     ## importa libreria vpython con clases de objetos graficos y funciones matematicas
canvas()  ## lienzo, escena

box()

micaja=box(axis=vec(0,2,0))
micaja.color=color.red
```

In [ ]:
```
from vpython import *     ## importa libreria vpython con clases de objetos graficos y funciones matematicas
scene2=canvas(background=color.white)  ## lienzo, escena

micaja=box(axis=vec(0,2,0))
micaja.color=color.red

t=0
dt=0.01
while True:
    rate(100)  # el lazose ejecuta a esterate por segundo, en un segundo se pasa de t=0 a t=1(vuelta en 2pi segundos)
    micaja.axis=vec(2*sin(t),2*cos(t),0)
    micaja.color=vec(sin(2*t)*sin(t),sin(2*t)*cos(t),cos(2*t))
    t=t+dt
```

In [ ]:
```
from vpython import *      # Cargamos las Librerias de VPython que incluyen Clases de Objetos Graficos, y funciones Matemáticas

canvas()   # Creamos una escena  (se crea una por default en glowscript)
#GlowScript 3.0 VPython ( para uso en glowscript en lugar de la importación de arriba)

# Constantes fisicas del sistema:
m1=1.  ##  no depende de este valor
s0=1.  ##  Cociente de masas   s0=m2/m1
s=0.1   ##  (s=3., 0.1)  ## Define la velocidad tangencial v0=sqrt(s*g*rho), s=1,phi2=0 :circular
m2=s0*m1
L=2.   #  Longitud de la cuerda
g=9.8   #  aceleracion de la gravedad

# Condiciones iniciales
rho=1.
rho_dot=0.
phi=0.
v0=sqrt(s*g*rho)  #  valor auxiliar
phi_dot=v0/rho
phi2=0.5   # (phi2= 0.5)
phi2_dot=0.


#Lz=m1*rho0*v0

#  Creamos el sistema en la escena
box(color=color.green,pos=vector(0,-0.1,0),height=0.1,width=4,length=4,opacity=0.4) ### mesa, instancia de la clase box
bola1=sphere(color=color.red,pos=vector(rho*cos(phi),0,rho*sin(phi)),radius=0.1,make_trail=True) ### masa m1, instancia de la clase sphe
bola2=sphere(color=color.magenta,pos=vector((L-rho)*sin(phi2),-(L-rho)*cos(phi2),0),radius=0.1,make_trail=True)  ## masa m2

cuerda=curve(pos=[bola1.pos,vector(0,0,0)],color=color.blue)  ### crea una curva llamada cuerda, con dos puntos
cuerda.append(bola2.pos)                                ### agrega un punto a la curva llamada cuerda

# graph(scroll=True, fast=False, xmin=0, xmax=5,ymin=0,ymax=1.1)
# fig = gcurve()
t=0                                                    ### Asigna a la variable t el valor 0.
dt=0.001                                                ### Asigna a la variable dt el valor 0.01
while True:                           # Lazo de iteración sin salida ( Mientras Verdadero haga:), notar los : y el indentado)
   rate(500)                         # Controla la taza de muestreo, normalmente hacer que dt*rate sea del orden de la unidad

   rho_dotdot=(1/(1.+s0))*(rho*phi_dot*phi_dot-s0*(L-rho)*(phi2_dot*phi2_dot)-s0*g*cos(phi2))
   phi_dotdot=-(2/rho)*phi_dot*rho_dot                  # Pendientes en t_i
   phi2_dotdot=(1/((L-rho)*(L-rho)))*(-g*(L-rho)*sin(phi2)+2*(L-rho)*phi2_dot*rho_dot)

   rho=rho+rho_dot*dt
   phi=phi+phi_dot*dt      #  Euler forward    variables en t_i+1
   phi2=phi2+phi2_dot*dt

   rho_dot=rho_dot+rho_dotdot*dt
   phi_dot=phi_dot+phi_dotdot*dt     #  Euler forward   variables en t_i+1
   phi2_dot=phi2_dot+phi2_dotdot*dt

   bola1.pos=rho*vector(cos(phi),0,sin(phi))          ## modifica la posicion de la bola 1
   bola2.pos=(L-rho)*vector(sin(phi2),-cos(phi2),0)
   cuerda.modify(0,bola1.pos)                     ## modifica la posicion del primer punto de la cuerda
   cuerda.modify(2,bola2.pos)                     ## modifica la posicion del tercer punto de la cuerda

#   fig.plot( t, m1*rho*rho*phi_dot )

   t+=dt                                         ## asignacion del nuevo tiempo t=t+dt
```

In [ ]:

In [ ]: