Actividad 10

Andrés Ignacio Rodríguez Mendoza

Introducción

En esta práctica se elaboran animaciones del espacio fase y el espacio físico para varias condiciones iniciales de un péndulo simple. Se utiliza como base el código de ejemplo del espacio físico de un péndulo doble.

El código del péndulo doble se modifica para convertirlo en un péndulo simple considerando que el péndulo doble se comporta como uno simple cuando la masa de la segunda partícula es igual a cero. Haciendo el cálculo para este valor, las ecuaciones del péndulo quedan desacopladas de la segunda partícula y para nuestros propósitos solo usamos los resultados correspondientes a la primera masa en la integración.

A la gráfica que se reproduce en la animación se le modifica para eliminar la segunda cuerda del péndulo y se agrega otra gráfica en la misma animación para simular el espacio fase con un punto trasladándose en las variables de la velocidad angular ω y el ángulo θ como coordenadas.

Código

```
# Double pendulum formula translated from the C code at
# http://www.physics.usyd.edu.au/~wheat/dpend_html/solve_dpend.c

from numpy import sin, cos
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import matplotlib.animation as animation

G = 9.8  # acceleration due to gravity, in m/s^2
L1 = 1.0  # length of pendulum 1 in m
L2 = 1  # length of pendulum 2 in m
M1 = 1.0  # mass of pendulum 1 in kg
M2 = 0.0  # mass of pendulum 2 in kg



def derivs(state, t):

    dydx = np.zeros_like(state)
    dydx[0] = state[1]

    del_ = state[2] - state[0]
    den1 = (M1 + M2)*L1 - M2*L1*cos(del_)*cos(del_)
    dydx[1] = (M2*L1*state[1]*state[1]*sin(del_)*cos(del_) +
               M2*G*sin(state[2])*cos(del_) +
```

```
                M2*L2*state[3]*state[3]*sin(del_) -
                (M1 + M2)*G*sin(state[0]))/den1

    dydx[2] = state[3]

    den2 = (L2/L1)*den1
    dydx[3] = (-M2*L2*state[3]*state[3]*sin(del_)*cos(del_) +
                (M1 + M2)*G*sin(state[0])*cos(del_) -
                (M1 + M2)*L1*state[1]*state[1]*sin(del_) -
                (M1 + M2)*G*sin(state[2]))/den2

    return dydx

# create a time array from 0..100 sampled at 0.05 second steps
dt = 0.05
t = np.arange(0.0, 10, dt)

# th1 and th2 are the initial angles (degrees)
# w10 and w20 are the initial angular velocities (degrees per second)
th1 = 135.0
w1 = 0.0
th2 = -10.0
w2 = 0.0

# initial state
state = np.radians([th1, w1, th2, w2])

# integrate your ODE using scipy.integrate.
y = integrate.odeint(derivs, state, t)

x1 = L1*sin(y[:, 0])
y1 = -L1*cos(y[:, 0])

x2 = L2*sin(y[:, 2]) + x1
y2 = -L2*cos(y[:, 2]) + y1

#Y=integrate.odeint(pend,[th1,w1],t)

th= y[:, 0]
w = y[:,1]

# gráficas
fig1 = plt.figure(figsize=(6, 6.1))

ax1 = fig1.add_subplot(212, autoscale_on=False, xlim=(-2, 2), ylim=(-1.05, 1.05))
ax2 = fig1.add_subplot(211, autoscale_on=False, xlim=(-2, 2), ylim=(-1.05, 1.05))

ax1.axis('off')
ax2.axis('off')
```

```
ax2.plot([-2,2],[0,0],'k',lw=1)
plt.title(r'$\theta___0=___%i___^o_\qquad_\omega___0=_%s$' %(th1,w1), fontsize=18)

line1,    = ax1.plot([], [], '*', color = 'y',     ms=13)
line1_1, = ax1.plot([], [], '-', color='g',       lw=0.8)
line1_0, = ax1.plot([], [], 'o', color='k',       lw=2)
line2,    = ax2.plot([], [], 'H', color='sienna', ms=8)
line2_1, = ax2.plot([], [], '-', color='c',       lw=0.4)
line2_0, = ax2.plot([], [], 'o', color='k',       lw=2)



time_template = 'time_=_%.1fs'
time_text = ax2.text(0.05, 0.9, '', transform=ax2.transAxes)


def init():
    line1.set_data([], [])
    time_text.set_text('')
    return line1, time_text

def init2():
    line2.set_data([], [])
    time_text.set_text('')
    return line1, time_text


def animate(i):
    thisx = [0, x1[i]]
    thisy = [0, y1[i]]

    thisth = [0, th[i]/np.pi]
    thisw = [0, w[i]/ (6*np.sqrt(2*G*(1-np.cos(th1))))]

    line1.set_data(thisx, thisy)
    line1_1.set_data(thisx, thisy)
    line1_0.set_data(0, 0)
    line2.set_data(thisth, thisw)
    line2_0.set_data(0, 0)
    line2_1.set_data(th/np.pi,w/ (6 * np.sqrt(2*G*(1-np.cos(th1)))))
#    time_text.set_text(time_template %(i*dt))
    return line1, line2, time_text


ani = animation.FuncAnimation(fig1,animate, np.arange(1, len(y)),
                              interval=25, blit=True, init_func=init)

ani.save('pendulum_%i_%s.mp4'%(th1,w1), fps=15)
```

```
plt.show()
```